CSI 4142 - Fundamentals of Data Science

Project Phase 4: Data Mining

Due Date: April 5, 2024, 11:59pm

Group 33: Tanner Guiddings, Coralie Ostertag, and Dana Shayakhmetova

Introduction

In this assignment, our team dives deeper into exploring our dataset using the Scikit-Learn library, which is a powerful tool for data mining and machine learning. We'll be breaking it down into two main parts: understanding the data through summarization, preprocessing, and feature selection, followed by classification using supervised learning algorithms.

For the first part, we'll be digging more into our data. We'll be using scatter plots, boxplots, and histograms to get a feel for the data's attributes and characteristics. Then, we'll handle important tasks like dealing with missing values, converting categorical attributes into a usable format, scaling numeric attributes, and picking out the most relevant features. These steps are crucial to ensuring that our data is clean, consistent, and ready for analysis.

In the second part, we'll be putting our data to work. We'll choose a label for classification and run it through three popular algorithms: Decision Tree, Gradient Boosting, and Random Forest. By training and testing these models, we'll be able to see how well they perform in terms of accuracy, precision, recall, and the time it takes to build them. This will give us valuable insights into both our dataset and the effectiveness of different algorithms in tackling our classification task.

Part A: One Page Summary (Deliverable Part A)

Original Exploration:
To gain insights into our dataset, we employed various visualizations. Pairplots helped us identify potential relationships between variables, while histograms provided distributions of individual features. Additionally, we utilized bar plots to visualize categorical variables like 'Education', 'Marital_Status', and 'Generation'.

Correlation Heat Map:
A correlation heat map showcased relationships between variables. Strong correlations, close to 1 or -1, suggested variables moved together, while those close to 0 implied little or no relationship. We used this insight to guide feature selection.

Box Plots to Identify Outliers:
Outliers, which significantly deviate from the data, were identified through box plots. Handling outliers is crucial for model robustness.

Handling Missing Values:
Missing values, identified in Phase 2, were replaced with the median to maintain data integrity.

Feature Engineering:
We converted 'Household_Size' to a categorical variable and applied one-hot encoding to categorical variables like 'Education', 'Marital_Status', and 'Generation'. This technique improved model performance by transforming categorical data into a suitable format for machine learning algorithms.

Normalization:
Numerical variables underwent min-max normalization, ensuring all features were comparable and contributed equally to the modeling process.

Correlation Analysis:
We leveraged correlation analysis to select the most relevant features. Date-related variables showed insignificant correlations, leading us to eliminate them from further analysis.

Label Encoding:
To include 'Response' in our analysis, we made necessary changes using LabelEncoder, ensuring consistency in data representation.

By summarizing, processing, and selecting features, we enhanced our dataset's suitability for modeling and gained valuable insights to drive informed decision-making. Our approach ensures robustness and accuracy in subsequent modeling tasks.

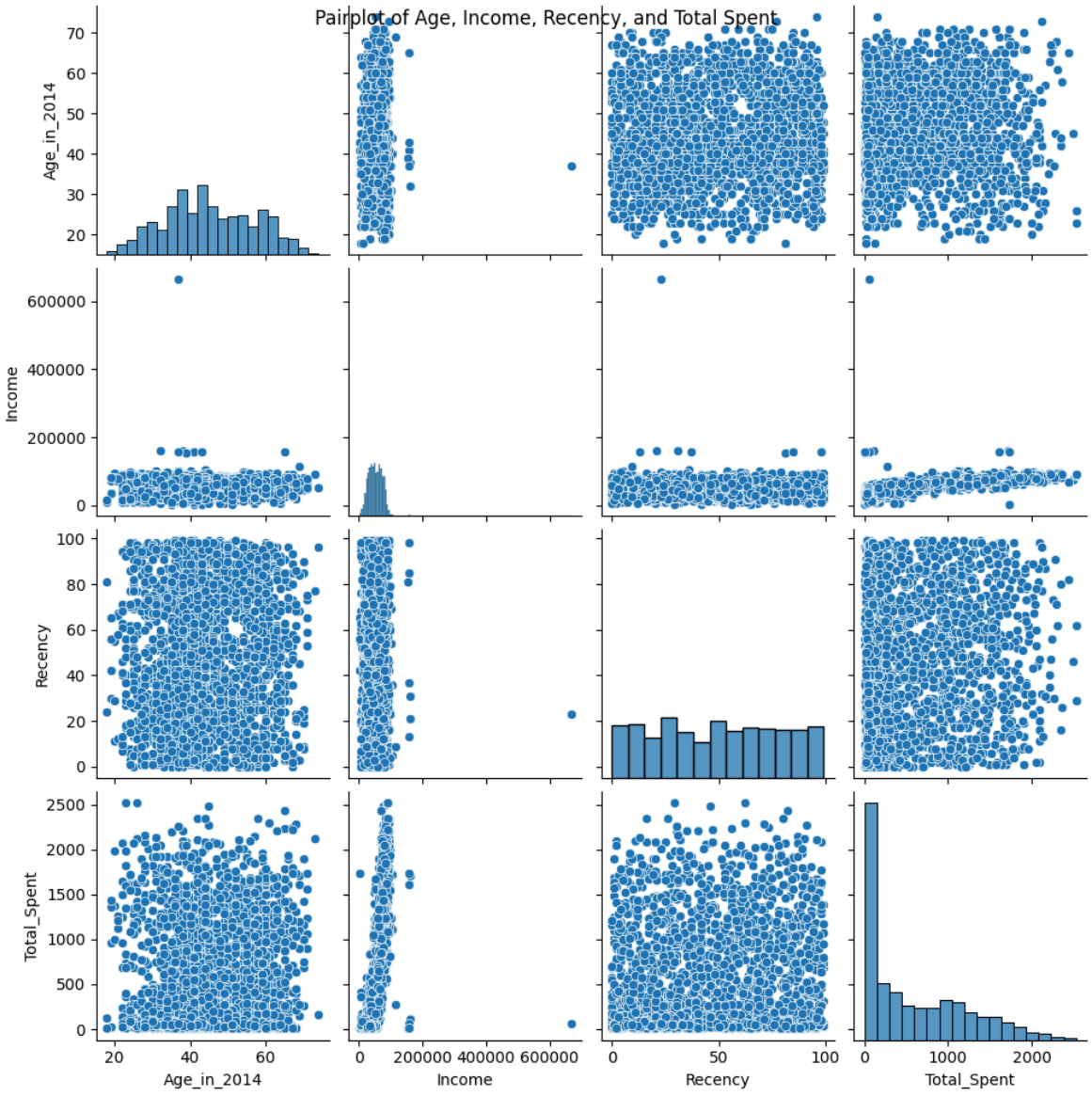
Part A: Data Summarization, Data Processing and Feature Selection (In Details)

