In-Vehicle Coupon Recommendation

Machine Learning Final Project

Arianna Tessari - Student ID: 19322

Outline

Introduction

The project presented in this report focuses on the implementation and comparison of different machine learning models aimed at predicting the likelihood that a customer will use a coupon during a car trip. The dataset used is the "In-Vehicle Coupon Recommendation" from the UCI Machine Learning Repository (retrievable from https://archive.ics.uci.edu/dataset/603/in+vehicle+coupon+recommendation), including over 12,000 instances with 25 features. The features are primarily categorical variables that contain user attributes, contextual signals, and coupon specifics. User attributes include demographic details such as gender, age, marital status, and education level, along with consumption patterns like frequency of bar visits, restaurant expenditures, and preferences for takeout food. Contextual attributes refer to factors such as destination, weather conditions, temperature, time of day, and the presence of passengers. Coupon characteristics indicate the remaining time before expiration, usually set at either two hours or one day, with a consistent offer of a 20% discount.

The dataset was collected through a survey on Amazon Mechanical Turk illustrating various driving scenarios, including destination, current weather conditions, passengers, and so on, and asked individuals about their willingness to accept a coupon if they were the driver. Given the dataset's characteristics, the primary objective revolves around predicting whether a driver with specific attributes would accept the offered coupon or not. This essentially translates into a classification task aimed at determining whether a customer will accept a coupon for a particular venue, considering both demographic and contextual attributes.

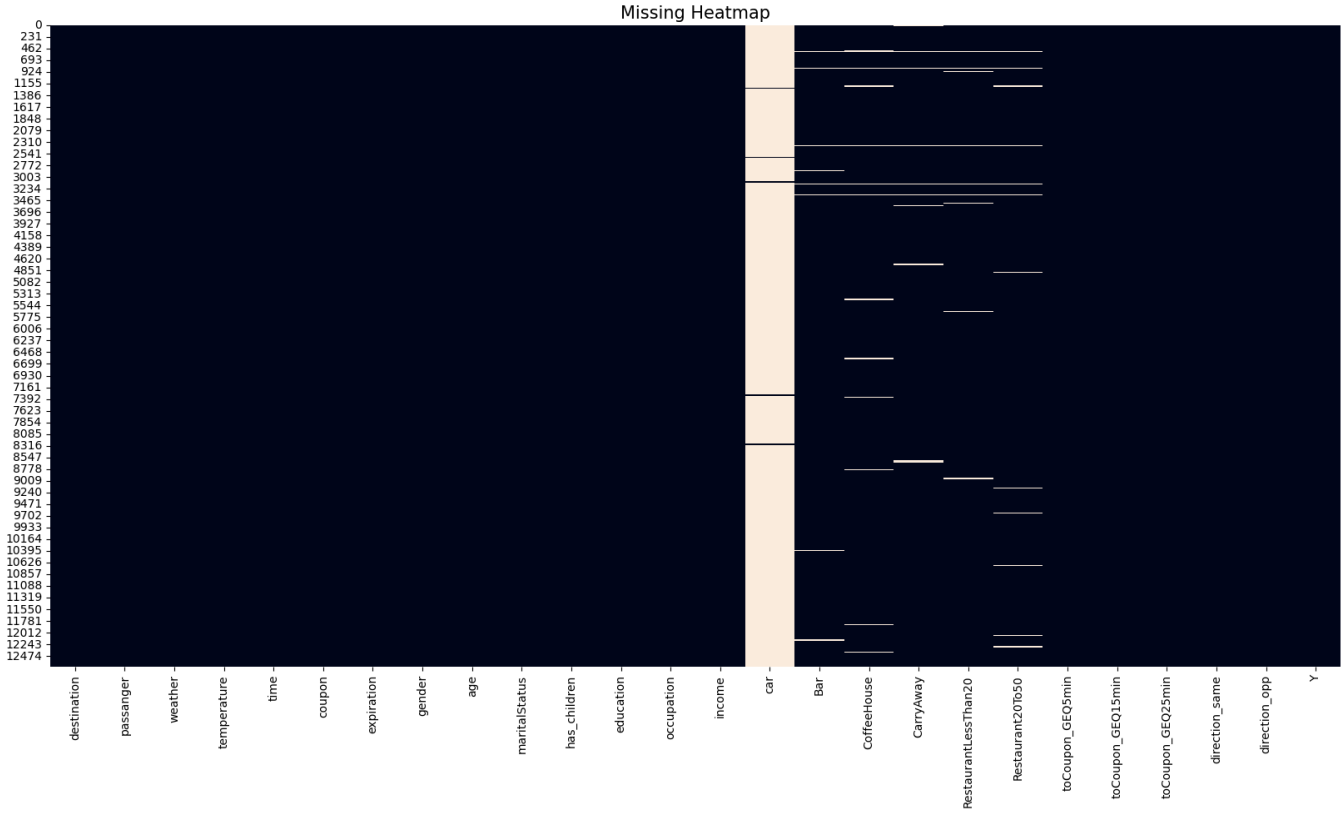
To address this challenge, algorithms tailored for classification were considered, with the main goal to underline two distinct classes: acceptance or non-acceptance.

Data Exploration, Cleaning, and Preprocessing

Approaching a new dataset involves initial exploration to understand its structure and integrity. It is common to encounter missing values during this phase, which can interfere with data analysis. In this specific case, during the initial dataset exploration, it was found that several attributes had missing values, particularly notable in the "car" column (please refer to the missing values heatmap below). Due to the high percentage of missing data, this column was entirely removed. Other columns affected by missing values included "Bar," "CoffeeHouse," "CarryAway," "RestaurantLessThan20," and "Restaurant20To50," which were managed by deleting the corresponding rows.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12684 entries, 0 to 12683
Data columns (total 26 columns):
 #   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  car                   108 non-null    object
 15  Bar                   12577 non-null  object
 16  CoffeeHouse           12467 non-null  object
 17  CarryAway             12533 non-null  object
 18  RestaurantLessThan20  12554 non-null  object
 19  Restaurant20To50      12495 non-null  object
 20  toCoupon_GEQ5min      12684 non-null  int64 
 21  toCoupon_GEQ15min     12684 non-null  int64 
 22  toCoupon_GEQ25min     12684 non-null  int64 
 23  direction_same        12684 non-null  int64 
 24  direction_opp         12684 non-null  int64 
 25  Y                     12684 non-null  int64 
dtypes: int64(8), object(18)
memory usage: 2.5+ MB

Number of missing values:
        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
        car: 12576
        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
```



Out[6]:

destination	0.0%
passanger	0.0%
weather	0.0%
temperature	0.0%
time	0.0%
coupon	0.0%
expiration	0.0%
gender	0.0%
age	0.0%
maritalStatus	0.0%
has_children	0.0%
education	0.0%
occupation	0.0%
income	0.0%
car	99.15%
Bar	0.84%
CoffeeHouse	1.71%
CarryAway	1.19%
RestaurantLessThan20	1.02%
Restaurant20To50	1.49%
toCoupon_GEQ5min	0.0%
toCoupon_GEQ15min	0.0%
toCoupon_GEQ25min	0.0%
direction_same	0.0%
direction_opp	0.0%
Y	0.0%

dtype: object

```
Number of missing values:
        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: 0
        CoffeeHouse: 0
        CarryAway: 0
        RestaurantLessThan20: 0
        Restaurant20To50: 0
        toCoupon_GEQ5min: 0
        toCoupon_GEQ15min: 0
        toCoupon_GEQ25min: 0
        direction_same: 0
        direction_opp: 0
        Y: 0
```

Afterward, to further manage the data structure, a transformation of the columns related to categorical variables, initially detected as "object" type upon dataset loading, was performed into "category." This operation is crucial for subsequent visualization and serves as the initial step in model construction.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12079 entries, 0 to 12078
Data columns (total 25 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   destination           12079 non-null  category
 1   passanger             12079 non-null  category
 2   weather               12079 non-null  category
 3   temperature           12079 non-null  category
 4   time                  12079 non-null  category
 5   coupon                12079 non-null  category
 6   expiration            12079 non-null  category
 7   gender                12079 non-null  category
 8   age                   12079 non-null  category
 9   maritalStatus         12079 non-null  category
 10  has_children          12079 non-null  int64
 11  education             12079 non-null  category
 12  occupation            12079 non-null  category
 13  income                12079 non-null  category
 14  Bar                   12079 non-null  category
 15  CoffeeHouse           12079 non-null  category
 16  CarryAway             12079 non-null  category
 17  RestaurantLessThan20  12079 non-null  category
 18  Restaurant20To50      12079 non-null  category
 19  toCoupon_GEQ5min      12079 non-null  int64
 20  toCoupon_GEQ15min     12079 non-null  int64
 21  toCoupon_GEQ25min     12079 non-null  int64
 22  direction_same        12079 non-null  int64
 23  direction_opp         12079 non-null  int64
 24  Y                     12079 non-null  int64
dtypes: category(18), int64(7)
memory usage: 877.2 KB
```

