PROJECT DM PRESENTATION

DETAILS OF PROJECT

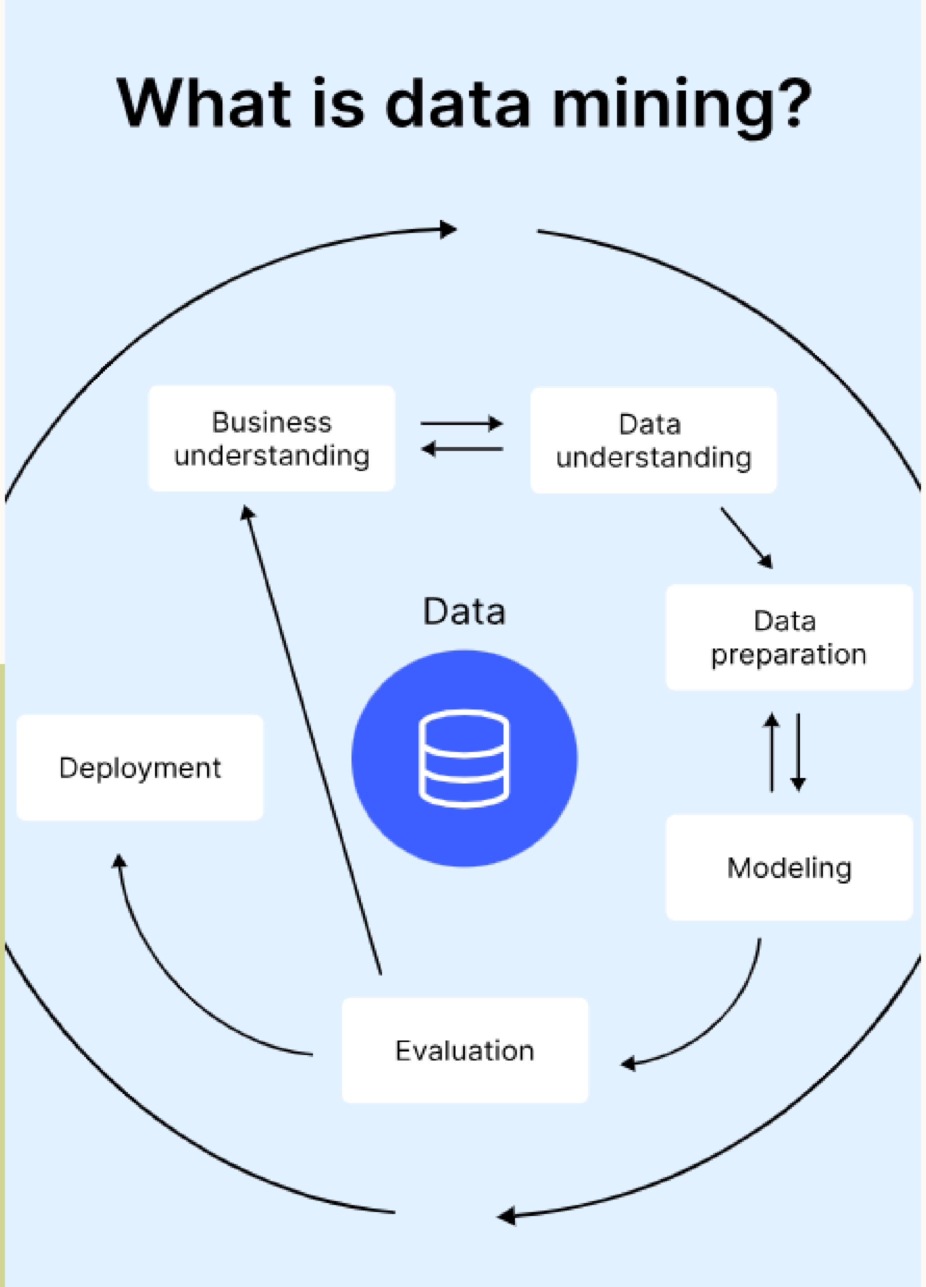
Introduction

Why we choose the data ?