I. Data Summarization

Original exploration:

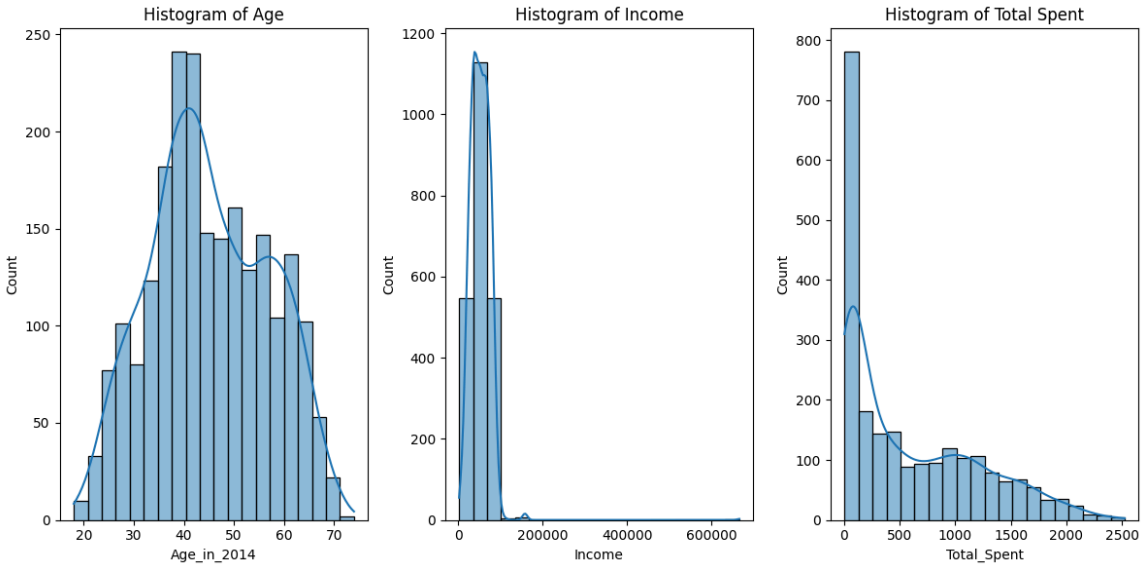
Pairplots

We did some pairplots like the following one, just to see if we could find anything interesting…



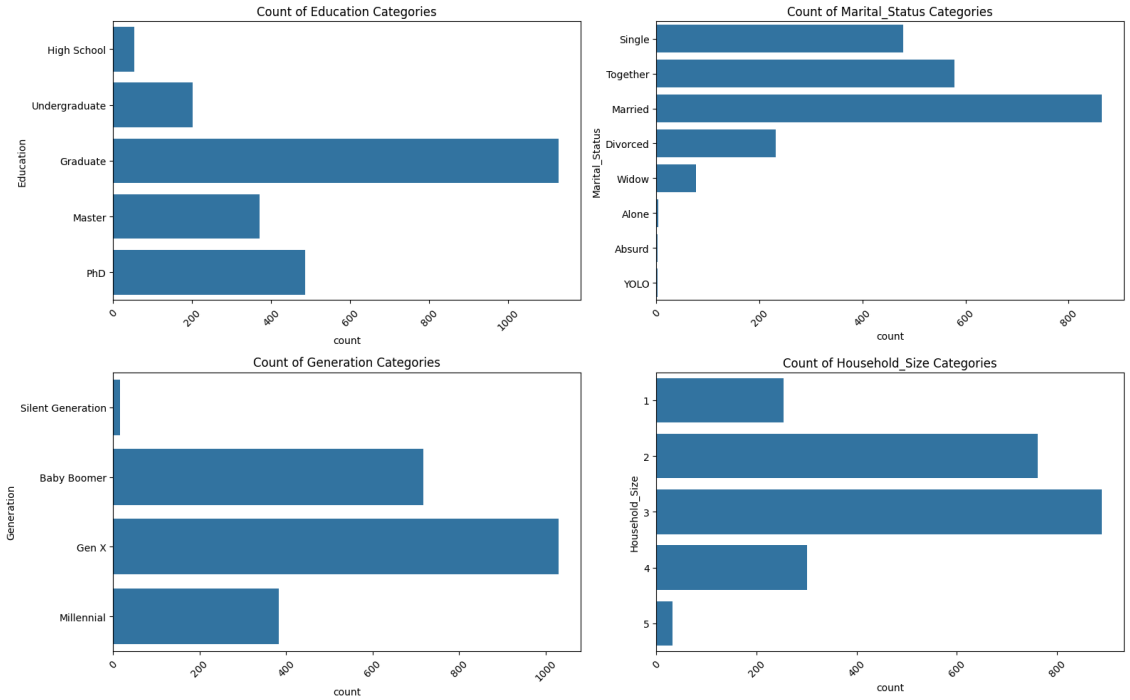
Histograms

Then we did some histograms to explore some more:



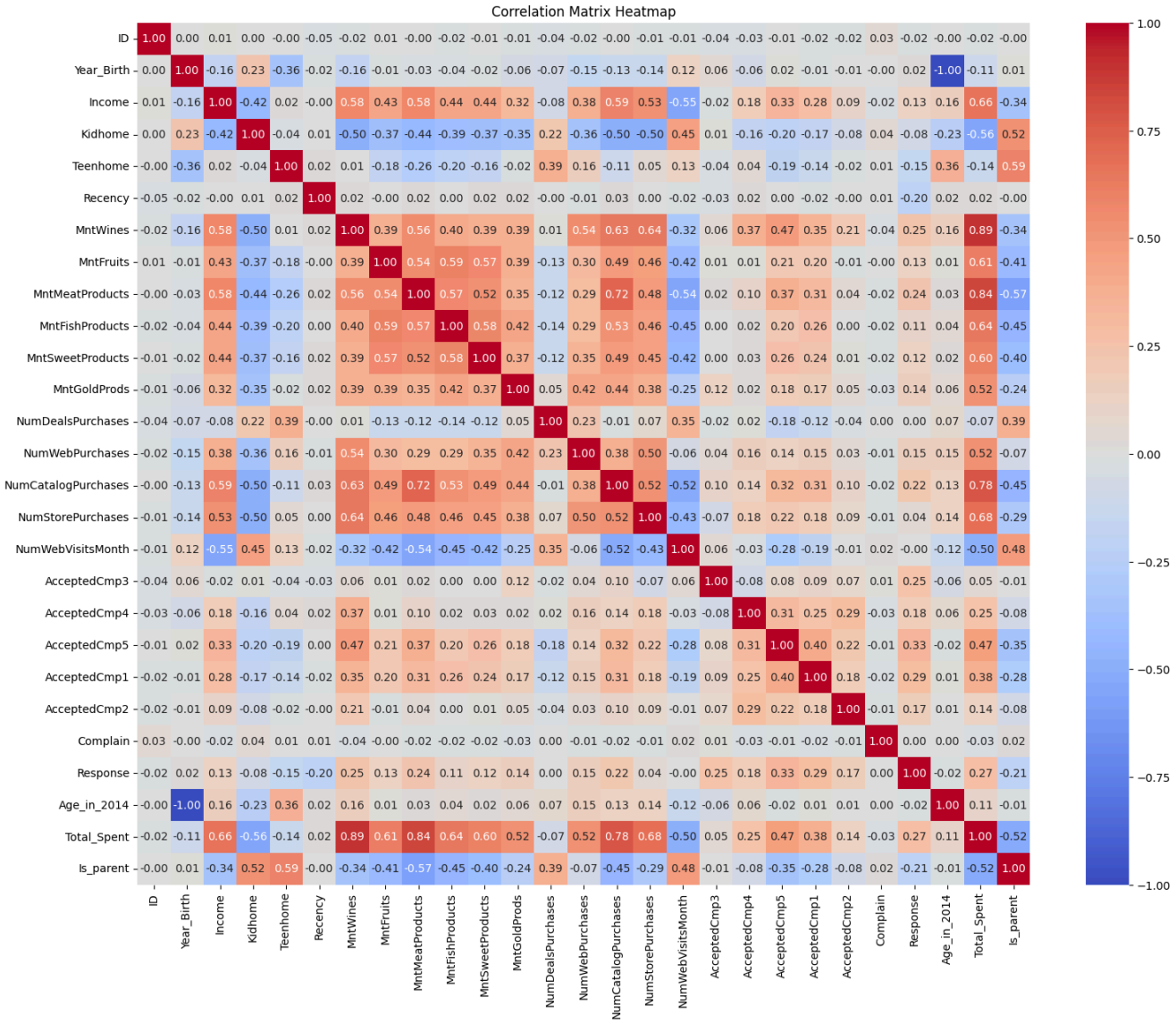
Bar Plots

We then preprocessed categorical variables in the dataset for visualization and analysis. The 'Household_Size' column is converted to categorical data type, and specific orders are defined for categorical variables such as 'Education', 'Marital_Status', and 'Generation'. Then, bar plots are generated for each categorical variable, showcasing the count of occurrences for each category. This preprocessing step helps with the interpretability of categorical variables and facilitates meaningful visual exploration of the dataset.



Correlation Heat Map

Here is the correlation map for our original data:



In a correlation matrix heatmap, the values represent the correlation coefficients between pairs of variables. The correlation coefficient is a statistical measure that describes the strength and direction of a relationship between two variables. It ranges from -1 to 1:

- **1**: Perfect positive correlation. When one variable increases, the other variable also increases proportionally. We see the red diagonal, showing the perfect correlation of 1, is present when the variable is paired to itself (which makes sense as it increases proportionally to itself).
- **0**: No correlation. There is no relationship between the variables.
- **-1**: Perfect negative correlation. When one variable increases, the other variable decreases proportionally.

The color intensity in the heatmap indicates the strength of the correlation, and helps us analyse the visualization with a bit more ease.