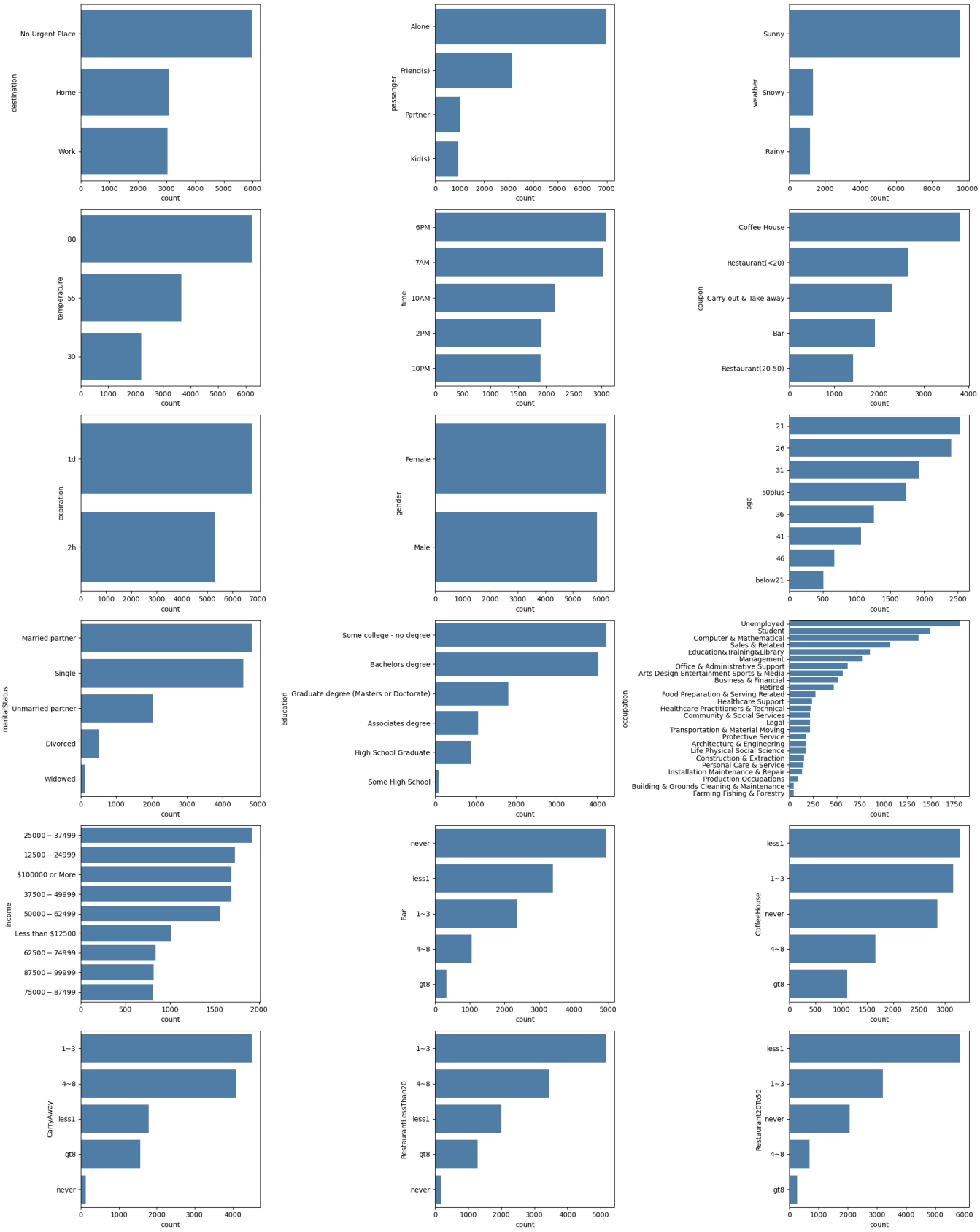
Finally, as a last step, it was decided to drop the column "toCoupon_GEQ5min" since it contains only one unique variable, rendering it useless for encoding categorical variables. After this cleaning and preprocessing phase, the dataset was reduced to 12,079 instances with 24 attributes.

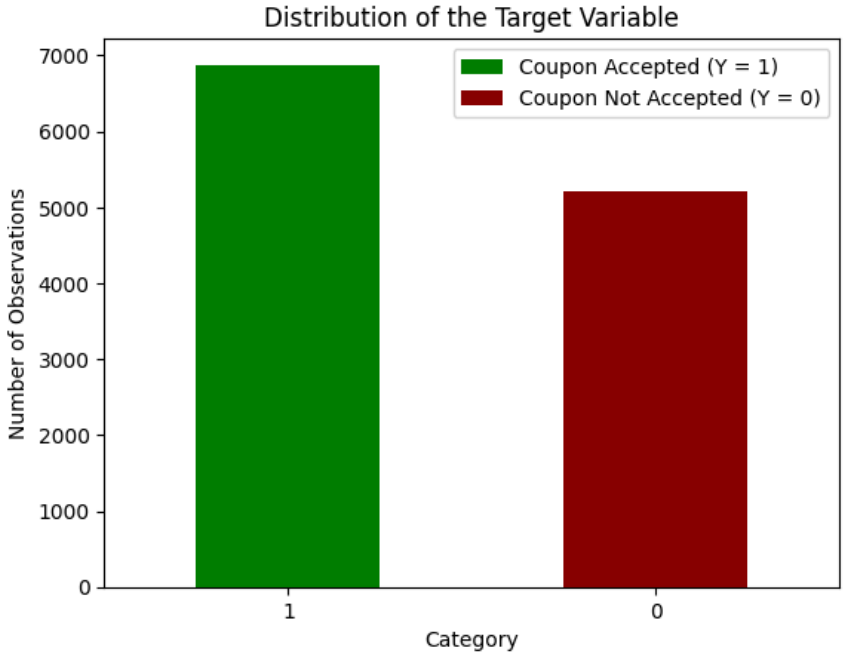
Out[10]:
```
has_children         2
toCoupon_GEQ5min     1
toCoupon_GEQ15min    2
toCoupon_GEQ25min    2
direction_same       2
direction_opp        2
Y                    2
dtype: int64
```

```
Number of instances post data cleaning and preprocessing = 12079
Number of attributes post data cleaning and preprocessing = 24
```

Data Visualization

Subsequently, various visualizations were conducted to understand the distribution of categorical variables and the balance of the target variable. Bar charts provided an overview of the distribution of each categorical variable, while the analysis of the target variable revealed an imbalance with 6,877 instances of coupon acceptance (Y=1) and 5,202 instances of non-acceptance (Y=0).

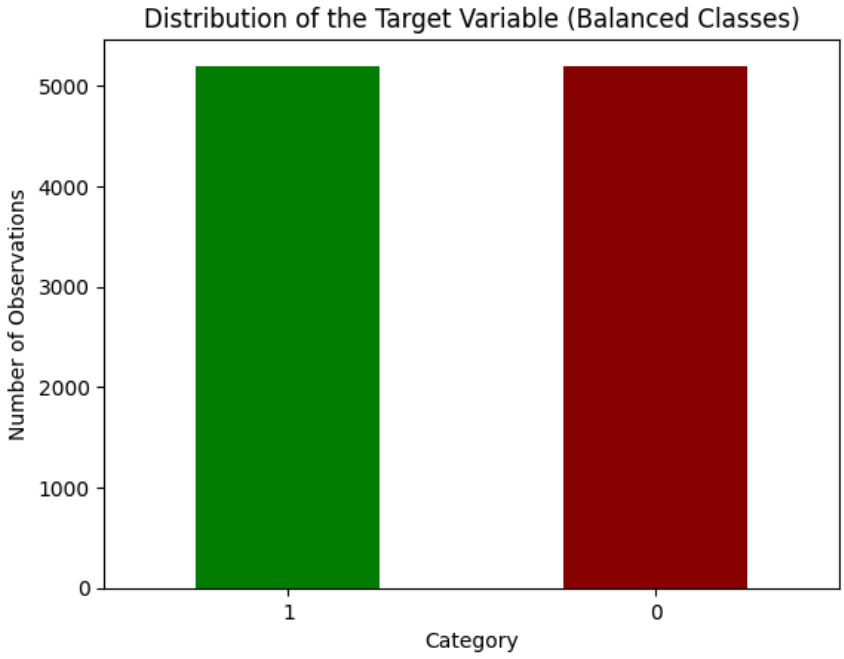





```
Number of observations for Coupon Accepted (Y = 1): 6877
Number of observations for Coupon Not Accepted (Y = 0): 5202
```

Balancing Classes Distribution to Achieve Equal Representation

To address this class imbalance, the majority class (Y=1) was undersampled to match the minority class (Y=0) using the sampling method from the sklearn library. After this operation, the dataset was balanced with 10,404 instances, fairly representing both classes. This balanced dataset was then used for model training and evaluation.




```
Number of instances after balancing classes = 10404
Number of attributes after balancing classes = 24
```

Cross-Tabulation Analysis

To gain an initial understanding of how various features of the dataset relate to the target variable (coupon acceptance), we employed the cross-tabulation technique. Specifically, the analysis focused on the interactions between user demographic and contextual variables and their likelihood of accepting a coupon. This involved segmenting the dataset based on column contexts to facilitate clearer interpretation. However, the results are not entirely clear, as there are no discernible columns that distinctly separate the classes.

Out[17]:

			Y	0	1
gender	**age**	**maritalStatus**	**has_children**		
Female	**21**	**Married partner**	**0**	61	55
			1	67	104
		Single	**0**	177	241
		Unmarried partner	**0**	81	137
			1	12	10
...	**...**	**...**	**...**	...	...
Male	**50plus**	**Unmarried partner**	**0**	30	12
			1	52	35
		Widowed	**0**	28	16
	below21	**Single**	**0**	59	138
		Unmarried partner	**0**	15	29

81 rows × 2 columns

Out[18]:

			Y	0	1
education	**occupation**	**income**			
Associates degree	**Arts Design Entertainment Sports & Media**	$12500-$**24999**		5	17
		$25000-$**37499**		14	8
		$75000-$**87499**		14	8
	Building & Grounds Cleaning & Maintenance	$37500-$**49999**		9	13
	Community & Social Services	$50000-$**62499**		7	15
...	**...**	**...**		...	...
Some college - no degree	**Unemployed**	$50000-$**62499**		25	41
		$62500-$**74999**		10	12
		$75000-$**87499**		11	11
		$87500-$**99999**		17	70
		Less than $12500		21	23

279 rows × 2 columns

Out[19]:

					Y	0	1
Bar	**CoffeeHouse**	**CarryAway**	**RestaurantLessThan20**	**Restaurant20To50**			
1~3	**1~3**	**1~3**	**1~3**	**1~3**		21	44
				less1		5	17
			4~8	**1~3**		9	13
				4~8		7	15
				less1		16	27
...	**...**	**...**	**...**	**...**		...	...
never	**never**	**less1**	**less1**	**never**		40	48
			never	**never**		31	13
		never	**1~3**	**never**		14	7
			gt8	**less1**		9	13
			less1	**less1**		3	19

