

**EEC 525 / CIS 660 DATA MINING**

**LAB 1**

**Part 3: Calculating Proximity of Two Binary Object Vectors With Simple Matching, Jaccard Similarity, Cosine Similarity**

**&**

**Part 4: Correlation Analysis to Discover Any Relationship between Any Two Features by Building a Correlation Matrix**

Submitted By

**Dinky Mishra**

**CSU ID: 2864923**

**Table of contents**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Content** | **Page No** |
| 1 | **Part 3: Calculating Proximity of Two Binary Object Vectors With Simple Matching, Jaccard Similarity, Cosine Similarity**  1.1. Normalization of Attributes  1.2. Application & Calculation of Correct Similarity Measure | 3-8 |
| 2 | **Part 4: Correlation Analysis to Discover Any Relationship between Any Two Features by Building a Correlation Matrix**  2.1. Correlation Matrix for Entire Data Set  2.2. Heat Map Visualization of Correlation Matrix  2.3. Data Set Division Based on Bike Buyer Status  2.4. Correlation Matrix for Bike Buyer = 1  2.5. Correlation Matrix for Bike Buyer = 0  2.6. Comparison of Correlation Values  2.7. Discussion on Strongest Correlations | 9-16 |
| 3 | **Extra Credit: Dissimilarity Matrix using Cosine** | 17-21 |

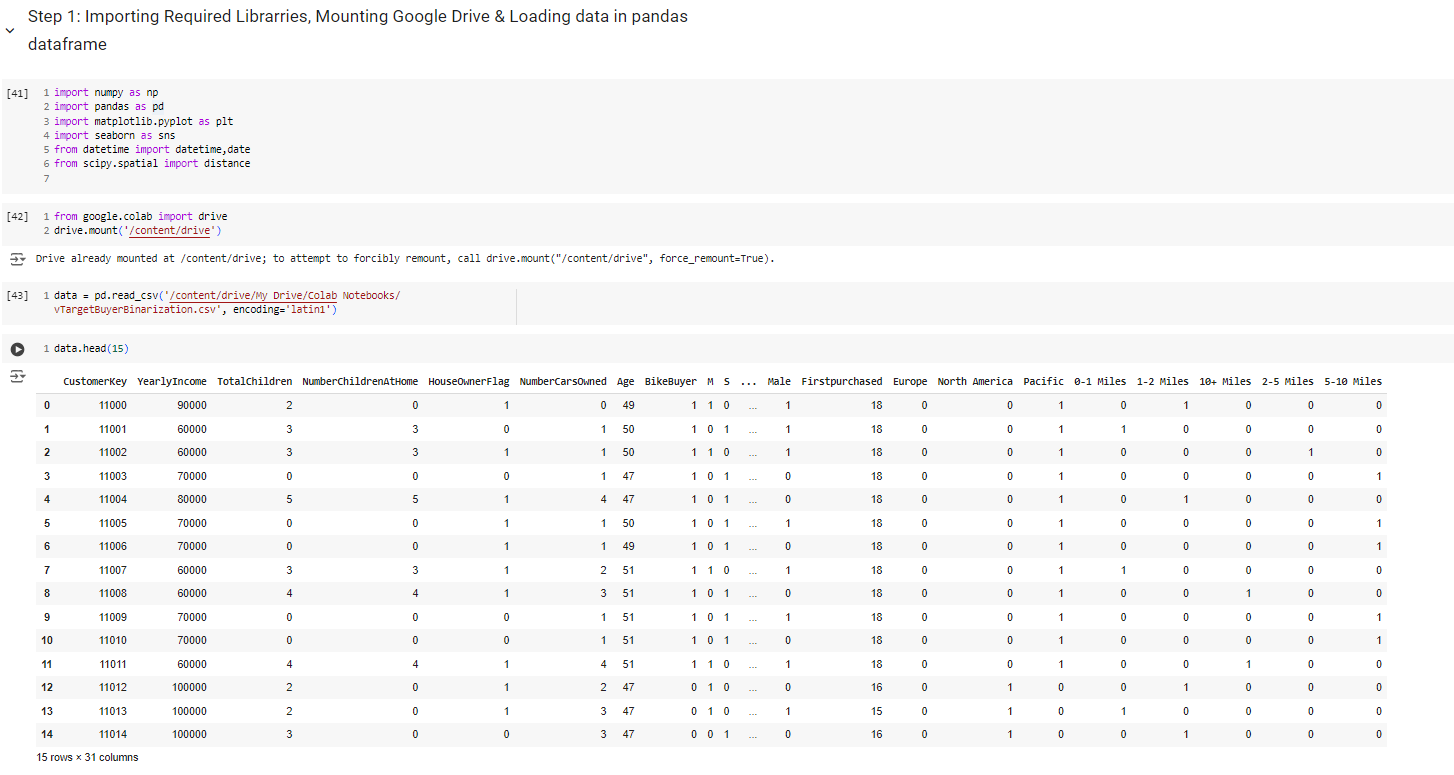
**Part 3: Calculating Proximity of Two Binary Object Vectors With Simple Matching, Jaccard Similarity, Cosine Similarity**

**1.1. Normalization of Attributes**

**Make sure that all the attributes are transformed in a same scale in normalization at the end for all the numeric attributes, all the categorical (nominal) attributes, and all the discretized numeric attribute.**

#### Data Loading and Normalization

This part of the code imports necessary libraries and mounts Google Drive to access the dataset. It then loads the dataset and displays the first 2 rows and the dataset information.



Normalized all the categorial(nominal) attributes:

This part normalizes the numeric attributes to ensure they are on the same scale. It then drops the original non-normalized columns.



## This part of the code displays 2 rows of the data information and then saving the dataframe to “Lab1\_part2&3\_normalized\_data.csv” file . Then load the data dataframe to df.