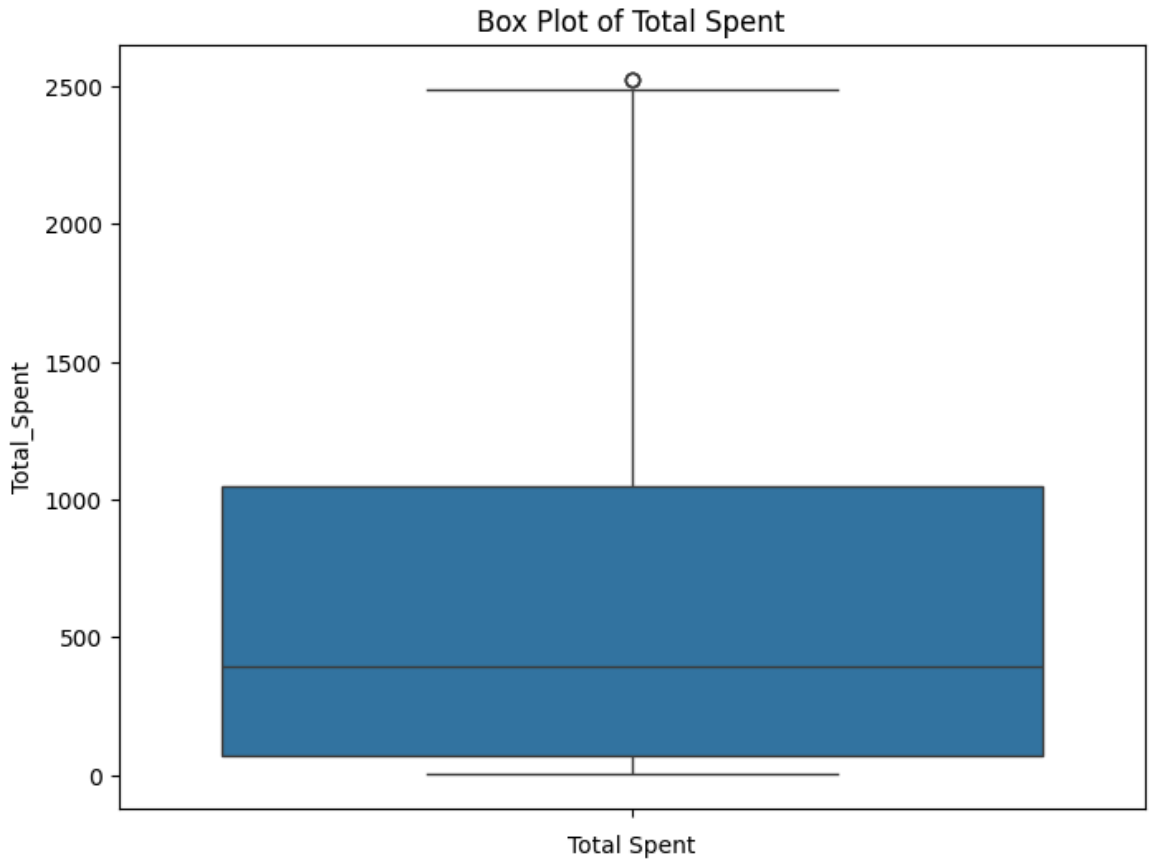
Interpreting the heatmap involves looking for patterns of correlation among variables. Strong correlations (close to 1 or -1) may indicate that the variables are related and move together in some way, while correlations close to 0 suggest little to no relationship between the variables. Positive correlations imply that the variables move in the same direction, while negative correlations imply that they move in opposite directions.

It's important to note that correlation does not imply causation. Even if two variables are highly correlated, it doesn't necessarily mean that changes in one variable cause changes in the other. Correlation analysis helps identify relationships between variables, but further analysis is often required to understand the underlying causes.

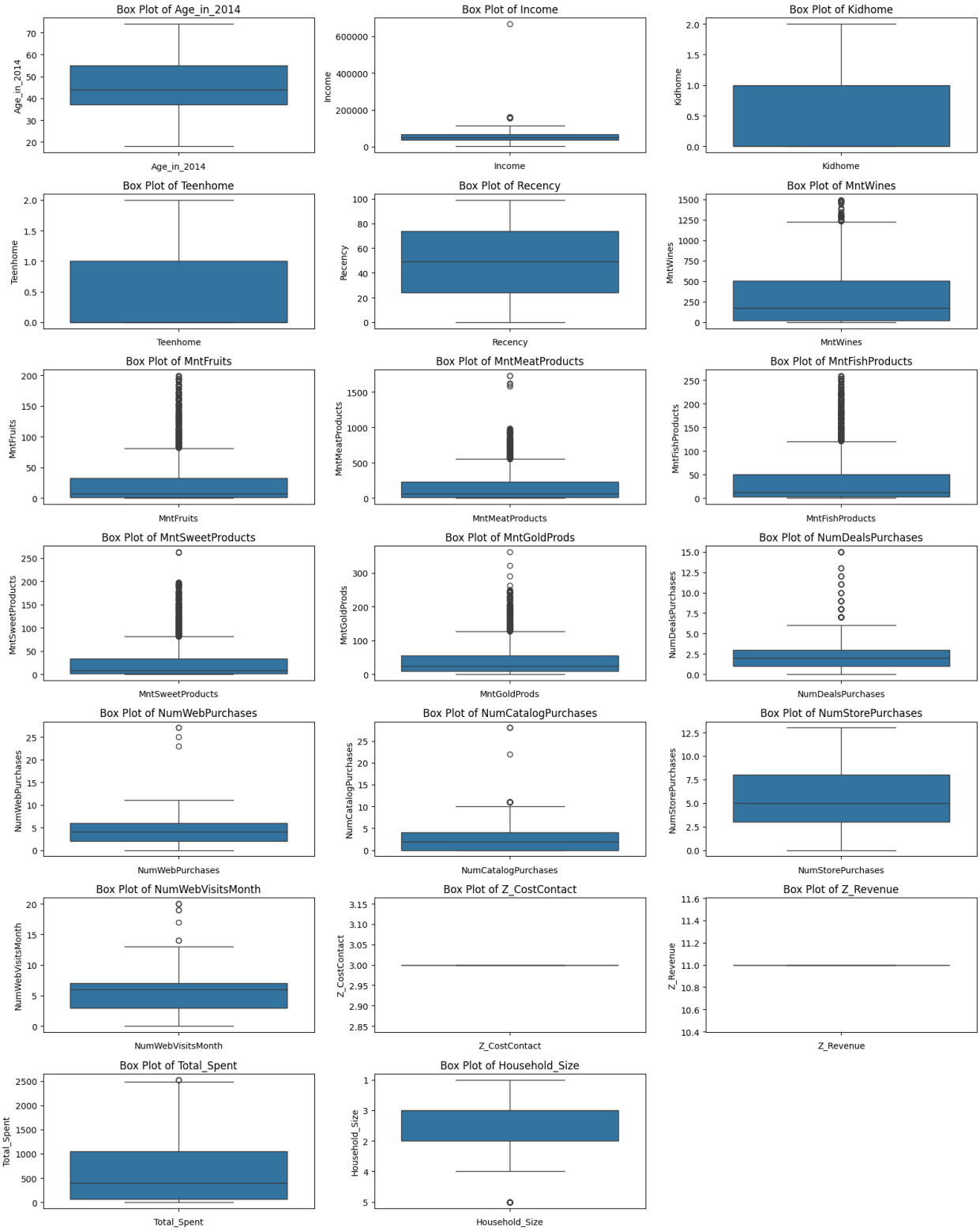
Box Plots to Identify Outliers

Outliers, or data points that significantly deviate from the rest of the data, can affect the performance of our model. Identifying and handling outliers is crucial to ensure the robustness of our analysis.