290 rows × 2 columns

Out[20]:

			Y	0	1
toCoupon_GEQ15min	**toCoupon_GEQ25min**	**direction_same**	**direction_opp**		
0	**0**	**0**	**1**	1094	1704
		1	**0**	570	968
1	**0**	**0**	**1**	1790	2409
		1	**0**	288	295
	1	**0**	**1**	745	541

Out[21]:

					Y	0	1
destination	**passanger**	**weather**	**temperature**	**time**			
Home	**Alone**	**Rainy**	**55**	**10PM**	10	18
				6PM	52	32
		Snowy	**30**	**10PM**	137	39
				6PM	58	38
		Sunny	**30**	**6PM**	29	95
...	**...**	**...**	**...**	**...**	...	...
Work	**Alone**	**Rainy**	**55**	**7AM**	199	97
		Snowy	**30**	**7AM**	154	127
		Sunny	**30**	**7AM**	86	44
			55	**7AM**	450	379
			80	**7AM**	425	697

76 rows × 2 columns

Out[22]:

		Y	0	1
coupon	**expiration**			
Bar	**1d**	726	500
	2h	202	112
Carry out & Take away	**1d**	249	935
	2h	255	454
Coffee House	**1d**	688	929
	2h	1096	877
Restaurant(20-50)	**1d**	368	389
	2h	252	133
Restaurant(<20)	**1d**	167	940
	2h	484	648

Machine Learning Models Implementation

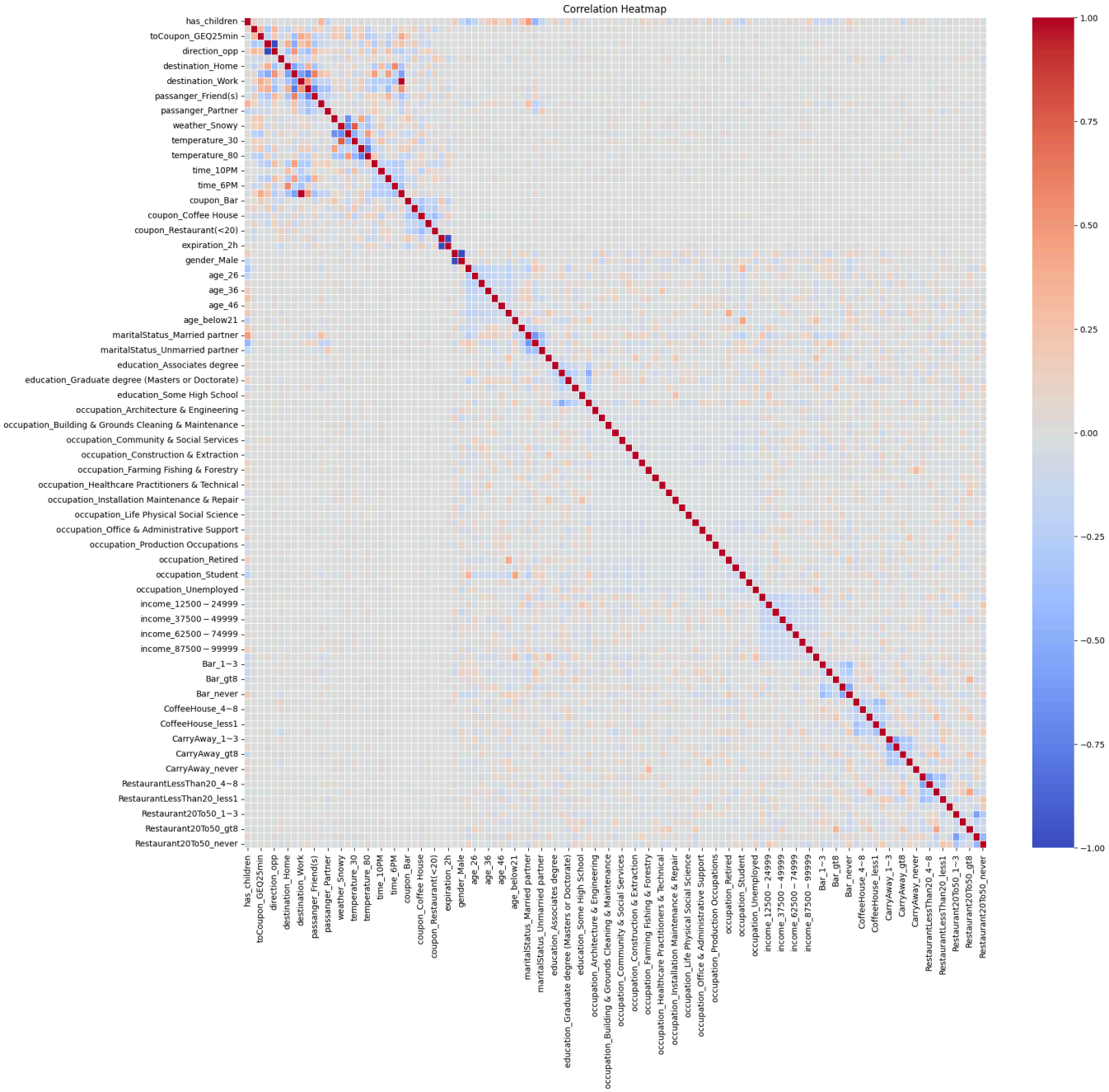
To tackle the main challenge of predicting coupon acceptance, an array of machine learning models was employed. The objective is to compare their performance both in terms of effectiveness (correct classification) and efficiency (time taken for training and prediction), with the ultimate goal to understand which one could be better suited for the task, given the available data and scenario. The models utilized include:

- Linear Classifiers: Logistic Regression, Support Vector Machine (SVM)
- Decision Tree
- K-Nearest Neighbors (KNN)
- Non-Linear SVM
- Naive Bayes
- Ensemble Methods: Random Forest
- Neural Networks

Before starting, to manage categorical variables, the One-hot encoding technique was applied. This method involves converting each category of a variable into a binary array where a "1" is placed in the position corresponding to the category and "0" in all other positions. One-hot encoding is chosen because machine learning algorithms perform better with numerical inputs. Additionally, it avoids erroneously assigning an order or scale to categories, which may not be appropriate for some categorical variables (as in this context, where several categorical variables are not ordinal).

This approach also allowed the generation of a correlation matrix to visualize the relationships between variables. Overall, the correlation matrix revealed low correlations between the features.

```
<class 'pandas.core.frame.DataFrame'>
Index: 10404 entries, 0 to 10403
Columns: 111 entries, has_children to Restaurant20To50_never
dtypes: bool(105), int64(6)
memory usage: 1.6 MB
```



Splitting Input Data into Training, Validation, and Test Sets

The balanced dataset was divided into training, validation, and test sets to ensure robustness and accurate parameter and metric estimation. The split was allocated as follows: 60% for training, 20% for validation, and 20% for testing. Solely utilized for model evaluation, the test set remained untouched during training. Meanwhile, the validation set played a critical role in parameter tuning, preventing overfitting on the training data and providing a refined estimate of model effectiveness. Subsequently, employing cross-validation techniques will further strengthen model evaluation.

```
Size of original set: 10404
Size of training set: 6242    Ratio: 60%
Size of validation set: 2081  Ratio: 20%
Size of testing set: 2081     Ratio: 20%
```

Defining Functions for Model Metrics Evaluation

Functions were defined to evaluate the performance of the models using various metrics such as accuracy, precision, recall, F1 score, and confusion matrix. These functions aimed to expedite the code writing process and ensure consistency in evaluation methodology across different models. To calculate these metrics, a 10-fold cross-validation approach was employed. This technique involves splitting the dataset into ten subsets, training the model on nine subsets, and evaluating its performance on the remaining subset. By repeating this process ten times and averaging the results, more robust estimates of model performance can be obtained.

In evaluating the models, several key metrics were considered:

- **Accuracy:** This metric measures the proportion of correctly classified instances among all instances. It provides an overall assessment of a model's correctness in predicting both positive and negative instances.
- **Precision:** Precision quantifies the proportion of true positive instances among instances predicted as positive. It indicates how many of the instances predicted as positive are actually positive, minimizing false positives.
- **Recall:** Recall, also known as sensitivity, measures the proportion of true positive instances that were correctly predicted. It indicates how many of the actual positive instances were correctly identified by the model, minimizing false negatives.
- **F1 Score:** The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, offering a single metric to assess a model's accuracy in classification tasks.
- **Confusion Matrix:** A confusion matrix is a table that displays counts of the true positive, true negative, false positive, and false negative predictions. It provides detailed insight into the performance of a classification model by visualizing the distribution of predictions across different classes.

Furthermore, to assess the discriminative power of the models, ROC curves were plotted, and the area under the ROC curve (AUC) was calculated. ROC curves visualize the trade-off between the true positive rate and false positive rate across various threshold settings. A higher AUC value indicates better discriminative performance of the model.

In addition to evaluating model performance, computational efficiency was also considered. Therefore, a dedicated function was designed to measure the training and prediction times of each model.

These functions were then applied across all models to strive for a comprehensive understanding of each model's effectiveness in predicting coupon acceptance.