## 

## **1.2. Application & Calculation of Correct Similarity Measure Make sure to apply a correct similarity measure for nominal (one hot encoding)/binary attributes and numeric attributes respectively. You can apply an identity (indicator) variable or a different weight for each attribute as needed.**

## **Calculate a Similarity measure between a pair of two objects below from your transformed input data based on the following measures in:**

1. **Simple Matching**
2. **Extended Jaccard Similarity**
3. **Cosine Similarity**

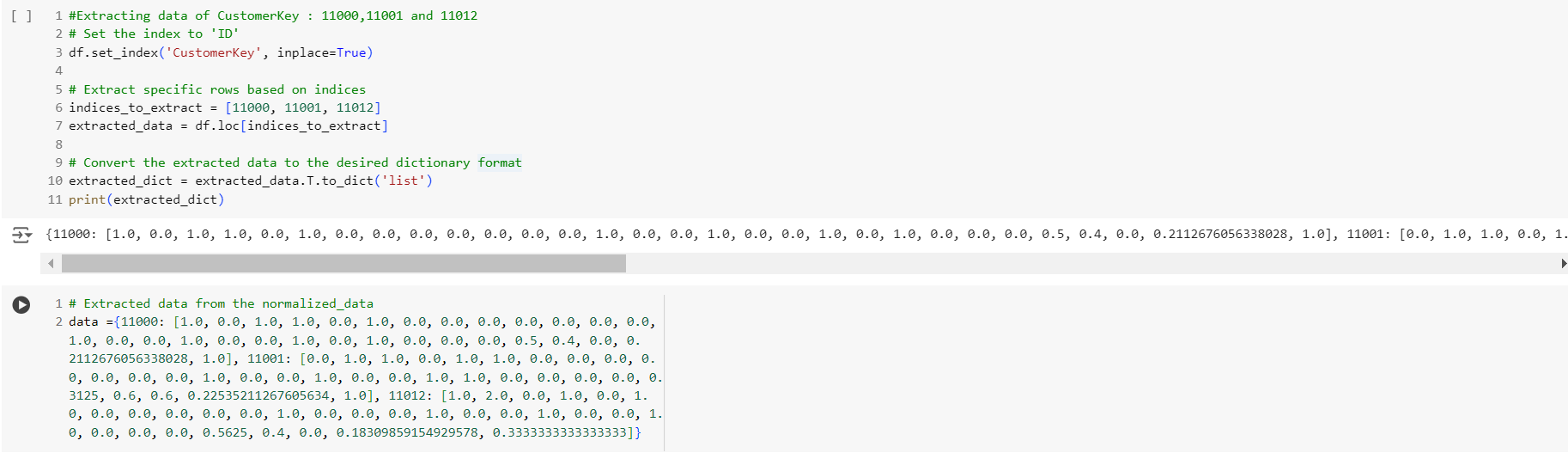
## **Use the CustomerKey column value to locate two object vectors to compare. You should NOT include the CustomerKey column as a part of the feature vector.**

## **1) Similarity between two vectors of (CustomerKey = 11000) and (CustomerKey = 11001)**

## **2) Similarity between two vectors of (CustomerKey = 11000) and (CustomerKey = 11012)**

#### Extracting Specific Rows and Calculating Similarity Measures:

This part extracts specific rows based on CustomerKey and calculates the similarity measures (Simple Matching, Extended Jaccard, and Cosine Similarity) between the pairs of vectors (11000, 11001) and (11000, 11012). The similarity measures are calculated using custom functions without using built-in libraries

The below code does the following :

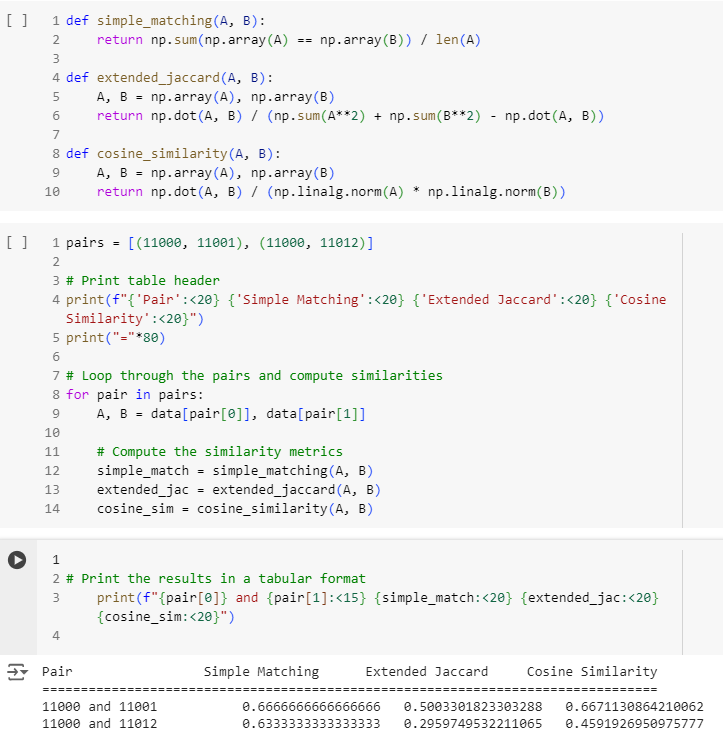
1.Loop Through Pairs: The code iterates over the list of pairs, which are tuples of CustomerKey values.

2.Extract Vectors: For each pair, it extracts the corresponding vectors A and B from the data dictionary.

3.Calculate Similarities: It calculates three similarity measures between the vectors:

* + Simple Matching: Proportion of matching attributes.
  + Extended Jaccard: Ratio of the dot product to the sum of squared elements minus the dot product.
  + Cosine Similarity: Dot product divided by the product of their magnitudes.

4.Print Results: It prints the similarity measures for each pair.