We can see one singular outlier in the Total Spent variable:



Here are all our box plots:



II. Data Processing

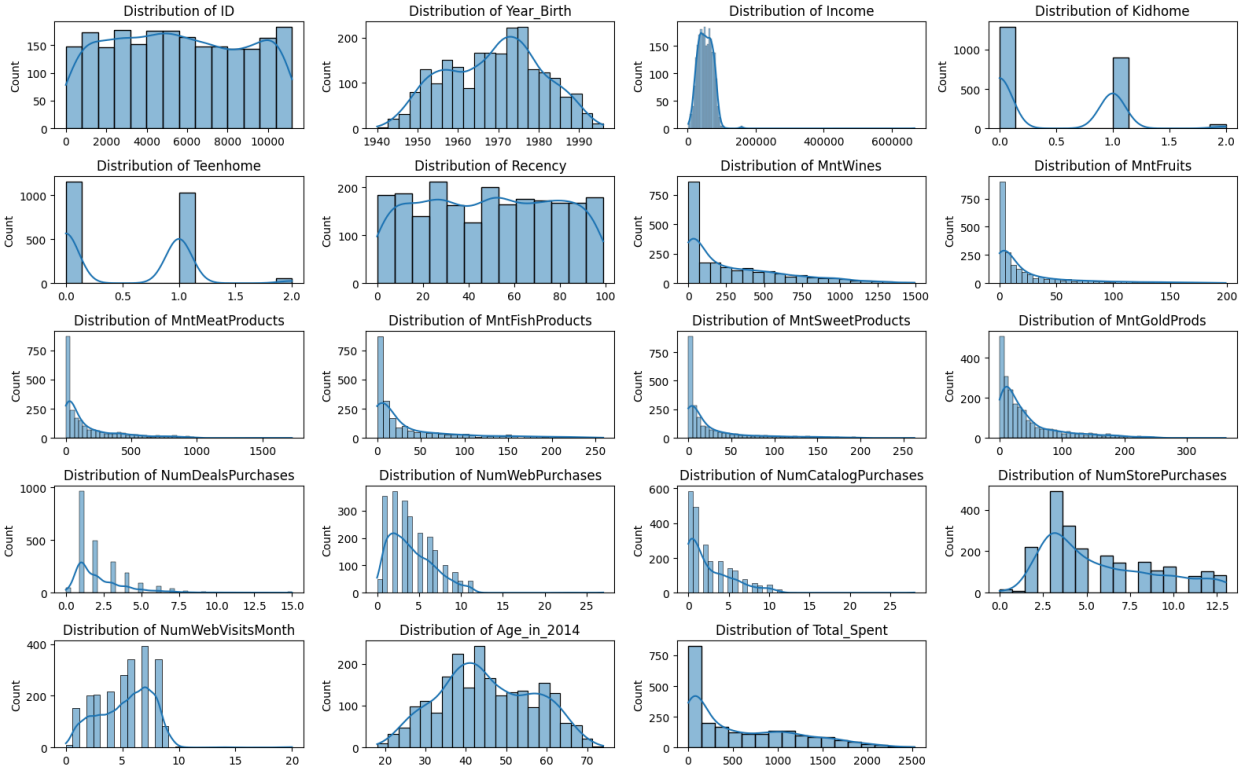
Data processing, a crucial step in data analysis and modeling, involves handling missing values, encoding categorical variables, and scaling numerical features.

During the preprocessing phase in phase 2, we addressed missing values by replacing them with the median, ensuring data integrity. Additionally, we performed feature engineering, including converting 'Household_Size' to a categorical variable. Furthermore, we applied one-hot encoding to categorical variables such as 'Education', 'Marital_Status', 'Generation', 'Household_Size', 'Living_Situation', and 'Is_parent'.

One-hot encoding is a technique that converts categorical variables into a format suitable for machine learning algorithms, enhancing model performance. It works by identifying categorical variables and creating binary columns for each unique category. By representing categories independently, it provides a numerical format that algorithms can interpret accurately, without introducing bias.

Moreover, we converted binary variables, such as ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2', 'Complain', 'Response'], to categorical using .astype('category'), ensuring consistency in data representation.

We then checked the distribution:



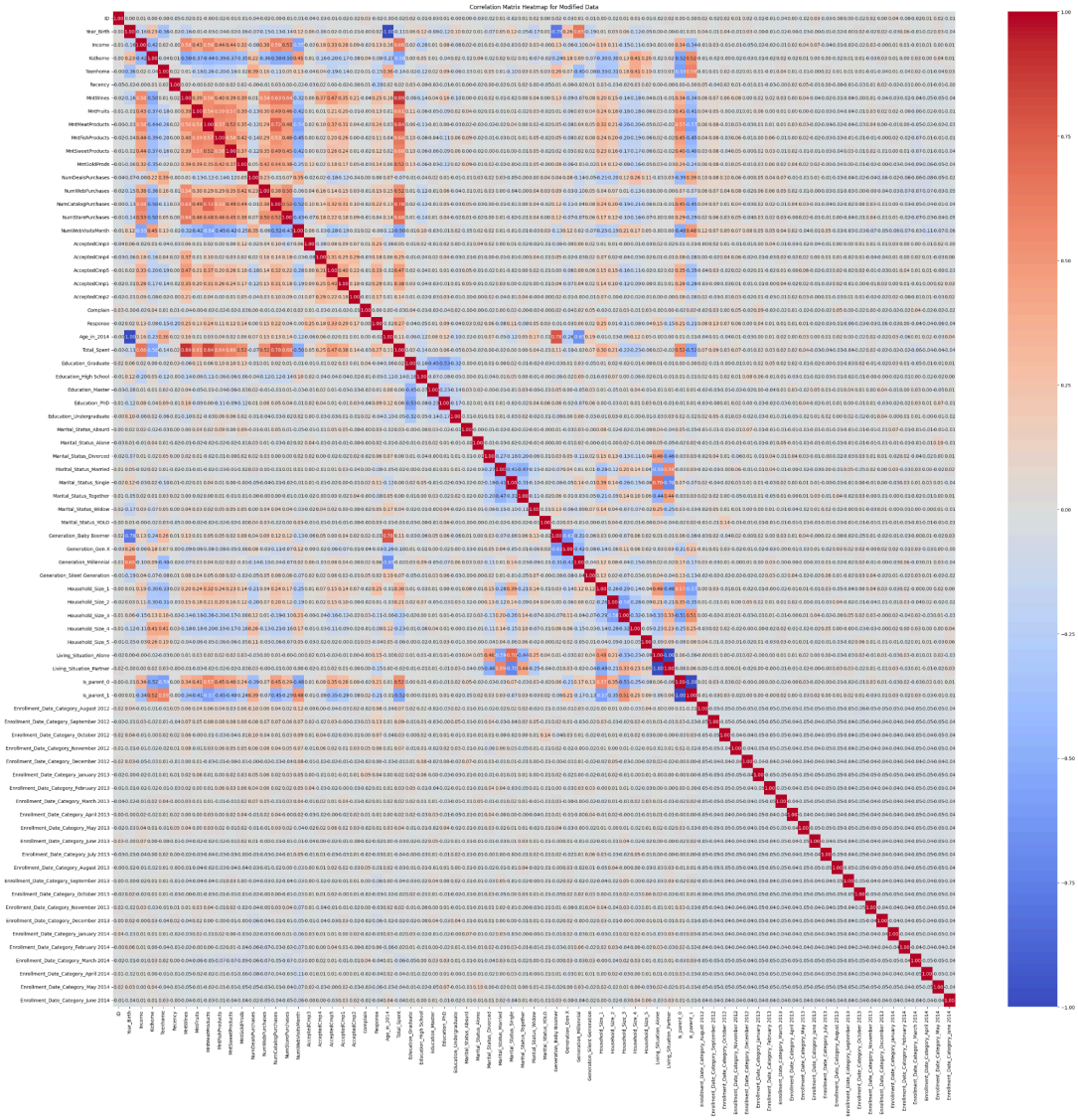
We then performed normalization on the numerical variables of our dataset. Specifically, we applied min-max normalization using the MinMaxScaler from the scikit-learn library.

Min-max normalization, also known as feature scaling, transforms numerical features to a common scale by rescaling them to a specified range, typically between 0 and 1. This technique preserves the relative relationships between data points while ensuring that all features are comparable and contribute equally to the modeling process.