Linear Classifiers

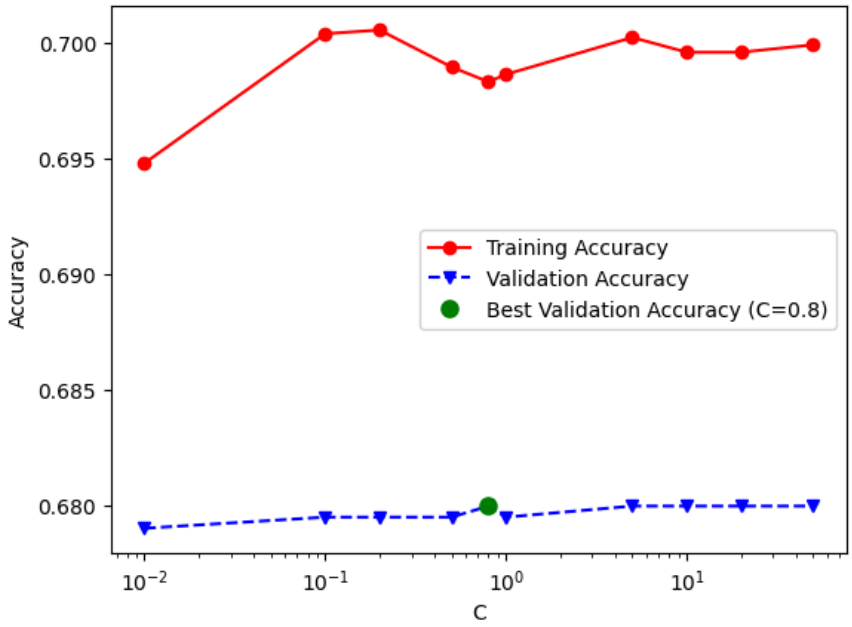
Logistic Regression

Logistic Regression is a linear classifier that predicts the probability of a binary outcome. In this project, it demonstrated moderate performance. The strength of this classifier lies in its simplicity and interpretability, making it an excellent choice as a baseline model. However, its linear nature may limit its ability to capture intricate relationships within the data, thereby explaining its moderate performance compared to more complex models.

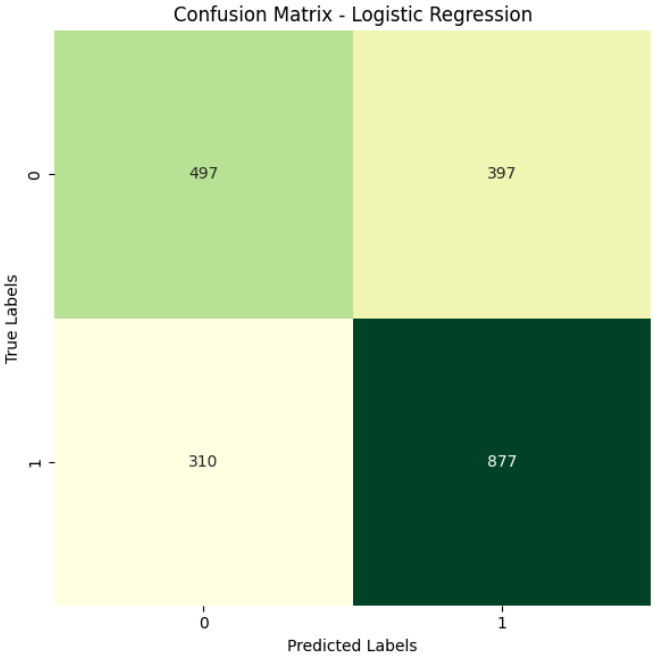
To optimize the performance of the Logistic Regression model, we engaged in parameter tuning, specifically adjusting the regularization parameter C. A range of C values was explored to find the optimal balance between model complexity and generalization. During this process, the model was trained using each C value and evaluated on both the training and validation sets. The validation set was employed to avoid including the test set in the parameter tuning process, ensuring a more unbiased evaluation of the model. The resulting plot illustrates the trade-off between model complexity and accuracy, highlighting the optimal C parameter as the one achieving the highest validation accuracy without sacrificing generalization performance.

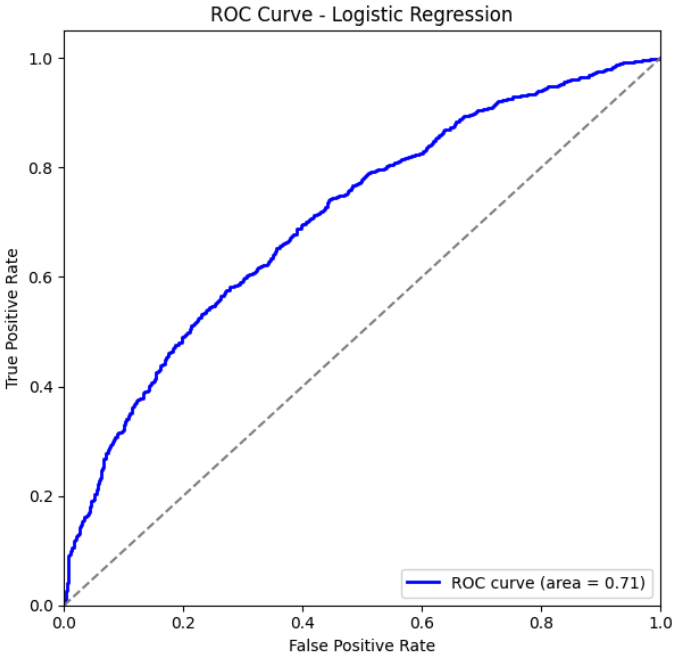
Finally, the classifier was built using the best parameter found and then evaluated on the training set applying the functions described above.

This process of parameter tuning and model evaluation was applied to all subsequent algorithms discussed in this project, thus the repetition these steps for each model will be avoided.




```
Logistic Regression:
  Accuracy: 66.03%
  Precision: 68.82%
  Recall: 73.88%
  F1 Score: 71.23%
```





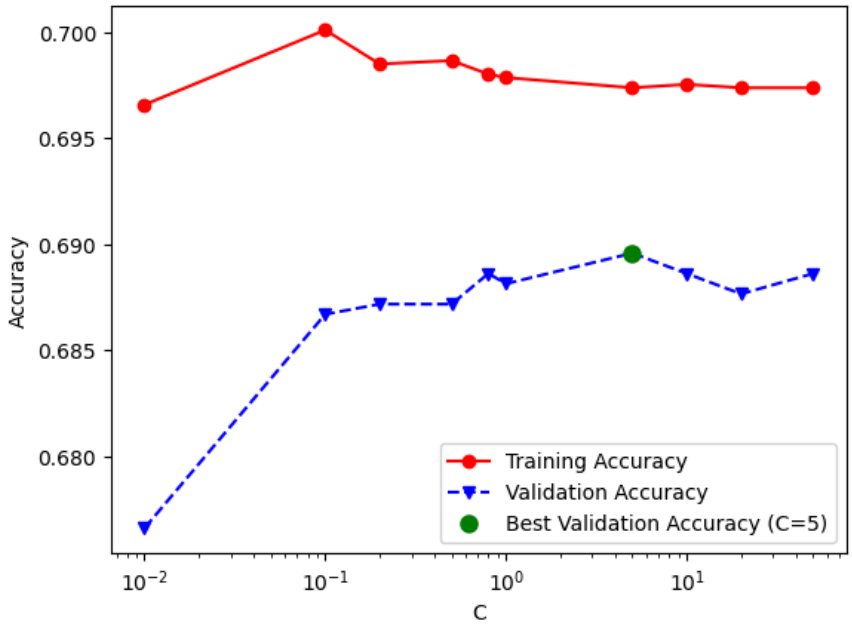

```
Training time: 0.049
Prediction time: 0.010
```

Support Vector Machine (SVM)

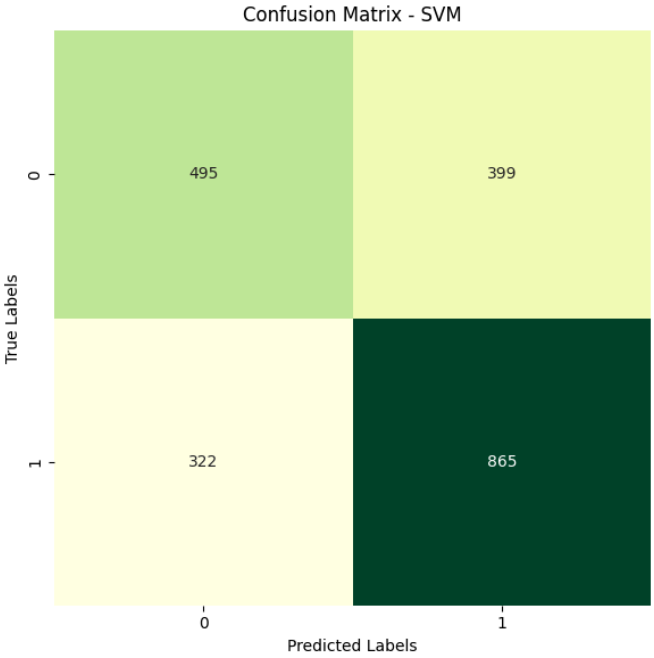
The Support Vector Machine (SVM) model, another linear classifier, aims to find the optimal hyperplane that maximizes the margin between classes. This technique is particularly effective in high-dimensional spaces and is robust to overfitting, especially when the number of dimensions exceeds the number of samples.

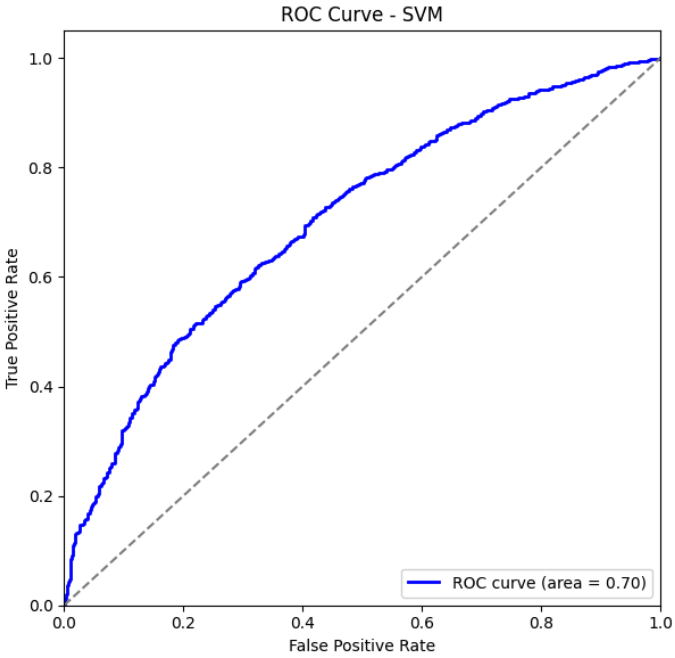
In this project, SVM demonstrated slightly lower performance compared to Logistic Regression. This may be due to the dataset's nature, where feature relationships might not be complex enough to benefit from SVM's margin maximization.

