Machine Learning Analysis of the in vehicle coupon recommendation system

Yufei Sun

206-591-1387

sun.yufe@northeastern.edu

Percentage of Effort Contributed by student: 100%

Signature of Student: Yufei Sun

Submission Date: 12/15/2023

Introduction

In this project, the primary focus is to optimize the targeting of various kinds of coupons to a broader audience, with a specific emphasis on recommending coupons to drivers in ideal scenarios. Presently, car owners in the United States generally possess higher disposable incomes than non-car owners, making them more inclined to dine out and take advantage of coupons, especially while on the go. Additionally, drivers enjoy greater convenience in accessing restaurants compared to passengers without predefined destinations, as they can make impromptu stops, such as for a quick coffee. Passengers, on the other hand, are less likely to introduce intermediate stops during their trips. Furthermore, strategically tailored coupons can assist travelers in finding suitable dining options while on the road, enhancing their overall experience. Implementing a coupon recommendation system in vehicles proves mutually beneficial, serving the interests of both drivers and recommended restaurants, resulting in a win-win solution. The crux of this business challenge lies in identifying the specific situations and scenarios wherein drivers are most inclined to accept and utilize coupons. The dataset, sourced from a survey on Amazon Mechanical Turk, comprises approximately 12,684 data points across 14 columns. These columns encompass various factors such as destination, weather, time, occupation, among others, all of which have the potential to influence drivers' decisions and give rise to distinct driving scenarios. Notably, the final column in the dataset indicates whether the driver opted to use the coupon or not, serving as a straightforward and insightful metric reflecting drivers' choices in this context.

Problem Statement

Considering realistic factors like weather, the dataset analysis has led to several hypotheses: Weather Influence: Weather conditions can significantly impact people's dining choices, with favorable weather potentially encouraging outdoor or leisurely dining. Directional Alignment: This alignment between dining and destination could be more convenient and appealing to travelers. Income and Coupon Usage: A higher disposable income may make individuals more open to dining out and taking advantage of discounts. The intention is to employ a multifaceted analytical approach to leverage this dataset effectively. Firstly, data cleaning and exploratory data analysis (EDA) will be the initial steps, enabling a clear understanding of the dataset's characteristics. Subsequently, I will construct multiple machine learning perspectives including the unsupervised learning and supervised learning to build up the models. After that, using the summary of the dataset to validate the models and test if it is the most suitable one as well as test the hypotheses. My machine learning approach is that after testing the data, the system could analyze the current driving scenario and recommend the most appropriate coupon to the driver. The system can be more “Artificial Intelligence” is the final goal that the project needs to achieve.If the test is successful, the recommendation system could find out the drivers’ preference and recommend the ideal coupon which can help the system to improve the operating efficiency. There is also a greater possibility of rolling out cars of all kinds, making more profit for companies and restaurants, even the advertising industry. More importantly, it will bring more convenience to our daily lives.

Dataset

The name of the dataset that I am using is in-vehicle coupon recommendation.csv, providing different factors of driving scenarios. After reading it into the python, I changed the name into ivcr for future reference. There are 12684 rows and 25 columns in the dataset.

In the below are specific detail interpretation for each column name:

Column Name	Column Content	Reference
Destination	No Urgent Place, Home, Work	
Passenger	Alone, Friend(s), Kid(s), Partne	Who are the passengers in the car
Weather	Sunny, Rainy, Snowy	
Temperature	55, 80, 30	
Time	2PM, 10AM, 6PM, 7AM, 10PM	

Coupon Restaurant	Coffee House, Carry out & Take away, Bar, Restaurant($20-$50)	Restaurant Type
Expiration	1d, 2h	the coupon expires in 1 day or in 2 hours
Gender	Female, Male	
Age	21, 46, 26, 31, 41, 50plus, 36, below21	
Marital Status	Unmarried partner, Single, Married partner, Divorced, Widowed	
Has Children	1, 0	1 represents yes, 0 represents no
Education	no degree, Bachelors degree, Associates degree, High School Graduate, Graduate	

	degree (Masters or Doctorate), Some High School	
Occupation	Unemployed, Architecture & Engineering, Student, Education&Training&Library , Healthcare Support, Healthcare Practitioners & Technical, Sales & Related, Management, Arts Design Entertainment Sports & Media, Computer & Mathematical, Life Physical Social Science, Personal Care & Service, Community & Social Services, Office & Administrative Support,	

	Construction & Extraction, Legal, Retired, Installation Maintenance & Repair, Transportation & Material Moving, Business & Financial, Protective Service, Food Preparation & Serving Related, Production Occupations, Building & Grounds Cleaning & Maintenance, Farming Fishing & Forestry	
Income	\$37500 - \$49999, \$62500 - \$74999, \$12500 - \$24999, \$75000 - \$87499, \$50000 - \$62499, \$25000 - \$37499, \$100000 or More,	

	$87500 - $99999, Less than $12500	
Bar	never, less1, 1~3, gt8, nan4~8	how many times do you go to a bar every month
Coffee House	never, less1, 1~3, gt8, nan4~8	how many times do you go to a coffeehouse every month
Carry Away	4~8, 1~3, gt8, less1, never	how many times do you get take-away food every month
Restaurant Less than 20	4~8, 1~3, less1, gt8, never	how many times do you go to a restaurant with an average expense per person of less than $20 every month
Restaurant 20 to 50	4~8, 1~3, less1, gt8, never	how many times do you go to a restaurant with average expense per person of $20 - $50 every month

To Coupon GEQ 5min	1, 0	driving distance to the restaurant/bar for using the coupon is greater than 5 minutes
To Coupon GEQ 15 min	1, 0	driving distance to the restaurant/bar for using the coupon is greater than 15 minutes
To Coupon GEW 25 min	1, 0	driving distance to the restaurant/bar for using the coupon is greater than 25 minutes
Direction Same	1, 0	whether the restaurant/bar is in the same direction as your current destination

Direction Opp	1, 0	whether the restaurant/bar is in the same direction as your current destination
Y	1, 0	whether the coupon is used