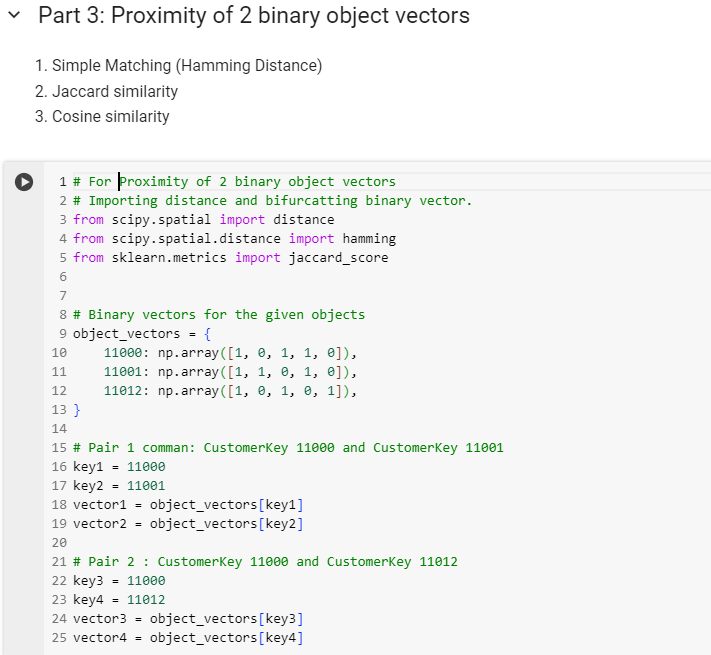
The below part of the code performs proximity of 2 binary vectors:

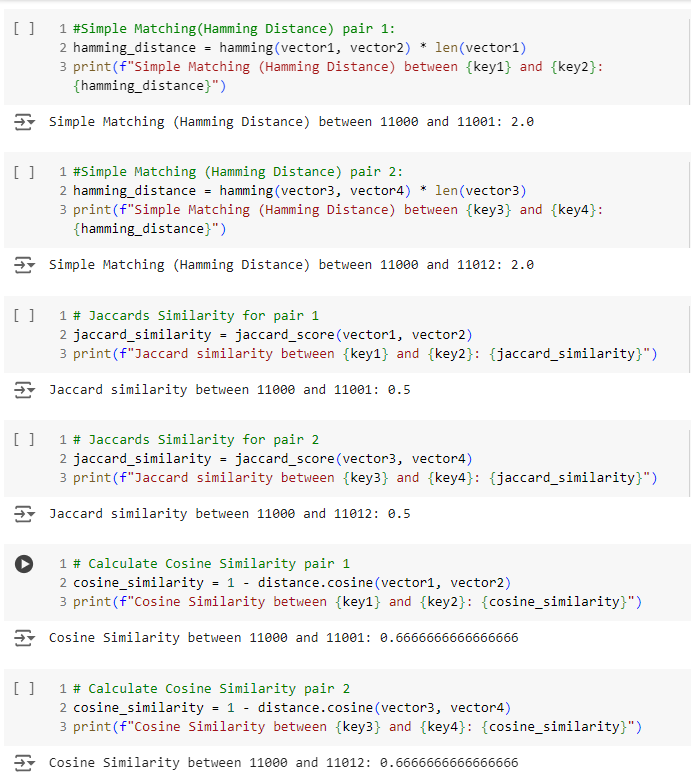
1.Extract Vectors: The vectors vector1, vector2, vector3, and vector4 are predefined for specific CustomerKey values.

2.Calculate Similarities: It calculates three similarity measures between the pairs of vectors:

* Simple Matching (Hamming Distance): Number of differing positions between the binary vectors, scaled by the length of the vectors.
* Jaccard Similarity: Ratio of the intersection to the union of the binary vectors.
* Cosine Similarity: Dot product divided by the product of their magnitudes.

3.Print Results: It prints the similarity measures for each pair.





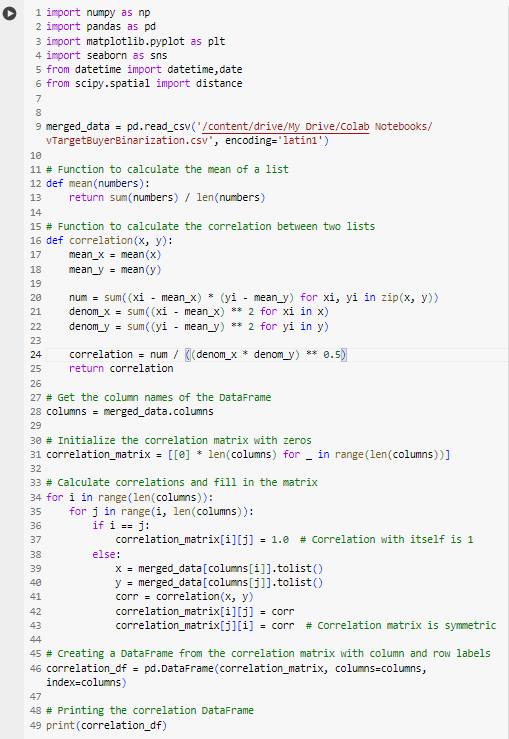
Both parts of the code aim to measure similarity but use different methods and data sources to achieve this. The first part is more comprehensive, while the second part focuses on specific types of similarity measures for binary vectors.

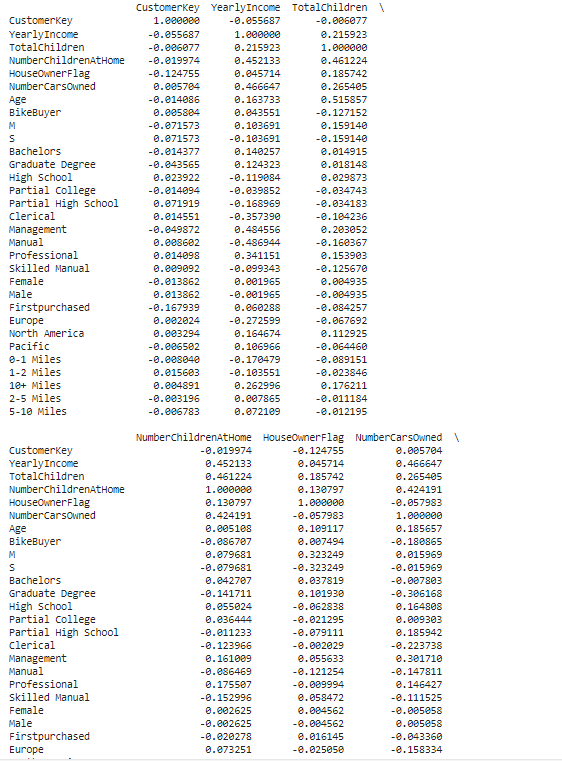
Part 4: Correlation Analysis to Discover Any Relationship between Any Two Features by Building a Correlation Matrix