Additionally, the computational cost of SVM can be significant, especially with large datasets. This was evident even in the current dataset, where the execution time for SVM was higher compared to other algorithms.




```
SVM:
  Accuracy: 65.35%
  Precision: 68.42%
  Recall: 72.87%
  F1 Score: 70.56%
```



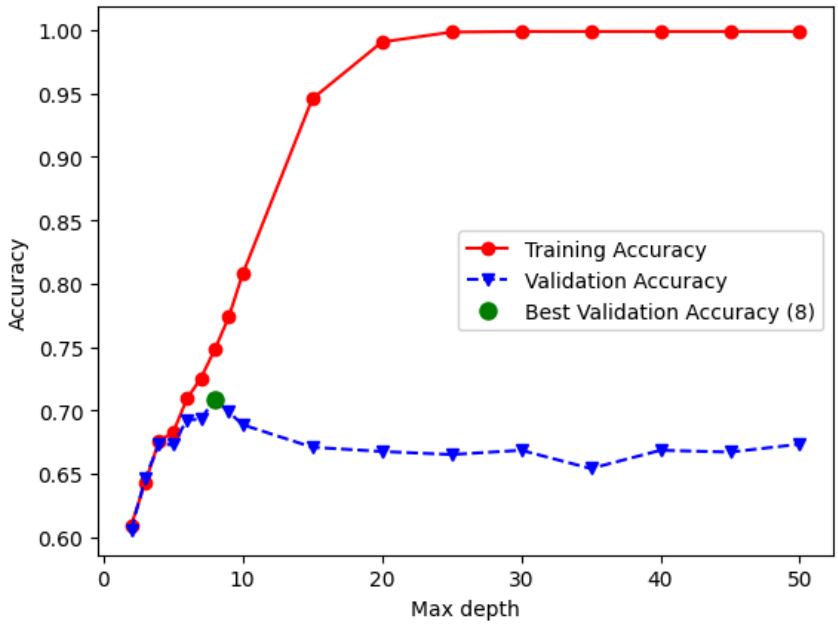



```
Training time: 4.925
Prediction time: 0.277
```

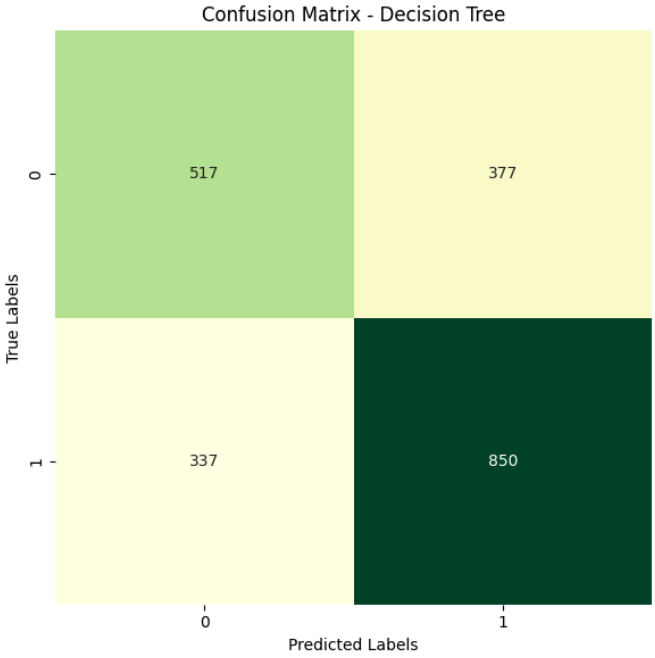
Decision Tree

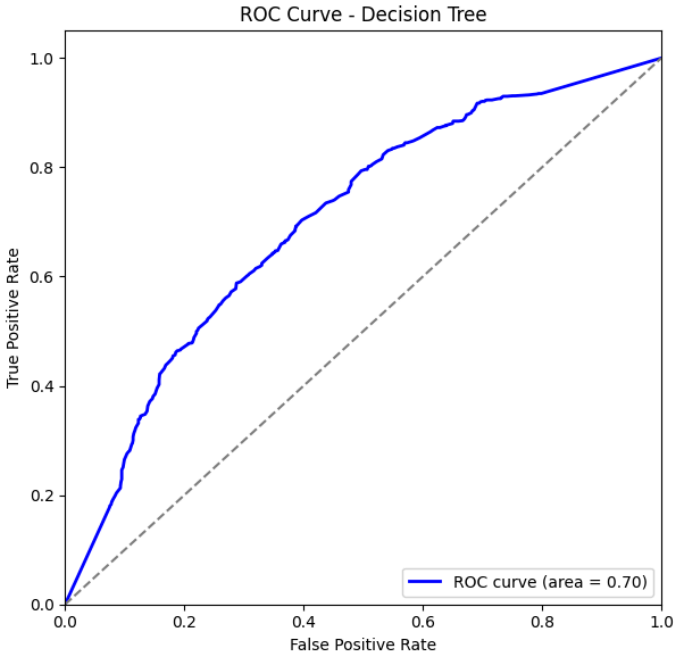
The decision tree model is a supervised learning method used for classification and regression. It operates by repeatedly dividing the dataset into smaller subsets based on the values of predictive variables, creating a tree-like structure that represents decisions. This model is valuable because it provides a visual representation of the decisions made during classification, making it easy to interpret the results. Additionally, determining the optimal maximum depth of the decision tree is crucial to avoid overfitting or underfitting the model. Striking the right balance in the tree's complexity can significantly enhance model performance.

In the context at hand, the performance of the decision tree has shown to be very similar to linear classifiers. Moreover, displaying the initial nodes of the tree already reveals a certain level of complexity, which could be a contributing factor to the model's not perfect classification.



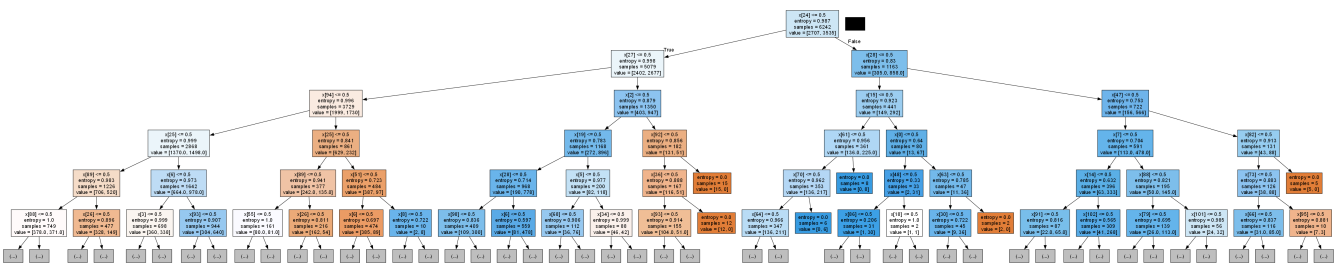

```
Decision Tree:
  Accuracy: 65.83%
  Precision: 69.29%
  Recall: 72.02%
  F1 Score: 70.53%
```






```
Training time: 0.036
Prediction time: 0.002
```

Out[35]:

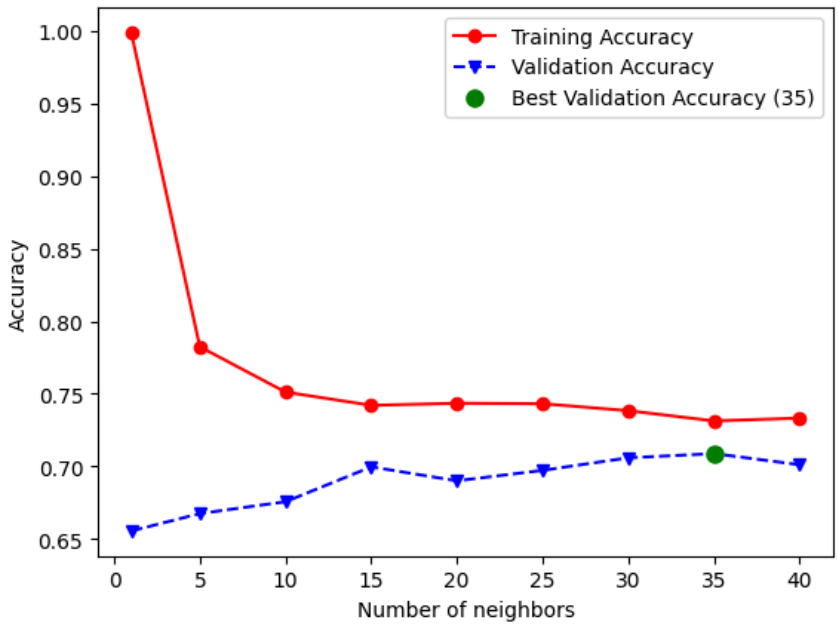


K-Nearest Neighbors

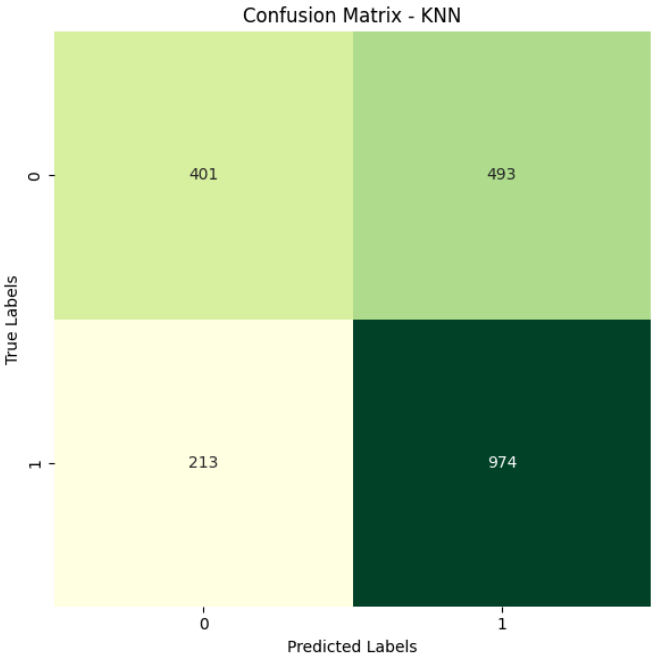
The K-Nearest Neighbors (KNN) algorithm is a simple yet effective method, which operates by finding the K nearest data points to a given query point and assigns the most common class label among them for classification tasks.