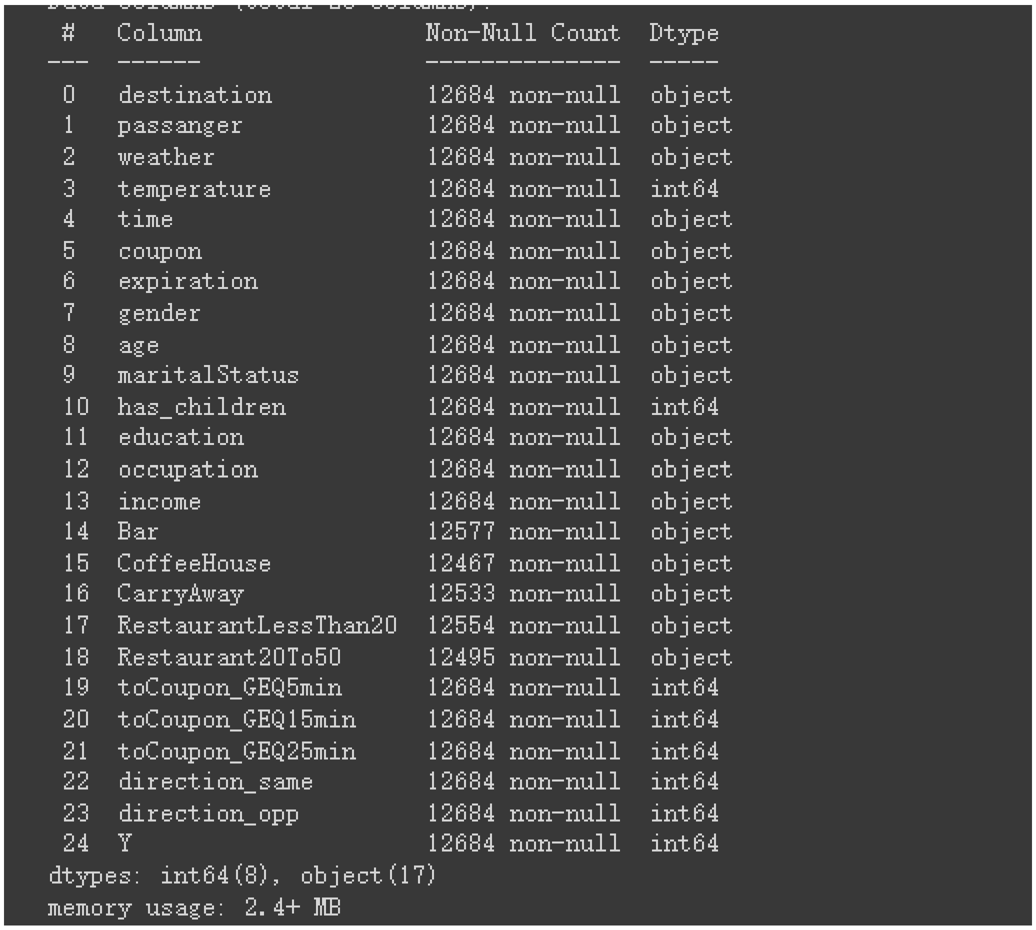
Table1. Introduction to the dataset

Originally, Around 17 of column types are objects, 7 of columns are int64 and 1 of columns is category. These column types are subject to change due to further analysis needed. For instance, the Y column represents whether the coupon is used. I change the column name into UsedorNot and replace the content from (1, 0) to (Used, Not) for clarification.

From the analysis of the data set, we can see that there are various variables and factors that will affect the behavior of the driver. Combined with the actual cases, it is human nature that the driver will be affected by these factors. From the subsequent EDA analysis, we can see how it supports my hypothesis.

EDA

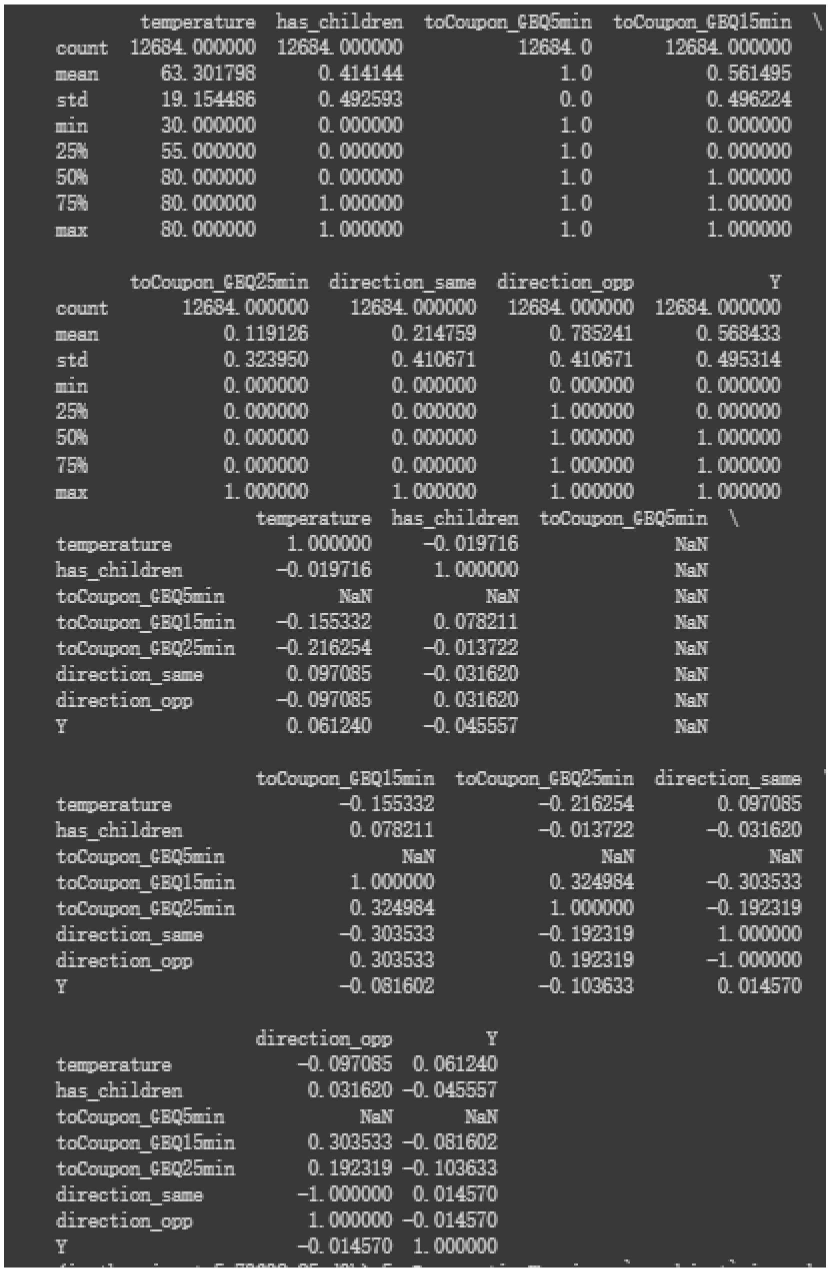
After reading the dataset into python, I use describe(), info(), corr() function to get a basic look of the dataset.



```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   destination           12684 non-null  object
 1   passanger             12684 non-null  object
 2   weather               12684 non-null  object
 3   temperature           12684 non-null  int64
 4   time                  12684 non-null  object
 5   coupon                12684 non-null  object
 6   expiration            12684 non-null  object
 7   gender                12684 non-null  object
 8   age                   12684 non-null  object
 9   maritalStatus         12684 non-null  object
 10  has_children          12684 non-null  int64
 11  education             12684 non-null  object
 12  occupation            12684 non-null  object
 13  income                12684 non-null  object
 14  Bar                   12577 non-null  object
 15  CoffeeHouse           12467 non-null  object
 16  CarryAway             12533 non-null  object
 17  RestaurantLessThan20  12554 non-null  object
 18  Restaurant20To50      12495 non-null  object
 19  toCoupon_GEQ5min      12684 non-null  int64
 20  toCoupon_GEQ15min     12684 non-null  int64
 21  toCoupon_GEQ25min     12684 non-null  int64
 22  direction_same        12684 non-null  int64
 23  direction_opp         12684 non-null  int64
 24  Y                     12684 non-null  int64
dtypes: int64(8), object(17)
memory usage: 2.4+ MB
```