**2.1. Correlation Matrix for Entire Data Set**

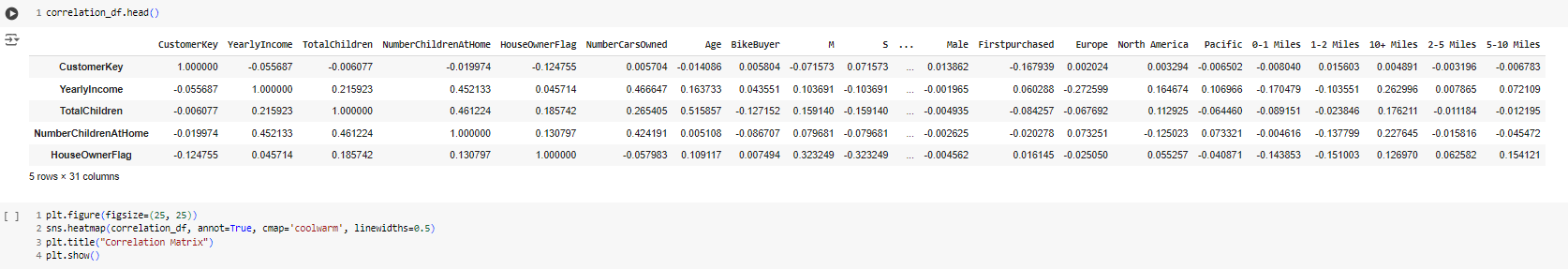
**1: Building a Correlation Matrix for the Entire Dataset  
This part calculates the correlation matrix for every pair of features in the entire dataset and prints the dataframe “correlation\_df” in the following manner:**

* The dataset is loaded into a DataFrame.
* A custom function correlation is defined to calculate the correlation between two lists.
* The correlation matrix is initialized and filled by calculating the correlation for each pair of features.
* The resulting correlation matrix is converted into a DataFrame for better readability.





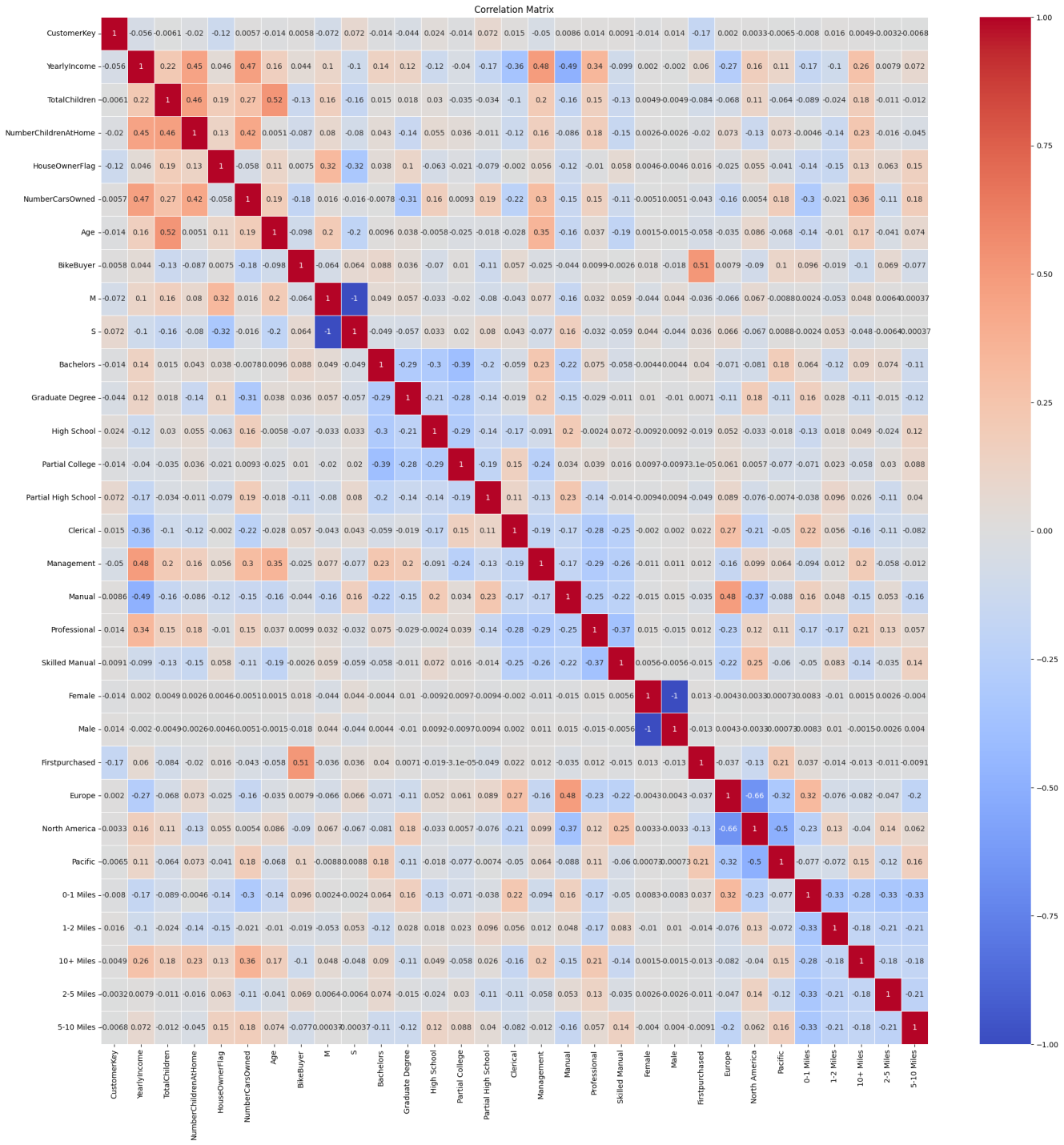
Displays correlation\_df:



**2.2. Heat Map Visualization of Correlation Matrix**

**2. Visualize Your Correlation Matrix of All the Features in a Heat Map**  
The correlation matrix is visualized using a heatmap.