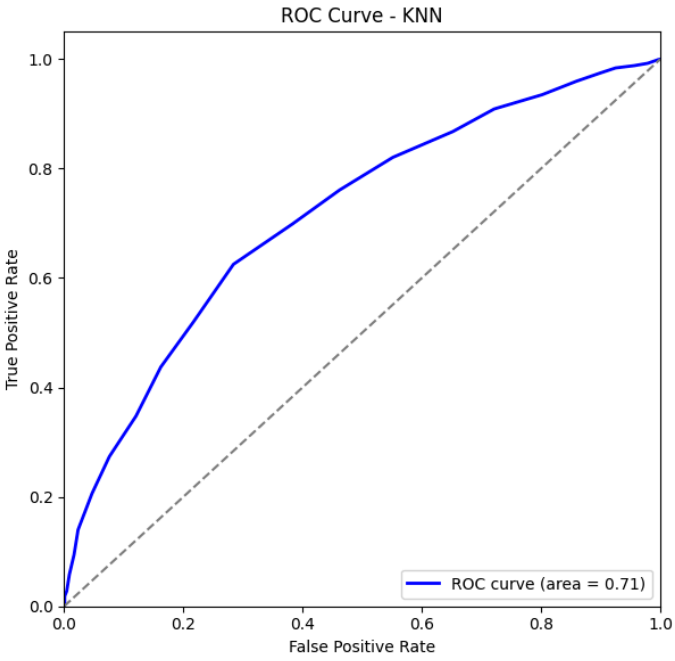
In comparison to decision trees and linear classifiers, KNN seems to exhibit slightly superior performance. Given the context of predicting coupon acceptance based on various demographic and contextual attributes, KNN's ability to capture similarities between instances in high-

dimensional space might be advantageous, since it can effectively leverage the relationships between features.




```
KNN:
  Accuracy: 66.07%
  Precision: 66.41%
  Recall: 82.06%
  F1 Score: 73.39%
```

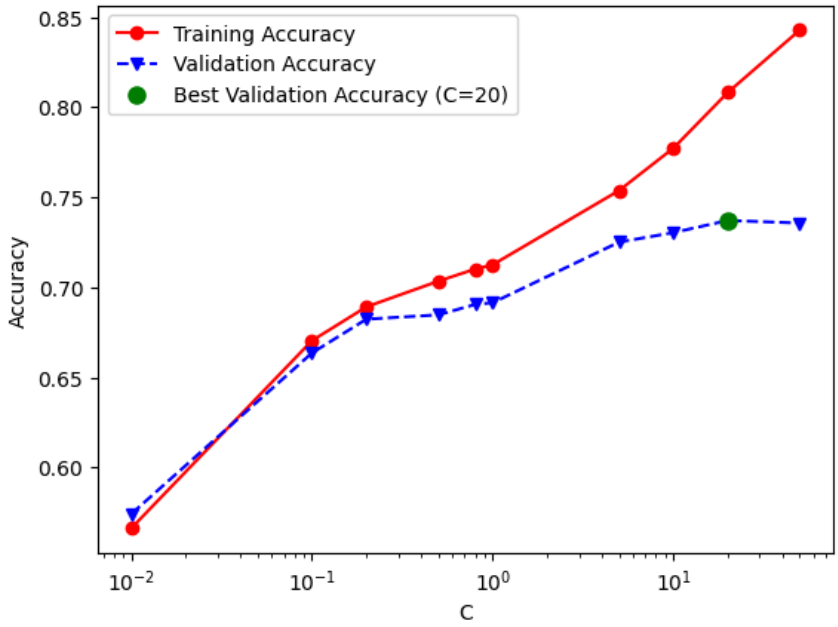




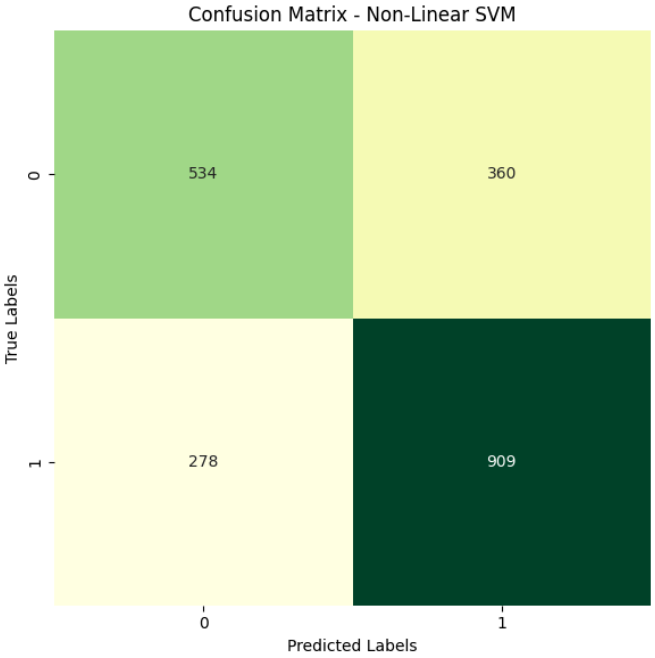

```
Training time: 0.024
Prediction time: 0.175
```

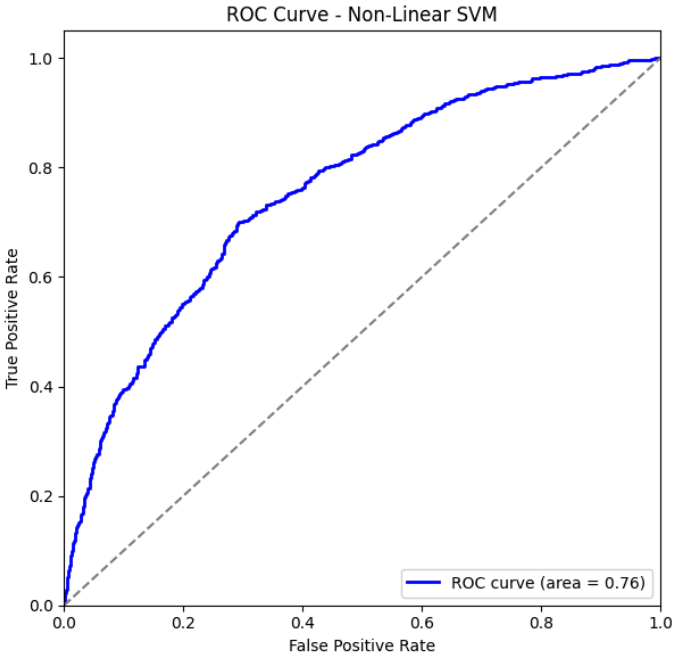
Non-Linear Support Vector Machine

Despite its significant computational time requirement, the nonlinear SVM model has exhibited superior performance compared to previous algorithms. This enhancement in performance can be attributed to SVM's capability to capture complex and nonlinear relationships in the data, particularly in contexts like the one presented, where the relationships between attributes may be intricate and not easily separable linearly.




```
Non-Linear SVM:
  Accuracy: 69.34%
  Precision: 71.67%
  Recall: 76.58%
  F1 Score: 74.00%
```



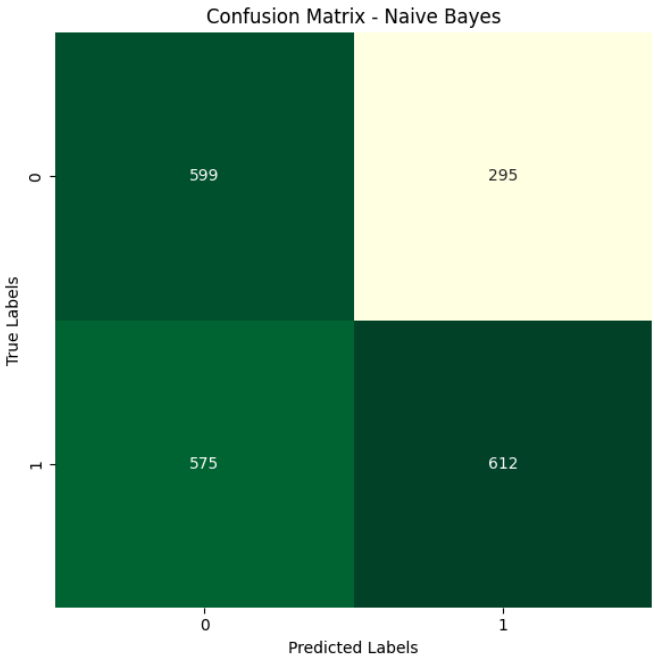


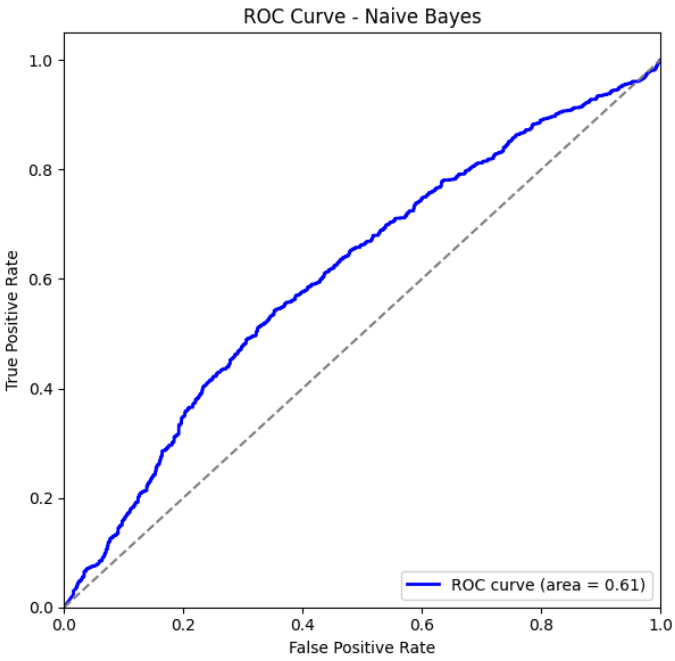

```
Training time: 2.380
Prediction time: 0.884
```