Image1. Overview of the columns

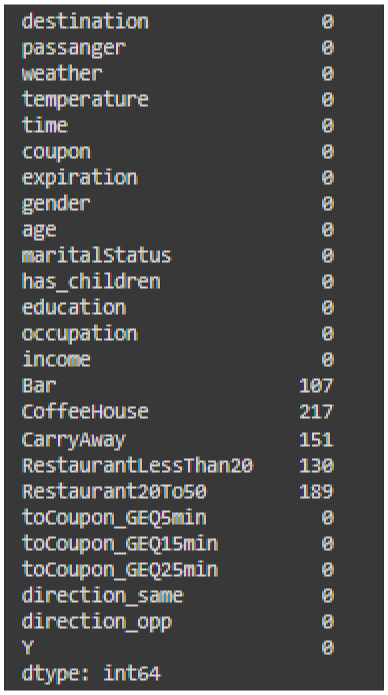
As I mentioned previously, there are 12684 rows and 25 columns in the dataset. We can generate the column name, count, mean, std, minimum, maximum, quantile of the columns. Since this dataset contains many categorical columns, the correlation might not be applicable in this phase.



```
        temperature  has_children  toCoupon_GEQ5min  toCoupon_GEQ15min  \
count  12684.000000  12684.000000           12684.0       12684.000000
mean      63.301798      0.414144               1.0           0.561495
std       19.154486      0.492593               0.0           0.496224
min       30.000000      0.000000               1.0           0.000000
25%       55.000000      0.000000               1.0           0.000000
50%       80.000000      0.000000               1.0           1.000000
75%       80.000000      1.000000               1.0           1.000000
max       80.000000      1.000000               1.0           1.000000

       toCoupon_GEQ25min  direction_same  direction_opp             Y
count       12684.000000    12684.000000   12684.000000  12684.000000
mean            0.119126        0.214759       0.785241      0.568433
std             0.323950        0.410671       0.410671      0.495314
min             0.000000        0.000000       0.000000      0.000000
25%             0.000000        0.000000       1.000000      0.000000
50%             0.000000        0.000000       1.000000      1.000000
75%             0.000000        0.000000       1.000000      1.000000
max             1.000000        1.000000       1.000000      1.000000
                   temperature  has_children  toCoupon_GEQ5min  \
temperature           1.000000     -0.019716               NaN
has_children         -0.019716      1.000000               NaN
toCoupon_GEQ5min           NaN           NaN               NaN
toCoupon_GEQ15min    -0.155332      0.078211               NaN
toCoupon_GEQ25min    -0.216254     -0.013722               NaN
direction_same        0.097085     -0.031620               NaN
direction_opp        -0.097085      0.031620               NaN
Y                     0.061240     -0.045557               NaN

                   toCoupon_GEQ15min  toCoupon_GEQ25min  direction_same  \
temperature                -0.155332          -0.216254        0.097085
has_children                0.078211          -0.013722       -0.031620
toCoupon_GEQ5min                 NaN                NaN             NaN
toCoupon_GEQ15min           1.000000           0.324984       -0.303533
toCoupon_GEQ25min           0.324984           1.000000       -0.192319
direction_same             -0.303533          -0.192319        1.000000
direction_opp               0.303533           0.192319       -1.000000
Y                          -0.081602          -0.103633        0.014570

                   direction_opp         Y
temperature            -0.097085  0.061240
has_children            0.031620 -0.045557
toCoupon_GEQ5min             NaN       NaN
toCoupon_GEQ15min       0.303533 -0.081602
toCoupon_GEQ25min       0.192319 -0.103633
direction_same         -1.000000  0.014570
direction_opp           1.000000 -0.014570
Y                      -0.014570  1.000000
```

Image2. Statistics of numeric columns

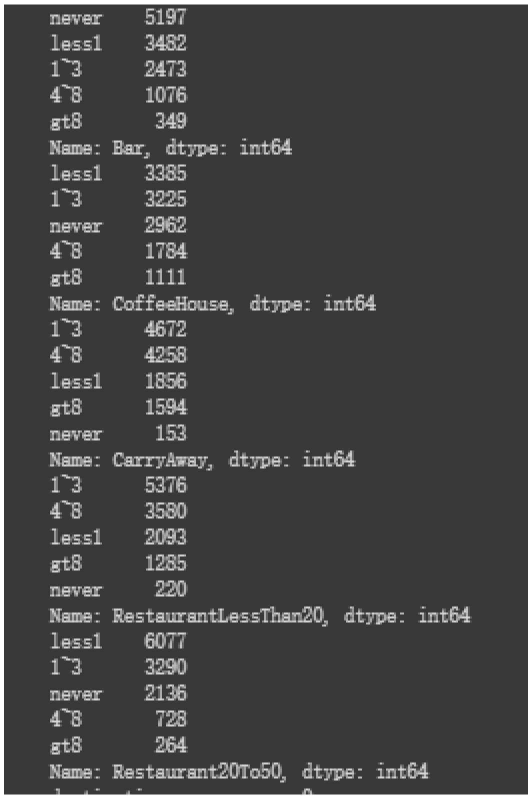
The second step is to check the missing value with the isnull().sum() function. Unfortunately, there are around 100 to 200 missing values in 5 columns.



```
destination               0
passanger                 0
weather                   0
temperature               0
time                      0
coupon                    0
expiration                0
gender                    0
age                       0
maritalStatus             0
has_children              0
education                 0
occupation                0
income                    0
Bar                     107
CoffeeHouse             217
CarryAway               151
RestaurantLessThan20    130
Restaurant20To50        189
toCoupon_GEQ5min          0
toCoupon_GEQ15min         0
toCoupon_GEQ25min         0
direction_same            0
direction_opp             0
Y                         0
dtype: int64
```

Image3. Missing value detection

In order to make the dataset more integrated. I am filling the missing value with each columns' mean or median. Besides, I also change the column name into UsedorNot and replace the content from (1, 0) to (Used, Not) for clarification.



```
never    5197
less1    3482
1~3      2473
4~8      1076
gt8       349
Name: Bar, dtype: int64
less1    3385
1~3      3225
never    2962
4~8      1784
gt8      1111
Name: CoffeeHouse, dtype: int64
1~3      4672
4~8      4258
less1    1856
gt8      1594
never     153
Name: CarryAway, dtype: int64
1~3      5376
4~8      3580
less1    2093
gt8      1285
never     220
Name: RestaurantLessThan20, dtype: int64
less1    6077
1~3      3290
never    2136
4~8       728
gt8       264
Name: Restaurant20To50, dtype: int64
```

Image4. Count of missing value column

After the data cleaning and preparation, I pick some essential columns to perform the explanatory data analysis. For example, the usage of coupons is the core. I used a pie chart and a histogram to visualize the usage. 57% of people use the coupons which means over half of people accept and are willing to use it.