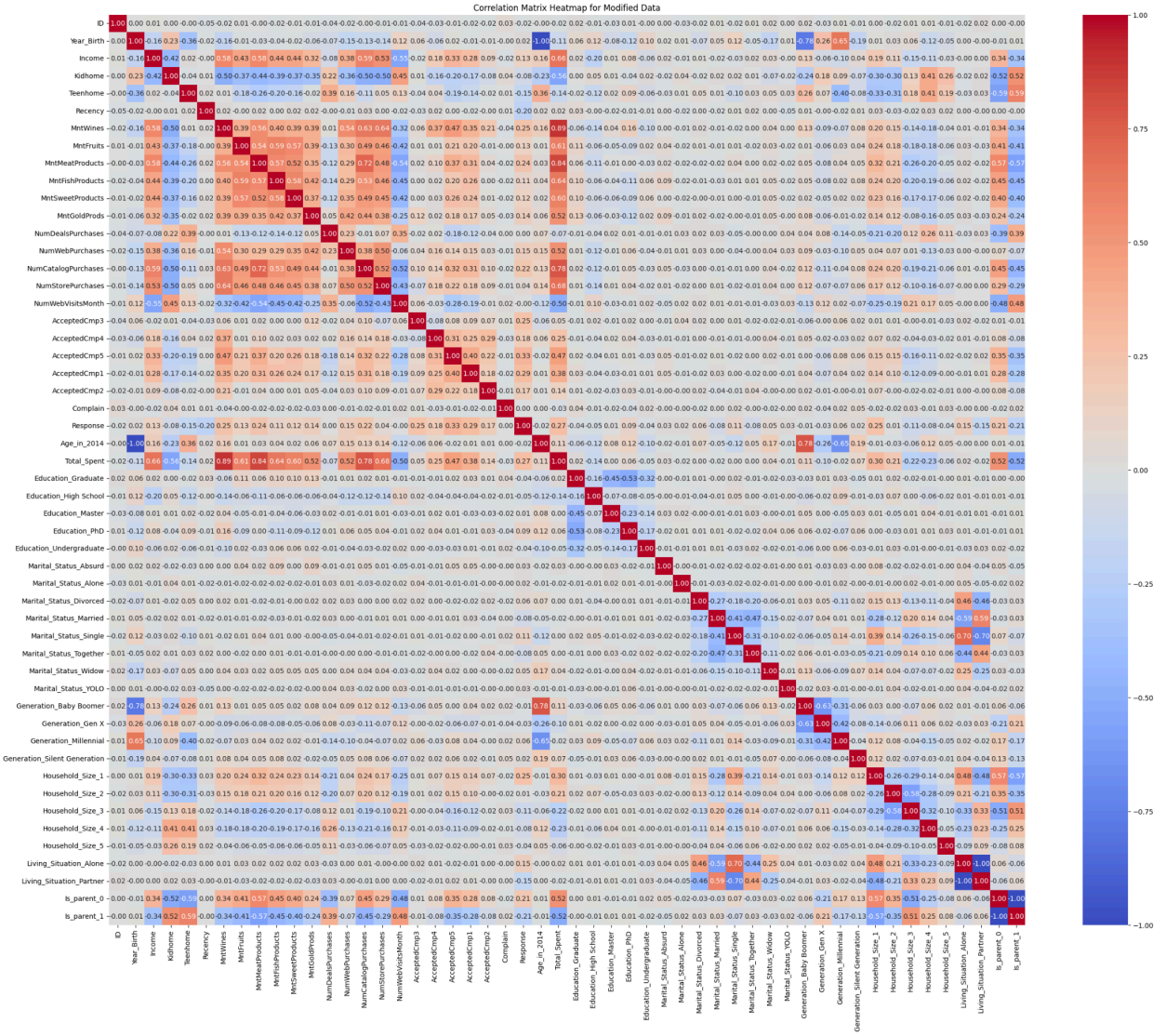
III. Feature Selection

Feature selection aims to identify the most relevant features that contribute to the predictive power of our model. Techniques such as correlation analysis, feature importance, and recursive feature elimination can help us select the most informative features.

Here is our correlation map again for our modified data (see colab link for more bigger picture):



We can see that the date when the customer was first enrolled or became a part of the company's database is significantly correlated to anything else. So we are going to eliminate those features by dropping them. We are also going to make the necessary changes (using labelEncoder) to be able to include ‘Response’ in our map:



We created a code block allowing us to select the top n highest correlating variables to ‘Response’. These are the selected features we are using for our models. Note that we will also be trying our models with all the features we ended up with at the step just above to add more to our comparison.

Part B: Classification (Supervised Learning) (Pre-Deliverable Part B)

Method 1: Decision Tree

Decision trees are a popular algorithm for classification tasks. They partition the feature space into regions and make predictions based on the majority class within each region. Decision trees are interpretable and can handle both numerical and categorical data.

Model Performance

We trained a decision tree classifier using the entire dataset (that we ended up with in the previous section) and evaluated its performance using metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into how well the model generalizes to unseen data and its ability to correctly classify instances.

Here are the results using our base model that includes all variables in the dataset we worked on in part A:

```
Accuracy: 0.8638392857142857
Precision: 0.4918032786885246
Recall: 0.5
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       388
           1       0.49      0.50      0.50        60

    accuracy                           0.86       448
   macro avg       0.71      0.71      0.71       448
weighted avg       0.86      0.86      0.86       448

Time to construct the model: 0.03536725044250488 seconds
```

Here are the results using our base model that includes the top 10 most highly correlated variables to ‘Response’ from the dataset we worked on in part A:

```
Accuracy with selected features: 0.8482142857142857
Precision with selected features: 0.43103448275862066
Recall with selected features: 0.4166666666666667
Classification Report with selected features:
              precision    recall  f1-score   support

           0       0.91      0.91      0.91       388
           1       0.43      0.42      0.42        60

    accuracy                           0.85       448
```

```
   macro avg       0.67      0.67      0.67       448
weighted avg       0.85      0.85      0.85       448

Time to construct the model with selected features: 0.015270709991455078
seconds
```

Not too much change, the second model is a bit faster, but because it is so fast, that isn't the most important deciding factor in our case.

Hyperparameter Tuning

To optimize the decision tree model, we performed hyperparameter tuning using grid search. This technique helps us find the best combination of hyperparameters that maximizes the model's performance.

Here are the results using our hyperparameter tuned model that includes all variables in the dataset we worked on in part A:

```
Accuracy: 0.8816964285714286
Precision: 0.5686274509803921
Recall: 0.48333333333333334
F1 Score: 0.5225225225225224
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       388
           1       0.57      0.48      0.52        60

    accuracy                           0.88       448
   macro avg       0.75      0.71      0.73       448
weighted avg       0.87      0.88      0.88       448

Time to construct the model: 0.03968453407287598 seconds
```

This seems to be the best for accuracy and precision and recall!

Overall results

The decision tree classifier achieved an accuracy of approximately 88%, with precision, recall, and F1-score of 56%, 48%, and 52%, respectively. These results indicate the model's ability to correctly classify instances and its overall performance.

Overall Time to Construct the Model

The time taken to construct the decision tree model was approximately 0.04 seconds. This metric provides insights into the computational efficiency of the model-building process. It runs in a blink, so we think it's pretty good!

Method 2: Gradient Boosting

Let's follow the same style of exploration for this method:

Model with all variables:

```
Accuracy: 0.8861607142857143
Precision: 0.6363636363636364
Recall: 0.35
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       388
           1       0.64      0.35      0.45        60

    accuracy                           0.89       448
   macro avg       0.77      0.66      0.69       448
weighted avg       0.87      0.89      0.87       448

Time to construct the model: 0.8288650512695312 seconds
```

Model with select features:

```
Accuracy: 0.8683035714285714
Precision: 0.5172413793103449
Recall: 0.25
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.96      0.93       388
           1       0.52      0.25      0.34        60

    accuracy                           0.87       448
   macro avg       0.70      0.61      0.63       448
weighted avg       0.84      0.87      0.85       448

Time to construct the model: 0.28998613357543945 seconds
```

Fine tuned model:

```
Accuracy: 0.8973214285714286
Precision: 0.6944444444444444
Recall: 0.4166666666666667
F1 Score: 0.5208333333333334
Classification Report:
               precision    recall  f1-score   support

           0       0.92      0.97      0.94       388
           1       0.69      0.42      0.52        60

    accuracy                           0.90       448
   macro avg       0.80      0.69      0.73       448
weighted avg       0.89      0.90      0.89       448

Time to construct the model: 1.9173784255981445 seconds
```

Method 3: Random Forest

Let's follow the same style or exploration for this method:

Model with all variables:

```
Accuracy: 0.8973214285714286
Precision: 0.75
Recall: 0.35
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94       388
           1       0.75      0.35      0.48        60

    accuracy                           0.90       448
   macro avg       0.83      0.67      0.71       448
weighted avg       0.89      0.90      0.88       448

Time to construct the model: 0.440685510635376 seconds
```