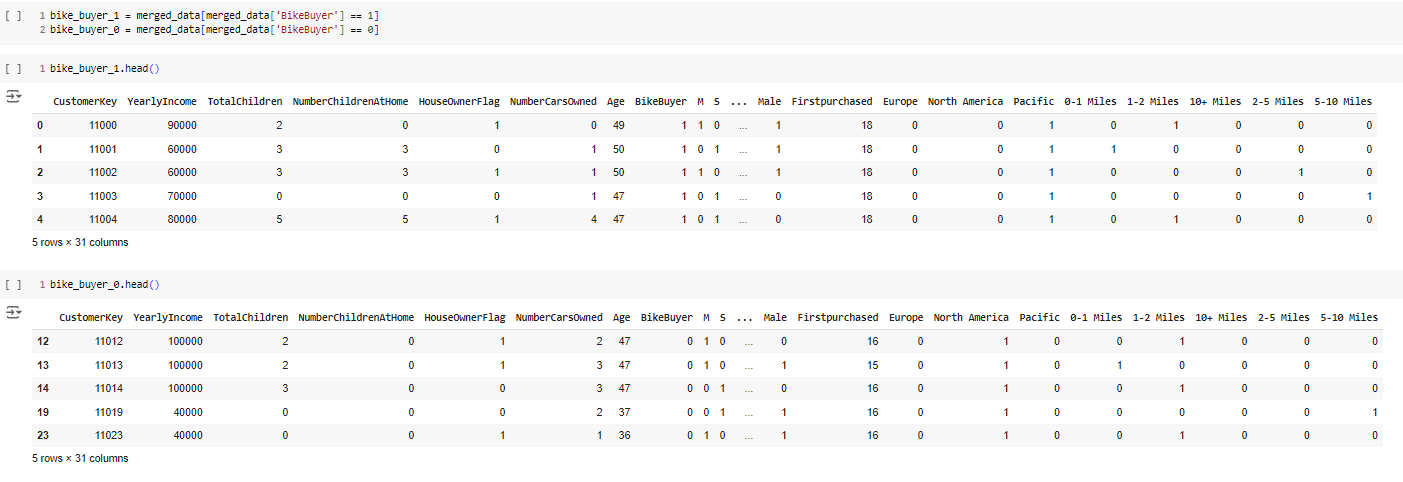
* The seaborn library is used to create the heatmap, with annotations and a color map to indicate the strength of correlations.



**2.3. Data Set Division Based on Bike Buyer Status**

**3. Divide your data set into two data sets: One with Bike Buyer = 1 and the other set with Bike Buyer = 0**  
The dataset is split into two subsets based on the BikeBuyer column.

* bike\_buyer\_1 contains records where BikeBuyer is 1.
* bike\_buyer\_0 contains records where BikeBuyer 0