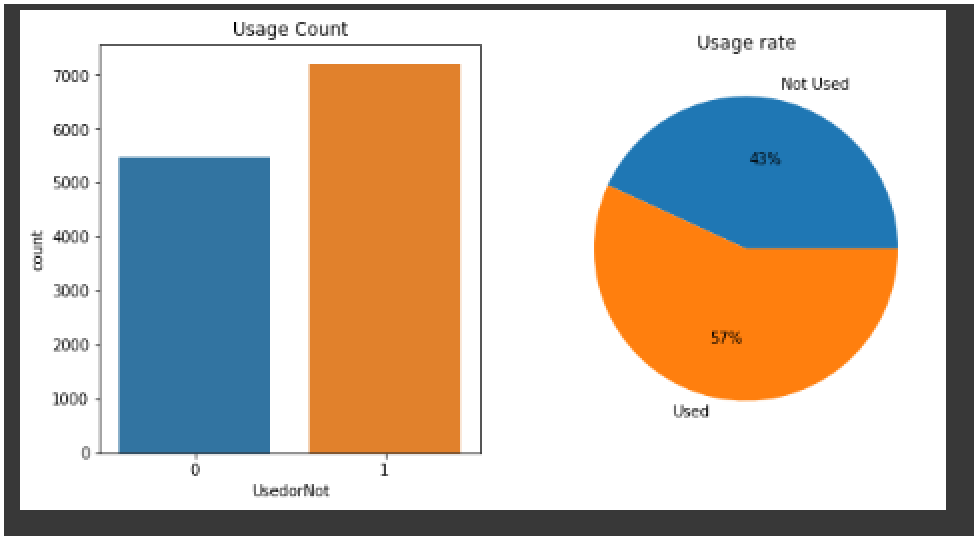
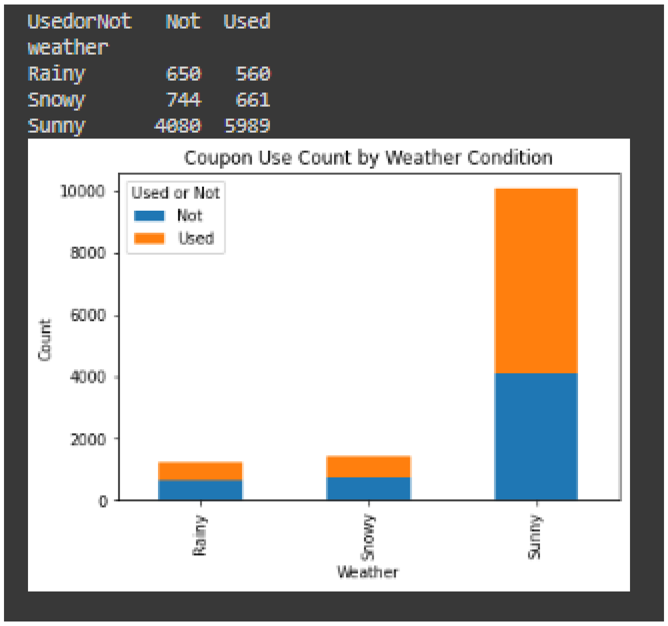


Image5. Count of usage of coupon

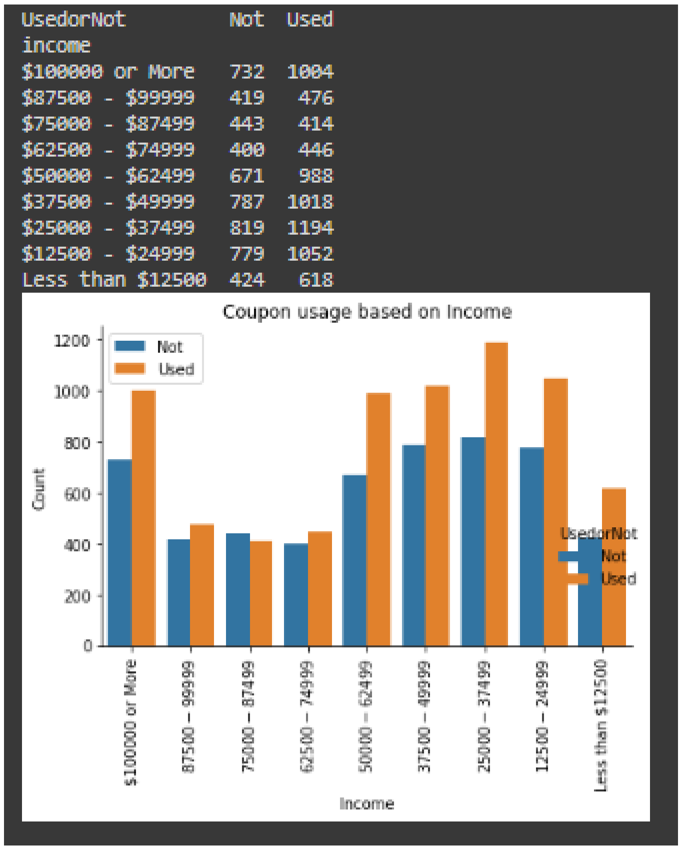
Therefore, I pick 3 columns that might influence the usage to do the analysis. The first one is the weather column, there are three types of weather: Sunny, Snowy, Rainy. The second one is the income column, there are 9 distributions from $100000 to $12500 with $12499 interval. The third one is the gender column which contains male and female. Then I perform the “weather - Coupon Usage”, “income - Coupon Usage”, and “gender - Coupon Usage” analysis. According to the result, People are tend to use the coupon if the weather is Sunny.



```
UsedorNot   Not  Used
weather
Rainy       650   560
Snowy       744   661
Sunny      4080  5989
```