Naive Bayes

Considering the dataset's association with a paper developing a Bayesian logic-based algorithm, an attempt was made to implement a similar approach. The Naive Bayes classifier, specifically the GaussianNB function, was chosen for this purpose. However, despite its simplicity and ease of implementation, the Gaussian Naive Bayes model exhibited notably poor performance in this context. This outcome could be attributed to the assumptions underlying the Gaussian Naive Bayes model, particularly the assumption of independence among features, which may not hold true for the given dataset. Violations of these assumptions can significantly impact the model's effectiveness. Consequently, exploring alternative distributions or adapting the model to relax the independence assumption might be necessary to improve performance.

```
Naive Bayes:
  Accuracy: 58.19%
  Precision: 67.35%
  Recall: 51.57%
  F1 Score: 57.13%
```






```
Training time: 0.005
Prediction time: 0.006
```

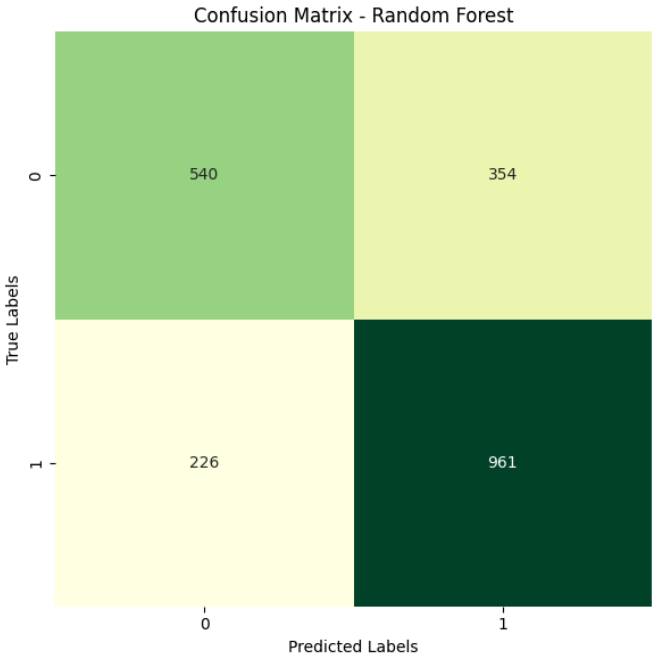
Random Forest

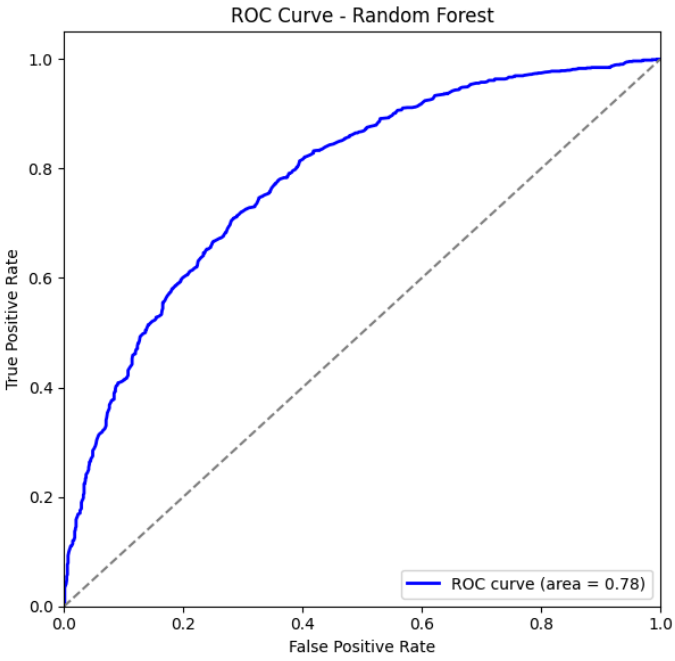
Ensemble methods, such as Random Forest, are approaches that combine multiple learning models to enhance predictive performance and overall model robustness. Random Forest is an ensemble learning technique based on decision trees, where an ensemble of independent decision trees is created, each trained on a random subset of the training data and features.

In this context, implementing a Random Forest led to improved model performance. The base classifier selected is a decision tree with a maximum depth equal to the best depth previously determined for the decision tree model. This means that each tree in the Random Forest is trained using a random subset of the training data and features, with the maximum depth of each tree matching the optimal depth found for the decision tree model.

The ensemble of decision trees in the Random Forest works together to enhance overall model accuracy while reducing the risk of overfitting compared to a single decision tree. Additionally, Random Forest automatically handles the selection of the most relevant features, making it a robust and reliable method.

```
Random Forest:
  Accuracy: 72.08%
  Precision: 73.11%
  Recall: 80.87%
  F1 Score: 76.74%
```






```
Training time: 7.201
Prediction time: 0.354
```

Neural Network

Lastly, an attempt was made to construct a neural network model using Keras. Hyperparameter tuning for the Keras model was conducted using GridSearchCV, with the optimal parameters determined to be a batch size of 30 and 50 epochs. The results of this tuning process are displayed in the image below, eliminating the need to rerun the code multiple times due to the significant execution time required.

```
...
209/209 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8010 - loss: 0.4328
Epoch 50/50
209/209 ━━━━━━━━━━━━━━━━━━━━ 0s 882us/step - accuracy: 0.8066 - loss: 0.4147
Best: 0.712913 using {'batch_size': 30, 'epochs': 50}
```

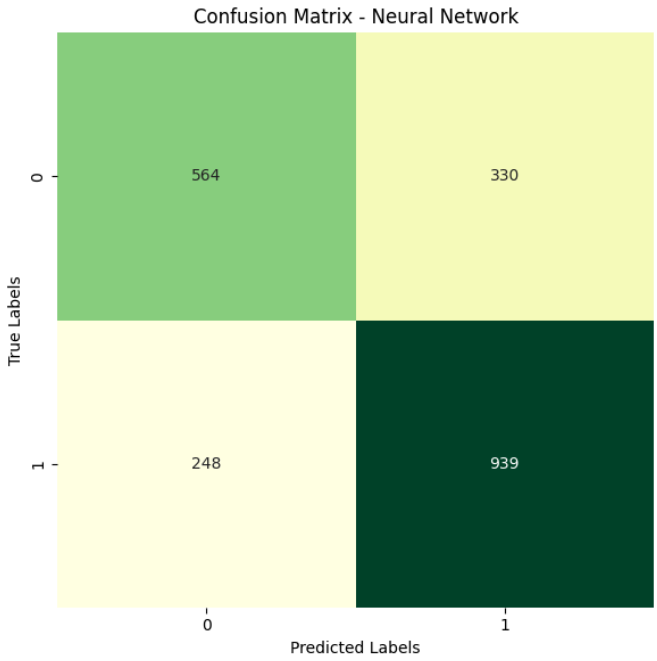
The neural network architecture consists of three densely connected layers with ReLU activation functions, followed by a final output layer with a sigmoid activation function. This architecture is commonly used for binary classification tasks, such as the one presented in the project. During model training, the Keras model was fitted to the training data for 50 epochs with a batch size of 30, while validation data was used to monitor performance and prevent overfitting.

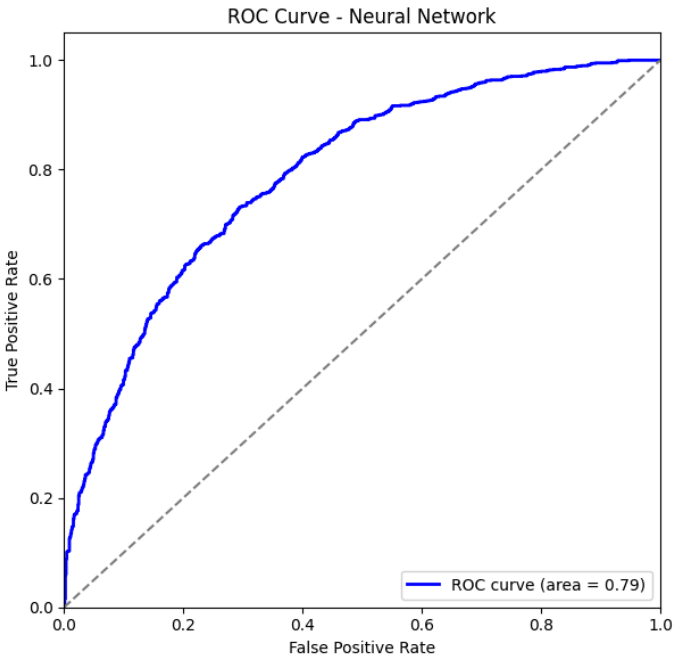
Finally, the model was used to make predictions on the test data, and the results were evaluated by comparing the predicted outcomes to the actual values. This process allowed for an assessment of the neural network's performance in predicting coupon acceptance.

```
66/66 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Training time: 26.683
Prediction time: 0.178
```