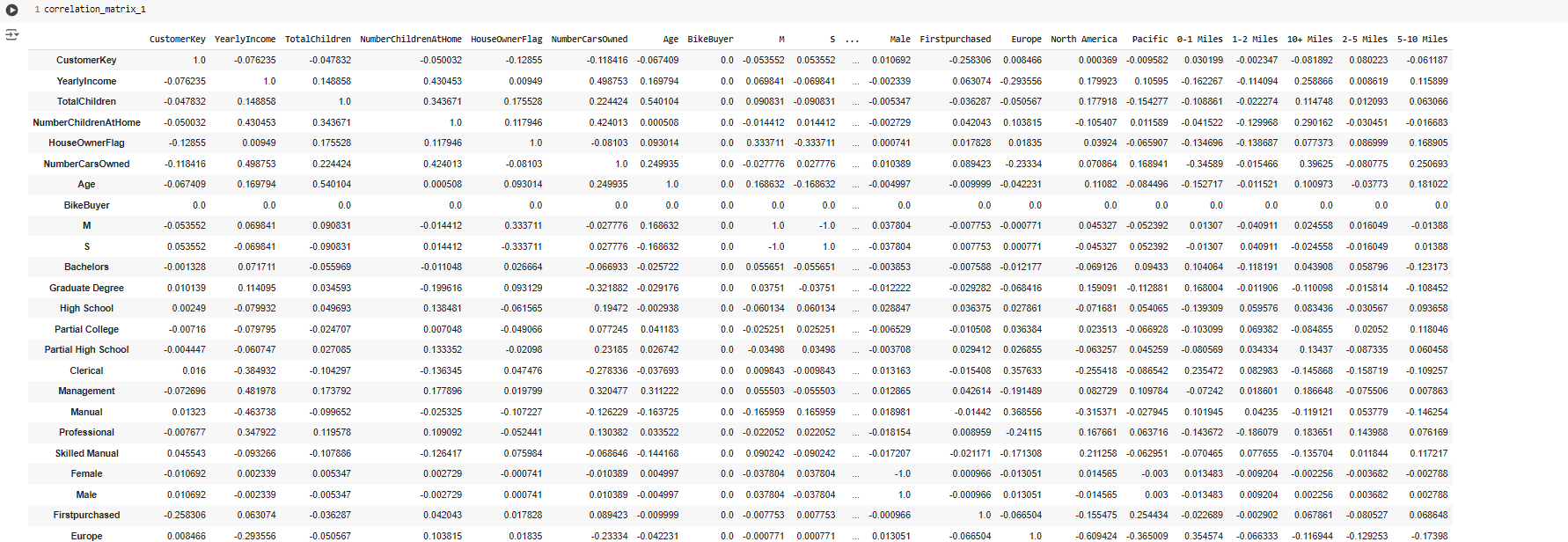


**2.4. Correlation Matrix for Bike Buyer = 1 & Bike Buyer = 0**

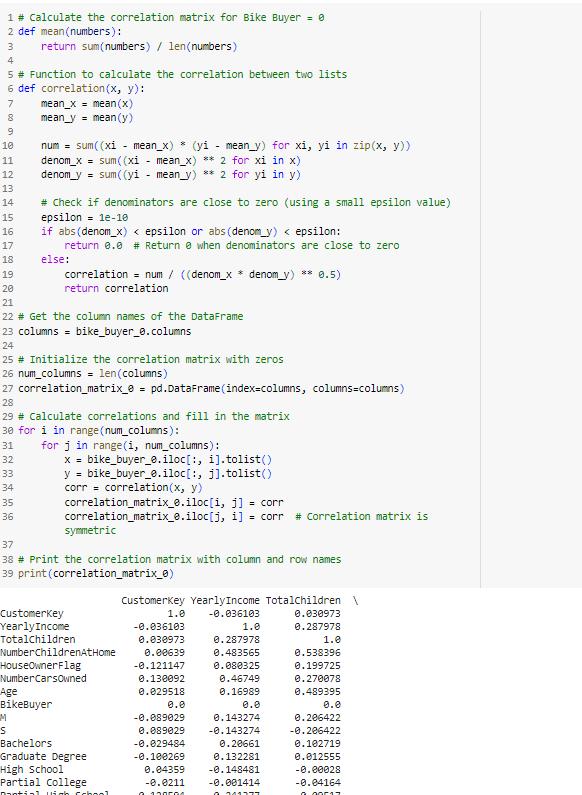
**4. Then build a Correlation Matrix for Every pair of the Features for the record set with Bike Buyer = 1 and the other record set with Bike Buyer = 0 respectively.**  
This part of the code builds a Correlation matrix for every pair of features for the record set with Bike Buyer =1 in the following manner:

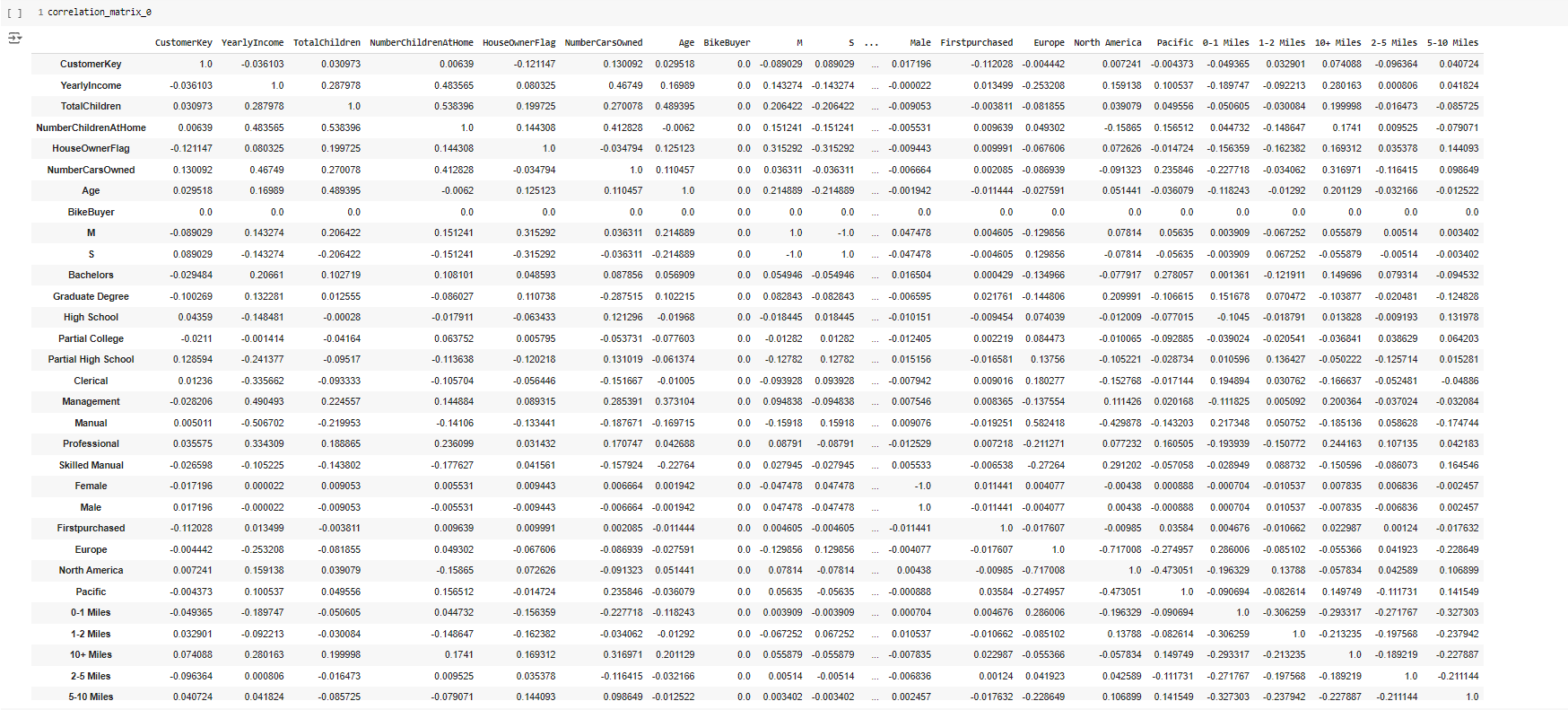
* The correlation matrices are calculated separately for the two subsets (bike\_buyer\_1 and bike\_buyer\_0).
* The same custom correlation function is used to fill the matrices.
* The resulting matrices are printed for inspection

## 



This part of the code builds a Correlation matrix for every pair of features for the record set with Bike Buyer =0 and prints the correlation matrix below:





**2.5. Comparison of Correlation Values**

**5. From the three Correlation Matrices built above, Compare the Correlation values between two features Age and Yearly Income and the Correlation values between two features Commute Distance and Yearly Income.**

In the following ,code snippet the CommuteDistance has been normalized in Part 3.  
Therefore,the correlation has been compared accordinlgy.