Image6. Count of usage in different weather

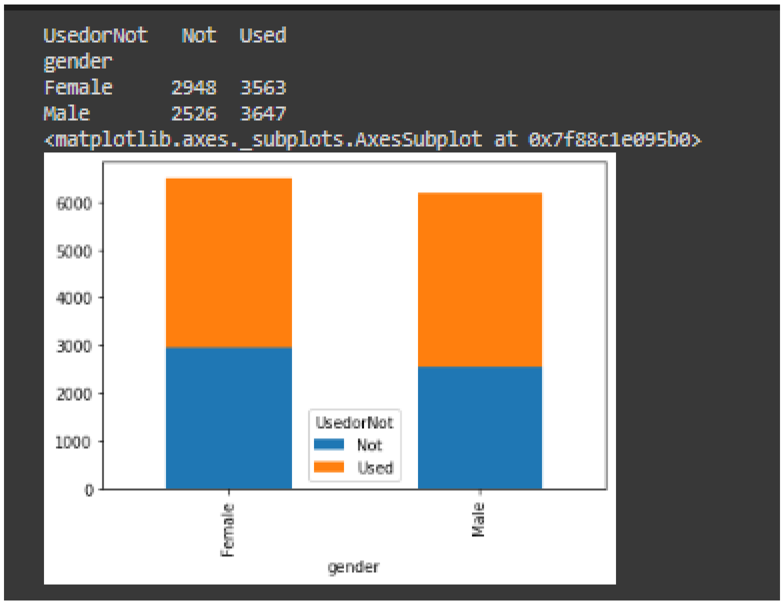
People whose income is between $25000 to $37499 are more likely to go to the restaurant and use the coupon.



```
UsedorNot          Not  Used
income
$100000 or More    732  1004
$87500 - $99999    419   476
$75000 - $87499    443   414
$62500 - $74999    400   446
$50000 - $62499    671   988
$37500 - $49999    787  1018
$25000 - $37499    819  1194
$12500 - $24999    779  1052
Less than $12500   424   618
```

Image7. Distribution of Income

Male is more interested in using coupons than the female.



```
UsedorNot   Not  Used
gender
Female     2948  3563
Male       2526  3647
<matplotlib.axes._subplots.AxesSubplot at 0x7f88c1e095b0>
```

Image8. Count of usage of Sex

I plan to detect more factors which might have an impact on the decision. I would like to test gender, income, age, marital Status, and has_children for my business problem. These factors will be considered as demographic information. Also, weather, time and temperature are in my consideration for the nature factors. Besides, the direction to the coupon restaurant, driving distance, and the coupon type are also important features to determine the drivers' behavior. After completing the cross columns analysis, I plan to build a linear regression model to detect their relations.

There are a few more steps in the EDA part.

The first step is the imputation: eliminate the missing value. After reading in the data, using isnull().sum() function to check if there is any missing. There are 107 missing values in the Bar column, 217 missing values in the CoffeeHouse column, 151 missing values in the CarryAway column, 130 missing values in the RestaurantLessThan20 column, 189 missing values in the Restaurant20To50 columns.

In order to make the dataset more integrated. I am filling the missing value with each columns' mean or median.

Then check it again to see if any left.

Modeling

In this section, the following is performing 4 types of model. They are adaboost, decision tree, Knn and neural network. Since this dataset is a classification dataset. Here are some reasons why I chose these models as well as the structure and methodology.

Adaboost:

Sequential Improvement: AdaBoost is particularly effective when dealing with complex datasets where simple models fall short. It sequentially applies weak learners, improving upon areas where previous learners were inaccurate. This adaptiveness makes it powerful for datasets with intricate decision boundaries or those with nuanced distinctions between classes.

Handling Imbalanced Data: AdaBoost is known for its ability to focus on difficult-to-classify instances, which often are the minority class in an imbalanced dataset. By emphasizing these instances in successive training rounds, AdaBoost can improve classification performance in imbalanced settings, which might be relevant if the dataset has disproportionate class representation.

Feature Interaction: AdaBoost can capture complex interactions between features by iteratively refining its understanding of the data. In the dataset, interactions between variables like passenger, weather, and coupon type may be crucial in predicting coupon acceptance.

Model Design and Overview

DecisionStump Class:

DecisionStump is a simple weak learner used within the AdaBoost algorithm.

It makes binary predictions (either +1 or -1) based on a single feature of the input data.

The key attributes of this class are:

polarity: Represents whether the decision stump classifies values below a certain threshold as +1 or -1.

feature_index: The index of the feature in the dataset used for classification.

threshold: The threshold value used to split the feature into two classes.

alpha: The weight or importance of this weak learner in the ensemble.

AdaBoost Class:

AdaBoost is the main ensemble classifier that combines multiple weak learners to create a strong classifier.

It uses decision stumps as weak learners by default, but other weak learners could be used as well.

The main attributes and methods of this class are:

n_learners: The number of weak learners (decision stumps) to train in the ensemble.

learners: A list to store the trained decision stumps (weak learners).

Decision Tree:

Capturing Non-Linear Relationships: Decision trees do not assume any linearity in the data. They can capture complex, non-linear relationships between features and the target variable which is beneficial.

Variable Importance: Decision trees provide clear insights on which features are most influential in predicting the target variable. This can be extremely useful for understanding the dynamics of the dataset, such as which aspects most strongly influence coupon acceptance.

Prone to Overfitting: Although decision trees are powerful, they can easily overfit the training data, especially if they grow deep with many branches. This overfitting can be mitigated by pruning the tree or limiting its depth.

Model Design and Overview

Decision Tree Node (Node Class):

The Node class represents a node in the Decision Tree. Each node can have the following attributes:

feature: The index of the feature used for splitting at this node.

threshold: The threshold value for the feature to determine the split.

left: A reference to the left child node.

right: A reference to the right child node.

value: The class label (target value) assigned to this node if it's a leaf node.

Decision Tree (DecisionTree Class):

The DecisionTree class represents the Decision Tree classifier.

It has the following attributes and methods:

max_depth: The maximum depth of the tree, a hyperparameter that limits the tree's growth.

fit: The method for fitting (training) the Decision Tree on the input data.

calculate_gini: A method to calculate the Gini impurity, used for evaluating splits.

predict: The method for making predictions on new data.

_predict_tree: A recursive helper method used by the predict method to traverse the tree and make predictions.

Fit Method:

The fit method builds the Decision Tree recursively. It takes the input features X and labels y as input, along with an optional depth parameter.

It checks whether a node should be a leaf node based on conditions like pure class labels or reaching the maximum depth.

It searches for the best split by iterating over features and thresholds to minimize the Gini impurity.

If a split reduces the Gini impurity, it continues to create child nodes, otherwise, it creates a leaf node.

The method returns the root node of the Decision Tree.

Calculate Gini Impurity:

The calculate_gini method computes the Gini impurity of a set of labels. It's used to measure the impurity or disorder of a set of class labels.

Predict Method:

The predict method takes a set of features X and returns predictions for each input based on the trained Decision Tree.

It iterates over each input and uses the _predict_tree method to traverse the tree and make predictions.

Recursive Prediction (_predict_tree Method):

The _predict_tree method is a recursive helper function used by the predict method to navigate the Decision Tree.

It checks whether the current node is a leaf node. If so, it returns the class label associated with that leaf.

If not, it compares the feature value of the input data with the node's threshold and traverses either the left or right child node accordingly.

Knn:

Simple Yet Effective: KNN's simplicity lies in its instance-based learning. The algorithm assumes that similar instances will be near each other.

Feature Scaling Importance: KNN relies on distance calculations, so proper feature scaling is crucial. Variables on larger scales can unduly influence the model, leading to poor performance.

Curse of Dimensionality: With a high number of features, KNN's performance can degrade.

Model Design and Overview

KNNClassifierOptimized Class:

This class is designed for KNN classification with an optimization feature.