Model with select features:

```
Accuracy: 0.8883928571428571
Precision: 0.65625
Recall: 0.35
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.97      0.94       388
           1       0.66      0.35      0.46        60

    accuracy                           0.89       448
   macro avg       0.78      0.66      0.70       448
weighted avg       0.87      0.89      0.87       448

Time to construct the model: 0.34423089027404785 seconds
```

Fine tuned model:

```
Accuracy: 0.890625
Precision: 0.7894736842105263
Recall: 0.25
F1 Score: 0.379746835443038
Classification Report:
               precision    recall  f1-score   support

           0       0.90      0.99      0.94       388
           1       0.79      0.25      0.38        60

    accuracy                           0.89       448
   macro avg       0.84      0.62      0.66       448
weighted avg       0.88      0.89      0.86       448

Time to construct the model: 0.9609782695770264 seconds
```

Review of Model Performances (Deliverable Part B Summary):

Decision Tree:

- Achieved 86% accuracy with all features.
- Precision and recall were 49% and 50%, respectively.
- F1-score reached 52%.
- Fast runtime: 0.04 seconds.
- Hyperparameter tuning boosted accuracy to 88%.
- Overall, it showed good performance with short construction time.

Gradient Boosting:

- Reached 89% accuracy with all features.
- Precision and recall: 64% and 35%.
- F1-score: 45%.
- Longer runtime: 0.83 seconds.
- After feature selection, accuracy slightly decreased to 87%, but construction time improved to 0.29 seconds.
- Fine-tuning further improved accuracy to 90%.

Random Forest:

- Attained 90% accuracy with all features.
- Precision and recall: 75% and 35%.
- F1-score: 48%.
- Moderate runtime: 0.44 seconds.
- After feature selection, accuracy slightly dropped to 89%, but construction time improved to 0.34 seconds.
- Fine-tuned model maintained accuracy at 89% with improved precision and recall.

Overall Review:

- Models performed better with all features compared to selected features, suggesting potential underfitting with feature selection.
- Hyperparameter tuning improved accuracy and other metrics for decision tree and gradient boosting models.
- Construction time varied, with decision tree being the fastest and gradient boosting the slowest.

- Fine-tuned models generally showed improved performance metrics, indicating the effectiveness of hyperparameter tuning.
- Random forest demonstrated competitive performance with relatively shorter construction time compared to gradient boosting.
- In summary, gradient boosting showed the best overall performance, albeit with longer construction time, while random forest offered a good balance between performance and efficiency.

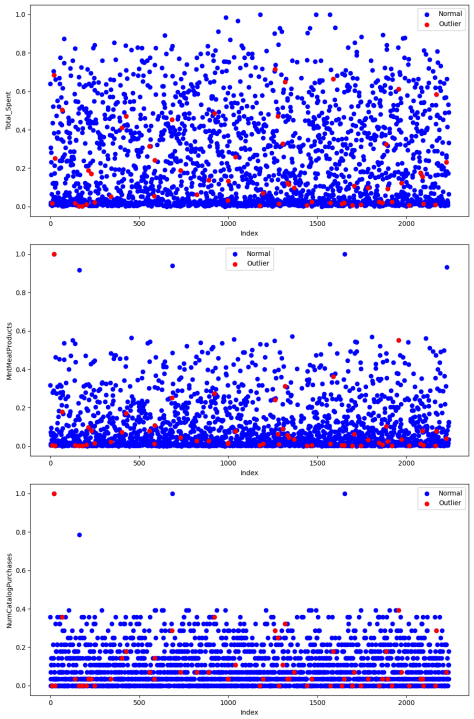
SVM (Deliverable Part C)

Global Outliers

We found the following global outliers when performing SVM on all our final data:

```
Indices of outliers:
[   9   21   27   67  138  160  178  204  215  230  250  341  401  427
  556  583  587  684  731  823  891  921  997  999 1039 1176 1196 1262
 1278 1285 1305 1320 1331 1339 1369 1441 1470 1573 1589 1638 1650 1696
 1707 1746 1785 1847 1866 1888 1895 1920 1958 1972 2015 2081 2090 2095
 2164 2168 2225]
```

While analyzing these outliers, we plotted a few attributes to gain insights into their characteristics. However, due to the complexity and multidimensionality of the data, it was challenging to discern the specific factors contributing to their outlier status visually.



Nevertheless, this challenge underscores the value of utilizing computational methods to detect outliers, as they can reveal patterns and anomalies that may not be immediately apparent to human observers. By leveraging advanced algorithms like SVM, we can efficiently identify outliers and scrutinize their underlying properties, enabling us to make informed decisions and derive meaningful insights from our data.

In conclusion, while visual inspection may not always provide definitive answers, leveraging technology enhances our ability to detect and analyze outliers effectively, ultimately empowering us to make data-driven decisions with confidence.

Team Planning

Deliverable Checklist	Sub-Tasks	Responsible Team Members	Completion Date
Data processing	Data Summarisation (Visualisation)	Coralie	April 2, 2024
	Data Transformation (Missing values, categorical data, numeric data, feature selection)	Coralie	April 2, 2024
Data Mining - Classification	Decision Tree	Coralie	April 2, 2024
	Gradient Boosting	Tanner	April 3, 2024
	Random Forest	Tanner	April 3, 2024
	Comparison of results	Coralie and Tanner	April 3, 2024
	Summary	Coralie and Tanner	April 4, 2024
Anomaly Detection	One-Class SVM	Coralie and Dana	April 4, 2024
	Summary	Coralie and Dana	April 4, 2024
Document creation and organization		Coralie	March 31, 2024
Document writing		Everyone	April 5, 2024
Team planning		Coralie	April 5, 2024

Note: The distribution of team planning may appear uneven on this tracker because Phase 3 was led by Dana, while Phase 4 was primarily focused on by Tanner and Coralie. We decided to split the work in this manner because the two phases were due on the same day. We can assure you that this was still a collaborative process and time spent on all tasks was evenly spent. :)

Colab Code

https://colab.research.google.com/drive/1DP1i6BoRLEJYO1Zz9gvMgE2qy1y0u2li?usp=sharing