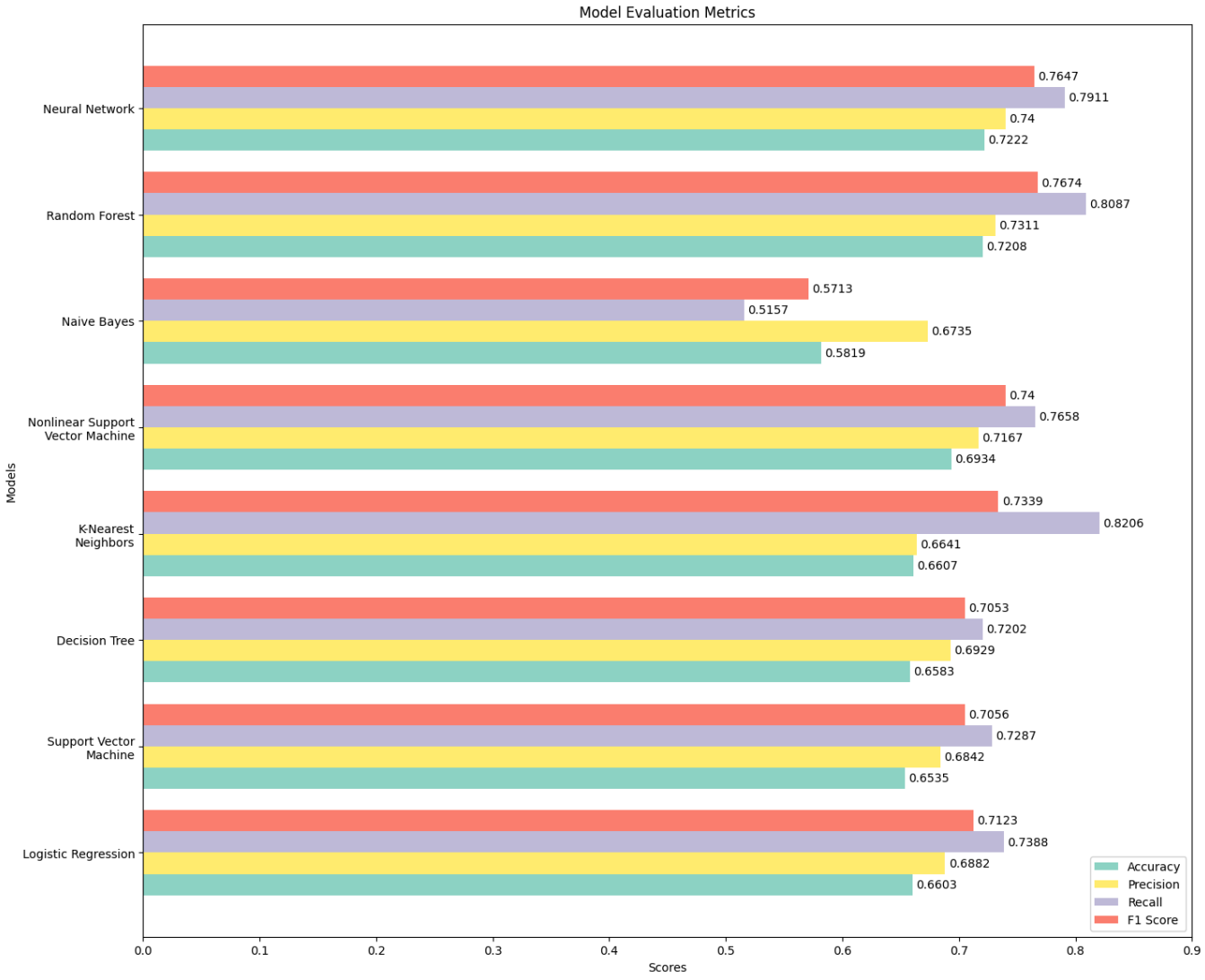
66/66 ━━━━━━━━━━━━━━━━━━━━ **0s** 642us/step
[0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 1 (expected 1)
[1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 1 (expected 1)
[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0] => 1 (expected 1)
[0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0] => 1 (expected 1)
[1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0] => 1 (expected 1)
[0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 1 (expected 1)
[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 1 (expected 1)
[1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0] => 1 (expected 1)
[0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0] => 0 (expected 1)
[1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0] => 0 (expected 0)

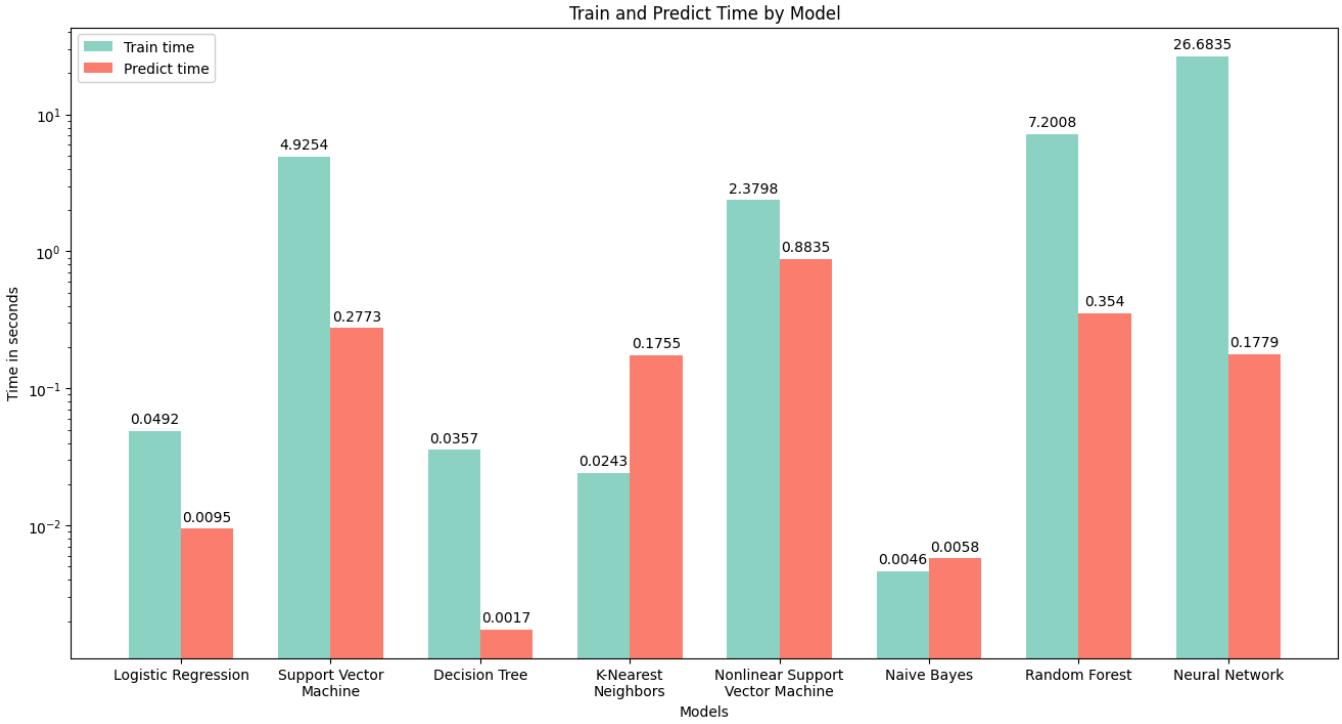
Neural Network:
  Accuracy: 72.22%
  Precision: 74.00%
  Recall: 79.11%
  F1 Score: 76.47%





Models Comparison





When comparing models based on their effectiveness (correct classification) and efficiency (time taken for training and prediction), an interesting trade-off emerges. Among them, neural networks and Random Forest stand out for their higher classification accuracy. However, they also require the longest training and testing times, as shown in the graphs. Despite their computational demands, they are more precise in correctly classifying instances. On the other hand, simpler models like Logistic Regression and Decision Trees offer faster training and prediction times but may sacrifice some accuracy. It is worth noting that across all algorithms, there is a consistent trend of better prediction for class 1 over class 0. This bias is evident when examining various confusion matrices and precision-recall values. Therefore, the choice between these models ultimately depends on finding a balance between classification effectiveness and computational efficiency, tailored to the specific needs and constraints of the application context.

Conclusions

In conclusion, the project aimed to address the challenge of predicting coupon acceptance during car trips using a variety of machine learning models. Different algorithms were applied, ranging from mathematical models like SVM to statistical models like Naive Bayes, along with rule-based models such as Random Forest, and one deep learning model.

The initial steps of data exploration, cleaning, and preprocessing were crucial to ensure dataset quality, while addressing class distribution imbalances. Through data analysis, visual exploration, class balancing, and the implementation of various algorithms, the effectiveness of different models in predicting coupon acceptance by drivers was assessed. The results indicate that more complex models such as neural networks and Random Forests tend to achieve the highest accuracy in classification, although at the cost of longer training and prediction times. Conversely, simpler models like logistic regression and decision trees offer greater computational efficiency but may sacrifice some accuracy. It is worth noting that all algorithms exhibit a tendency to predict class 1 better than class 0, highlighting a certain imbalance in the data. This could be attributed to the possibility that, despite balanced class frequencies, there might be an intrinsic complexity associated with class 0 compared to class 1.

The choice of model ultimately depends on a trade-off between classification accuracy and computational efficiency, taking into account the specific needs and constraints of the application. Additionally, logistic regression emerges as an excellent choice as a baseline model due to its simplicity and interpretability. Conversely, more complex algorithms such as neural networks and Random Forests can be preferred when maximizing prediction accuracy is paramount, even if it entails longer computational time.

Overall, the project provides an overview of the performance of different machine learning models in predicting coupon acceptance under different scenarios, offering guidance for selecting the most suitable model for specific requirements. These insights extend beyond the immediate context and can benefit various sectors employing personalized marketing strategies, enabling businesses to enhance their marketing endeavors, better engage customers, improve satisfaction levels, and ultimately drive revenue growth.

References

In-Vehicle Coupon Recommendation. (2020). UCI Machine Learning Repository. https://doi.org/10.24432/C5GS4P.