It takes the number of neighbors k as a hyperparameter during initialization.

fit Method:

The fit method is used to train the KNN classifier.

It takes the training data X and corresponding labels y as input.

The method stores the training data X_train and labels y_train for later use.

It also constructs a cKDTree from the training data, which is an optimized data structure for efficient nearest neighbor queries.

predict Method:

The predict method takes a set of data points X and returns predictions for each input.

For each data point in X, it uses the cKDTree to efficiently find the k nearest neighbors.

It then calls the _predict method to determine the class label based on the majority class among the nearest neighbors.

_predict Method:

The _predict method takes the indices of the k nearest neighbors and returns the predicted class label for a single data point.

It extracts the labels of the nearest neighbors and counts the occurrences of each label.

The predicted label is the class label that occurs most frequently among the k nearest neighbors.

Neural Network:

Complex Non-Linear Relationships: Neural networks excel at capturing complex, non-linear relationships in data. The dataset likely contains intricate patterns (interaction effects between different features like weather, time of day, personal preferences, etc.), a neural network can learn these subtleties more effectively than simpler models.

High-Dimensional Data: If the dataset has a large number of features, neural networks can handle this high dimensionality well, especially deep learning models with multiple hidden layers.

Feature Interactions: Neural networks can automatically detect and learn interactions between different features without manual feature engineering. This is particularly useful in a dataset where interactions (e.g., between weather conditions and the type of coupon) might be crucial for predicting coupon acceptance.

Model Design and Overview

Class Initialization (__init__):

layers: Specifies the number of neurons in each layer (input, hidden, output).

learning_rate: The step size at each iteration while moving toward a minimum of a loss function.

iterations: Number of times the learning algorithm will work through the entire training dataset.

activation: Type of activation function to use ('relu' or 'tanh').

optimizer: Method to use for optimization ('gd' for Gradient Descent or 'sgd' for Stochastic Gradient Descent).

Activation Functions:

tanh: Hyperbolic tangent function.

relu: Rectified Linear Unit function.

Weight Initialization (init_weights):

Initializes weights randomly for both layers.

Forward Propagation (forward_propagation):

Computes the output of the neural network for a given input.

Uses the chosen activation function in the hidden layer and sigmoid activation in the output layer.

Loss Calculation (entropy_loss):

Calculates the loss using binary cross-entropy, a common loss function for binary classification problems.

Back Propagation (back_propagation):

Calculates gradients of the loss function with respect to the network's weights.

Adjusts the weights using the calculated gradients and the learning rate.

Training (fit):

Trains the neural network using the training data.

Iterates through the dataset, performing forward and backpropagation, and updates weights.

Supports both Gradient Descent and Stochastic Gradient Descent (SGD) as optimization methods.

Prediction (predict):

Outputs predictions for the input data after the network is trained.

Uses sigmoid activation to output probabilities.

Accuracy Calculation (acc):

Calculates the accuracy of the model by comparing the predicted values to the actual values.

Plotting Loss (plot_loss):

Visualizes the loss of the model over training iterations.

Model Overview:

Two-Layer Neural Network:

The network consists of an input layer (size based on data), one hidden layer, and an output layer with a single neuron (for binary classification).

The number of neurons in the hidden layer is defined by layers[1].

Binary Classification:

The network is designed for binary classification tasks (output either 0 or 1).

Activation Functions:

The hidden layer uses either ReLU or tanh activation.

The output layer uses a sigmoid activation function to output a probability.

Loss Function:

Binary cross-entropy is used to measure the error between the predicted and actual values.

Optimization Methods:

Gradient Descent (GD):

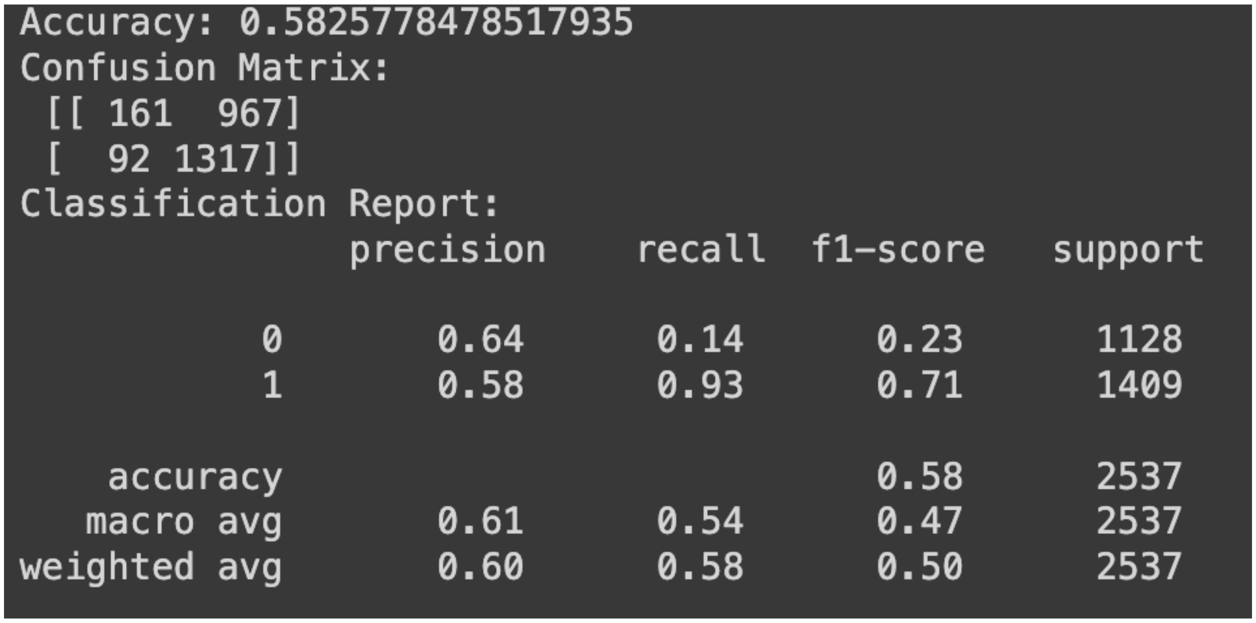
Updates the weights using the whole dataset.

Stochastic Gradient Descent (SGD):

Updates the weights using a single data point at each iteration.

After training these four models, I obtain the following output.

Adaboost:



```
Accuracy: 0.5825778478517935
Confusion Matrix:
 [[ 161  967]
 [  92 1317]]
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.14      0.23      1128
           1       0.58      0.93      0.71      1409

    accuracy                           0.58      2537
   macro avg       0.61      0.54      0.47      2537
weighted avg       0.60      0.58      0.50      2537
```

Image9. Adaboost Output

Overall Interpretation

AdaBoost showed a moderate performance with a tendency to favor predicting positive outcomes (coupon acceptance).

Pros: Good at handling complex relationships and can focus on difficult-to-classify instances, which is beneficial.