* The correlation values between Age and YearlyIncome and between CommuteDistance and YearlyIncome are extracted from the correlation matrices.
* These values are printed for comparison.



**2.6. Discussion on Strongest Correlations**

**6.Compare and discuss which two features are correlated more strongly than the others for each correlation matrix.**

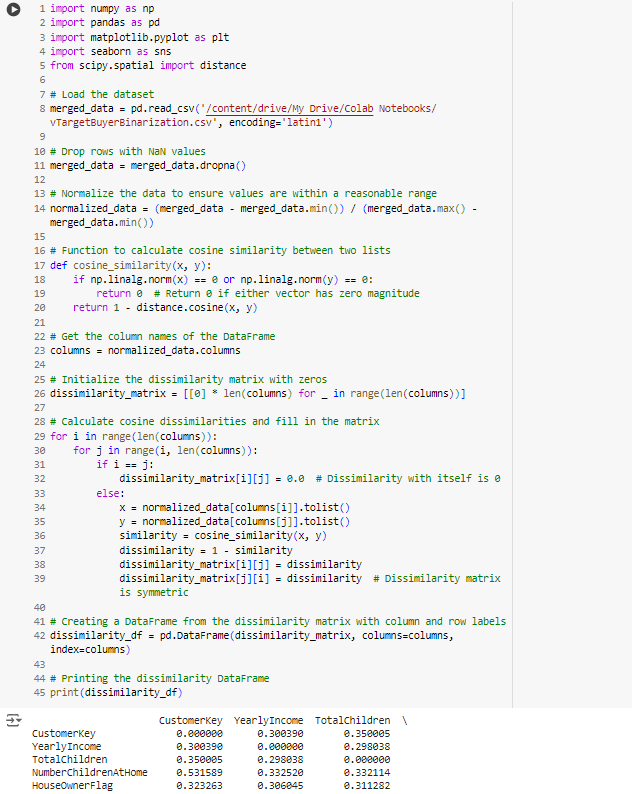
* **Strongest Positive Correlation**: There is a strong positive correlation between age and yearly income, which means that as people get older, they generally earn more. Similarly, there's also a strong positive relationship between commute distance and yearly income, indicating that those who earn higher incomes tend to commute longer distances.
* **Moderate Positive Correlation**: Bike buyers who travel more than 10 miles have a moderate positive correlation (0.26) with their annual income, suggesting that people with longer commutes might earn more.
* **Key Insights**: Correlations close to 1 indicate a strong positive relationship, while values near -1 indicate a strong negative relationship.

**Extra Credit: Dissimilarity Matrix using Cosine**

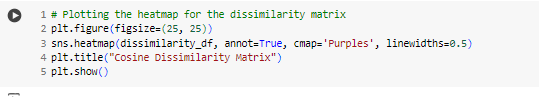
**Perform a dissimilarity matrix using cosine similarity on the above dataset for EXTRA CREDIT:**

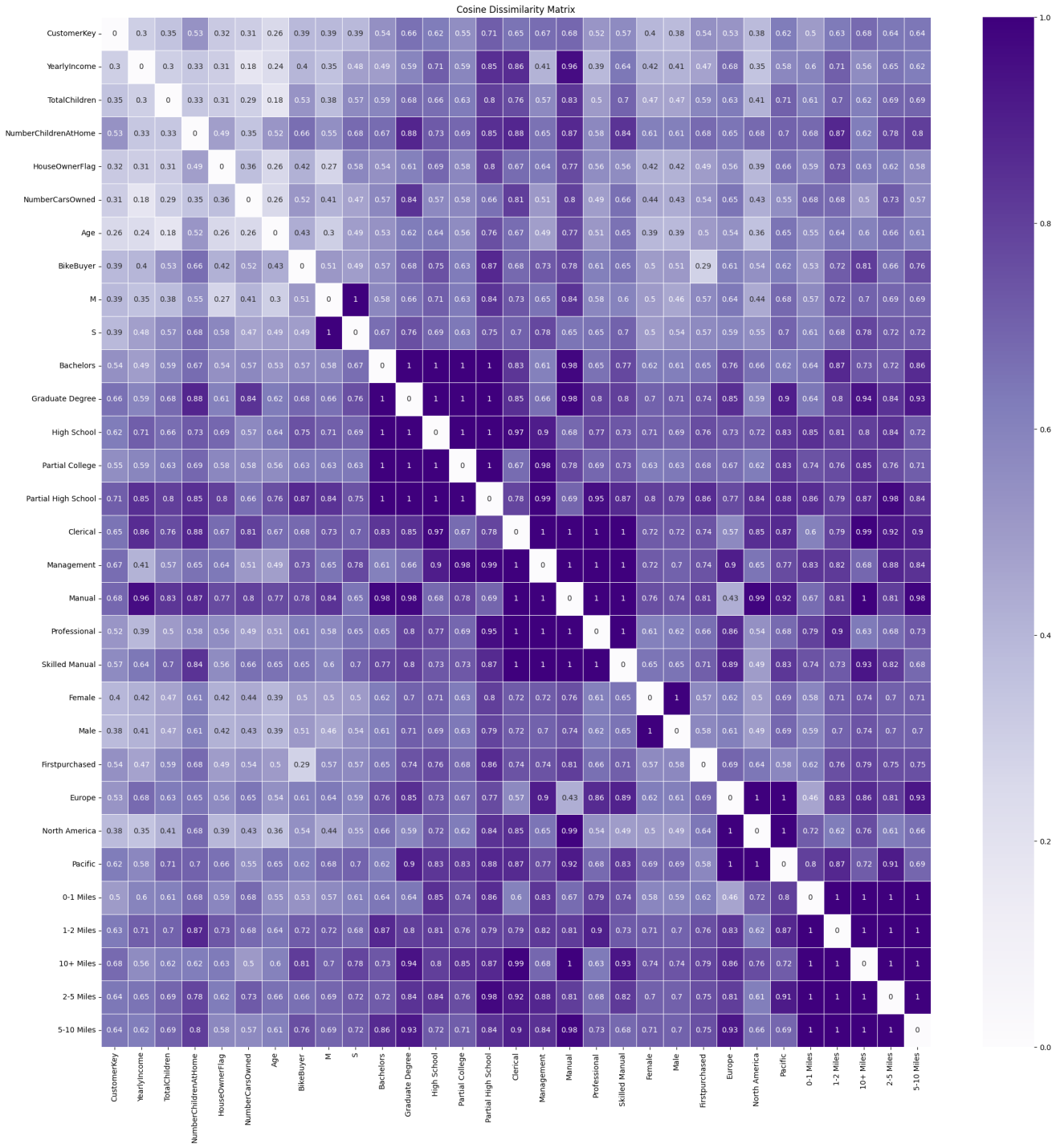
* The necessary libraries are imported.
* The dataset is loaded into a DataFrame.
* Rows with missing values are dropped to ensure clean data.
* The data is normalized to bring all features to a common scale, which is essential for calculating cosine similarity.
* A function cosine\_similarity is defined to calculate the cosine similarity between two vectors.
* If either vector has zero magnitude, the function returns 0 to avoid division by zero.
* The cosine similarity is calculated using the distance.cosine function from scipy.spatial.
* The column names of the normalized DataFrame are retrieved.
* A dissimilarity matrix is initialized with zeros.
* The cosine dissimilarity between each pair of features is calculated and stored in the matrix.
* The dissimilarity between a feature and itself is set to 0.
* The matrix is symmetric, so the dissimilarity value is mirrored across the diagonal.
* The dissimilarity matrix is converted into a DataFrame for better readability.
* The DataFrame is printed to inspect the values.

Code Snippet:



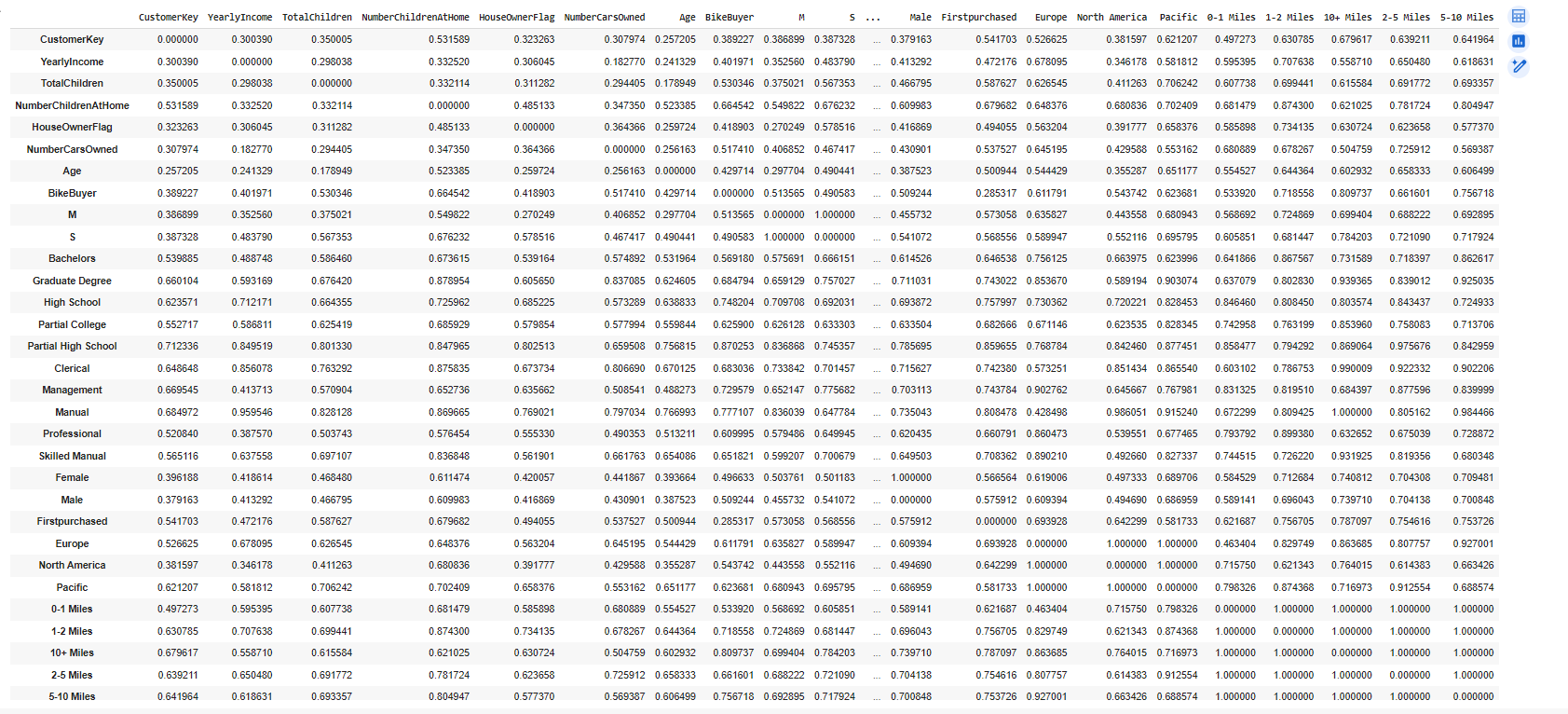
* The heatmap uses a purple color map to indicate the dissimilarity values, with annotations for clarity for full dissimilarity matrix using cosine





Similarly, the same process for building a lower triangular dissimilarity matrix using cosine similarity.

The following code snippet is shown below and the dissimilarity matrix is displayed for reference.



* The heatmap uses a purple color map to indicate the dissimilarity values, with annotations for clarity for lower triangluar dissimilarity matrix using cosine