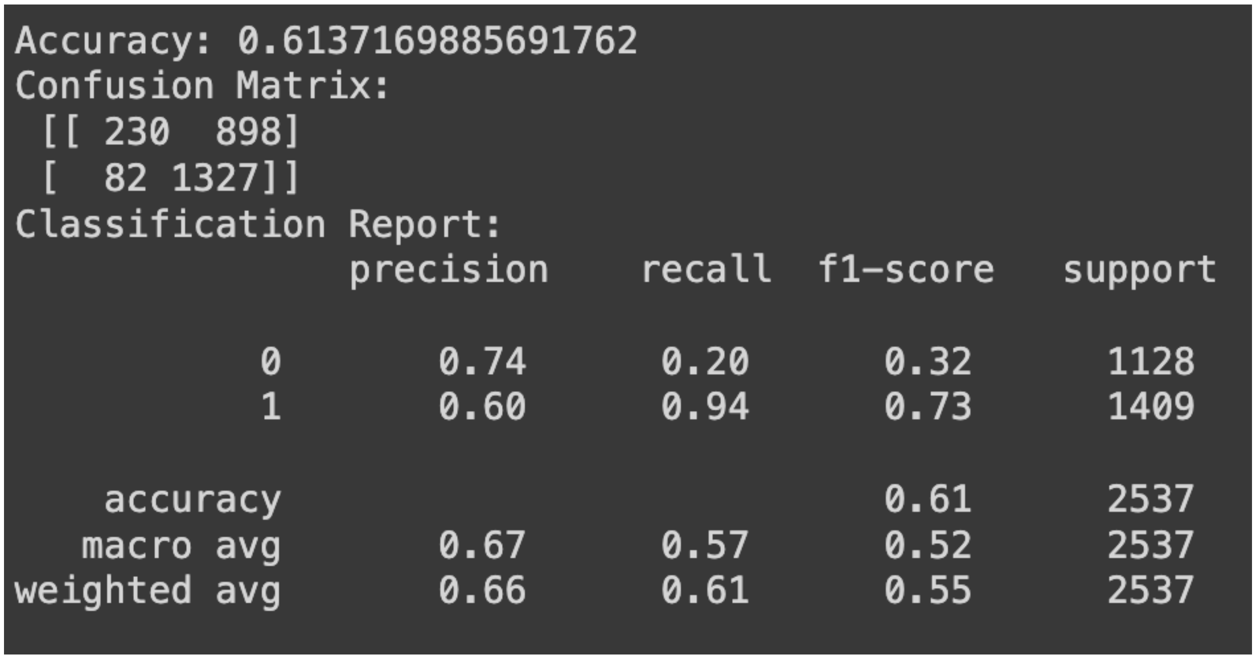
Cons: Bias towards predicting one class (coupon acceptance) indicates it might be overly influenced by the majority class or specific dominating features.

The model is heavily biased towards predicting coupons as accepted (Class 1). This is evident from the high recall for Class 1 and the low recall for Class 0.

The precision for both classes is moderate, but the recall disparity causes the F1-scores to differ significantly.

The model's low performance on classifying non-accepted coupons (Class 0) might be a concern.

Decision Tree:



```
Accuracy: 0.6137169885691762
Confusion Matrix:
 [[ 230  898]
 [  82 1327]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.20      0.32      1128
           1       0.60      0.94      0.73      1409

    accuracy                           0.61      2537
   macro avg       0.67      0.57      0.52      2537
weighted avg       0.66      0.61      0.55      2537
```

Image10. Decision Tree Output

Overall Interpretation

Performance: The Decision Tree model had a moderate accuracy and showed a better balance in predicting both classes compared to AdaBoost, but it still leaned towards predicting coupon acceptance.

Pros: Easy to interpret and understand which features are driving predictions.

Cons: Prone to overfitting and might not capture complex relationships as effectively as ensemble methods.

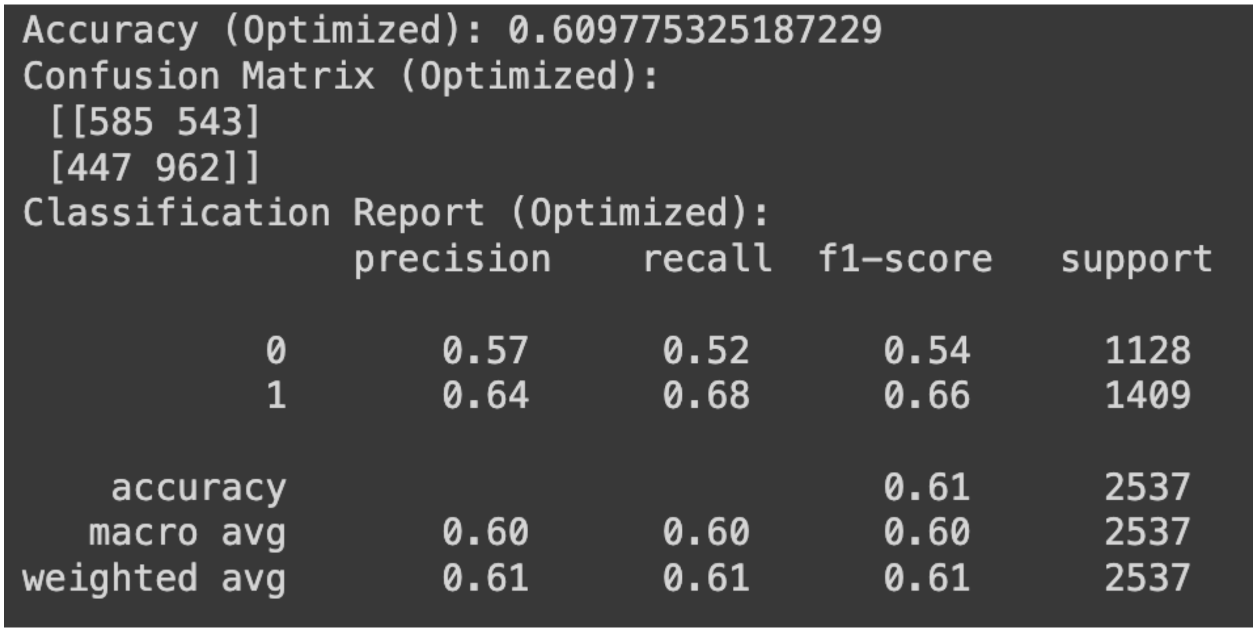
The model exhibits a strong bias towards predicting that coupons will be accepted. This is evident from the high number of False Positives and the high recall for Class 1.

The low recall for Class 0 indicates that the model struggles to correctly identify instances where coupons are not accepted.

The moderate precision for Class 1 and high precision for Class 0 suggest that the model, when it predicts non-acceptance, is usually correct, but it rarely makes this prediction.

The overall accuracy, while moderate, masks these underlying issues in class-specific performance.

Knn:



```
Accuracy (Optimized): 0.609775325187229
Confusion Matrix (Optimized):
 [[585 543]
 [447 962]]
Classification Report (Optimized):
               precision    recall  f1-score   support

           0       0.57      0.52      0.54      1128
           1       0.64      0.68      0.66      1409

    accuracy                           0.61      2537
   macro avg       0.60      0.60      0.60      2537
weighted avg       0.61      0.61      0.61      2537
```

Image11. Knn Output

Overall Interpretation

Performance: KNN displayed the most balanced performance among the models, with moderate accuracy and a relatively equal distribution of true positives and negatives.

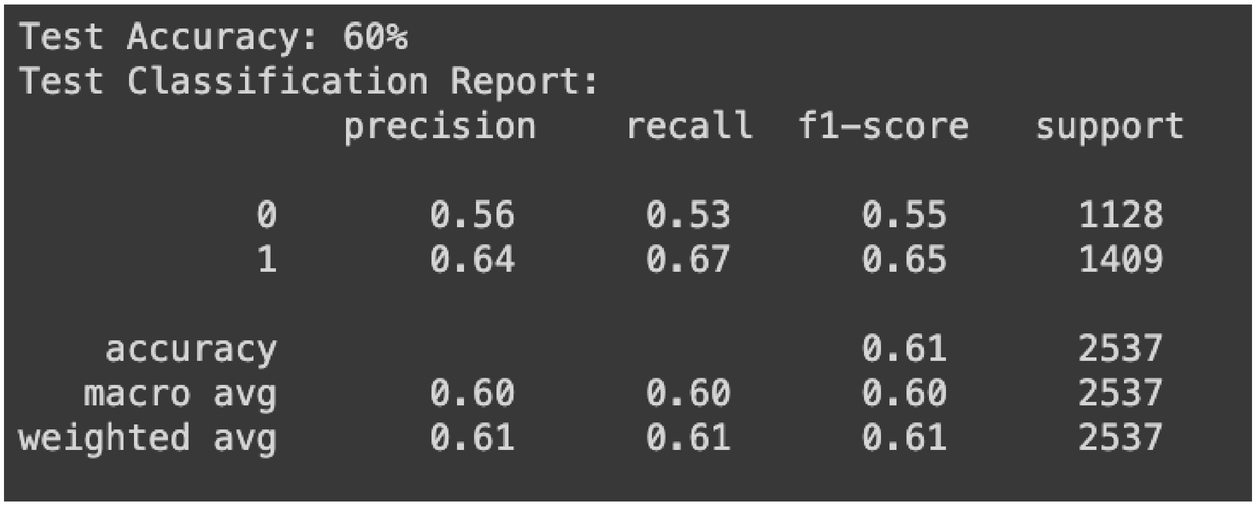
Pros: Simple and effective, especially the dataset contains distinct clusters. Does not make any assumptions about data distribution.

Cons: Sensitive to the choice of 'k' and the scaling of features. May struggle with high-dimensional data.

The model displays a more balanced performance in predicting both classes compared to other models discussed earlier. It is neither heavily biased towards predicting acceptance nor rejection.

The moderate values in precision, recall, and F1-score for both classes suggest that the model is somewhat consistent but not highly accurate in its predictions.
The relatively equal distribution of true positives and true negatives indicates a fair ability to distinguish between the two classes.

Neural Network:



```
Test Accuracy: 60%
Test Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.53      0.55      1128
           1       0.64      0.67      0.65      1409

    accuracy                           0.61      2537
   macro avg       0.60      0.60      0.60      2537
weighted avg       0.61      0.61      0.61      2537
```

Image12. Neural Network Output

Overall Interpretation

Performance: The Neural Network demonstrated a uniform performance across both classes, showing no significant bias toward predicting one class over the other. The accuracy is competitive, suggesting it has a good generalization on the data provided.

Pros:

Can model complex non-linear relationships due to the multiple layers and non-linear activation functions.

Highly flexible and can be tuned by adjusting the number of layers, neurons, and types of activation functions.

With enough data and computational power, it can outperform simpler models.

Cons:

Requires a large amount of data to perform well and to avoid overfitting.

Computationally intensive, especially as the network size grows.

Performance is highly sensitive to the choice of architecture and hyperparameters, which may require extensive experimentation and fine-tuning.

Can be considered a "black box" model, making it difficult to interpret and understand the decision process.

Conclusion & Discussion

Upon comparing all these models, there is the analysis and recommendation for the model.

AdaBoost:

Exhibits moderate performance with a strong bias towards predicting positive outcomes, indicating potential issues with class imbalance or feature dominance.

It is effective for boosting the performance of weak learners and focusing on challenging classifications.

Decision Tree:

Offers an interpretable model structure, providing insights into decision-making through feature importance.

However, it is susceptible to overfitting and may not handle complex relationships as effectively as ensemble or neural network models.

KNN:

Demonstrates a balanced approach to classification, performing comparably across both classes without a strong bias.

The performance of KNN is contingent on the appropriate selection of 'k' and is sensitive to feature scaling. It can also be computationally expensive with high-dimensional data.

Neural Network:

Shows uniform performance with no significant bias towards either class and competitive accuracy, suggesting good generalization capabilities on the provided data.

Neural networks can capture complex, non-linear relationships and are highly tunable but require large datasets and are computationally intensive.

AdaBoost might be preferred when the focus is on minimizing false negatives at the cost of accepting more false positives.

Decision Trees could be chosen for their simplicity and interpretability, particularly when the model's decisions need to be explained easily.

KNN is useful when the dataset has distinct, separable clusters, and computational efficiency is not a primary concern.

Neural Networks are suitable for complex problems where the dataset is sufficiently large and computational resources are available for training and fine-tuning.

Balanced Performance: KNN seems to offer the most balanced performance, with no extreme bias towards one class which indicates a better general understanding of the dataset.

Potential for Improvement: AdaBoost has the potential for better performance with further tuning and addressing the class imbalance issue. However, the current results suggest that they are not effectively capturing the nuances of the dataset.

Primary Choice: Neural Network would be the recommended model for its balanced classification ability and moderate accuracy.
Secondary Choice: Decision Tree, for its interpretability and moderate performance. It could be particularly useful.

Though the neural network model and the decision tree model have very similar performance and minor differences overall, the neural network is slightly better on the score comparison and the test dataset size.
In the future, we can design and implement both a content based recommendation model and the collaborative filtering recommendation model. For example, if a user wants to get a target coupon, a content based recommendation model to recommend the best driving scenario. If an advertising company or restaurants want to advertise the restaurant precisely and efficiently, it can use a collaborative filtering recommendation model by detecting whether the corresponding coupon is used or not. More detailedly, we can add more layers in the neural network model in order to let the network learn more smartly. The business problem tends to be solved for both customers and merchants.

Reference & Appendix

Wang, Tong, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. 'A bayesian framework for learning rule sets for interpretable classification.' The Journal of Machine Learning Research 18, no. 1 (2017): 2357-2393.

https://www.tastyad.com/everything-you-need-to-know-about-programmatic-taxi-screen-advertising/

Full code and Report can find in https://github.com/allensun4/IE7300