Clean the data

Algorithms that we use

Explain results



DATASET :
A HOTEL'S CUSTOMERS DATASET

DESCRIPTION :THE SELECTED DATASET IS A COMPREHENSIVE COLLECTION OF CUSTOMER TRANSACTIONAL DATA, CONTAINING INFORMATION ABOUT THE PURCHASING BEHAVIOR OF CUSTOMERS IN A PARTICULAR BUSINESS DOMAIN. IT INCLUDES VARIABLES SUCH AS CUSTOMER ID, TRANSACTION DATE, PURCHASED ITEMS, QUANTITIES, AND MONETARY VALUES. THE DATASET COVERS A SUBSTANTIAL TIME PERIOD, CAPTURING A SIGNIFICANT NUMBER OF CUSTOMER TRANSACTIONS.

COLUMN THAT WE WORK ON IT

NATIONALITY : COUNTRY OF ORIGIN. CATEGORIES ARE REPRESENTED IN THE ISO 3155-3:2013

LODGING REVENUE : TOTAL AMOUNT SPENT ON LODGING EXPENSES BY THE CUSTOMER (IN EUROS). THIS VALUE INCLUDES ROOM, CRIB, AND OTHER RELATED LODGING

OTHER REVENUE :TOTAL AMOUNT SPENT ON OTHER EXPENSES BY THE CUSTOMER (IN EUROS). THIS VALUE INCLUDES FOOD, BEVERAGE, SPA, AND OTHER

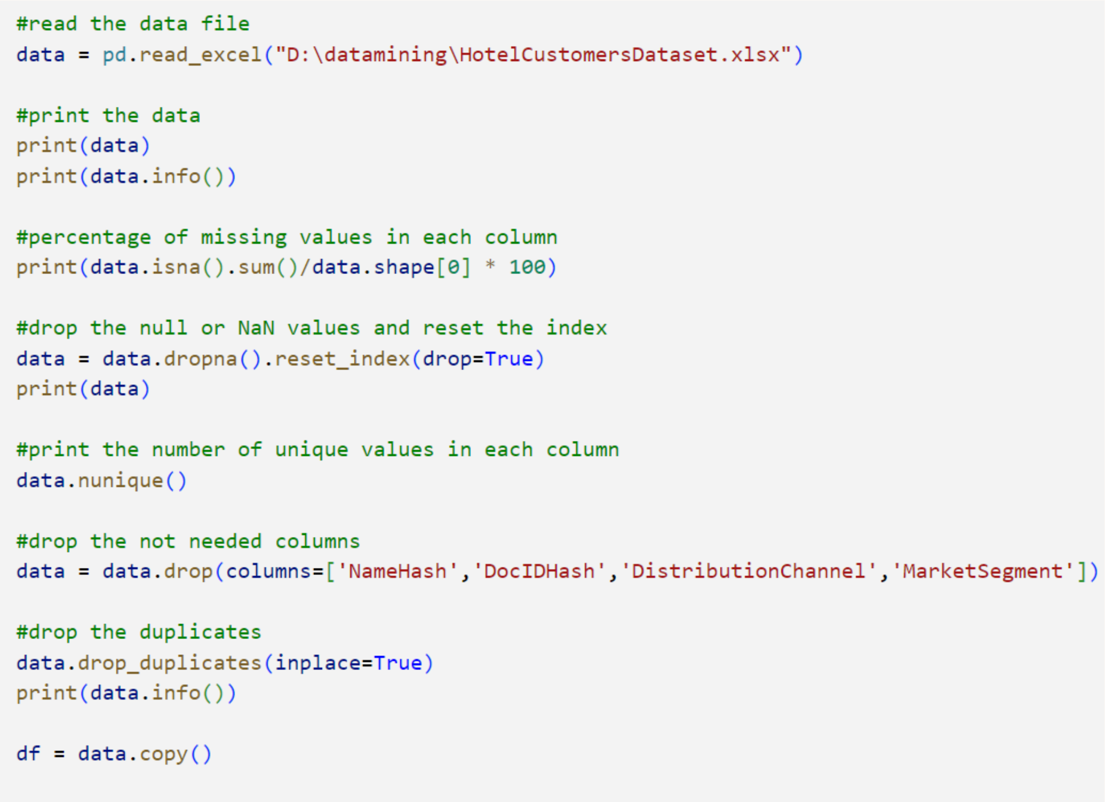
MARKET SEGMENT : CURRENT MARKET SEGMENT OF THE CUSTOMER

DISTRIBUTION CHANNEL : DISTRIBUTION CHANNEL USUALLY USED BY THE CUSTOMER TO MAKE BOOKINGS AT THE HOTEL

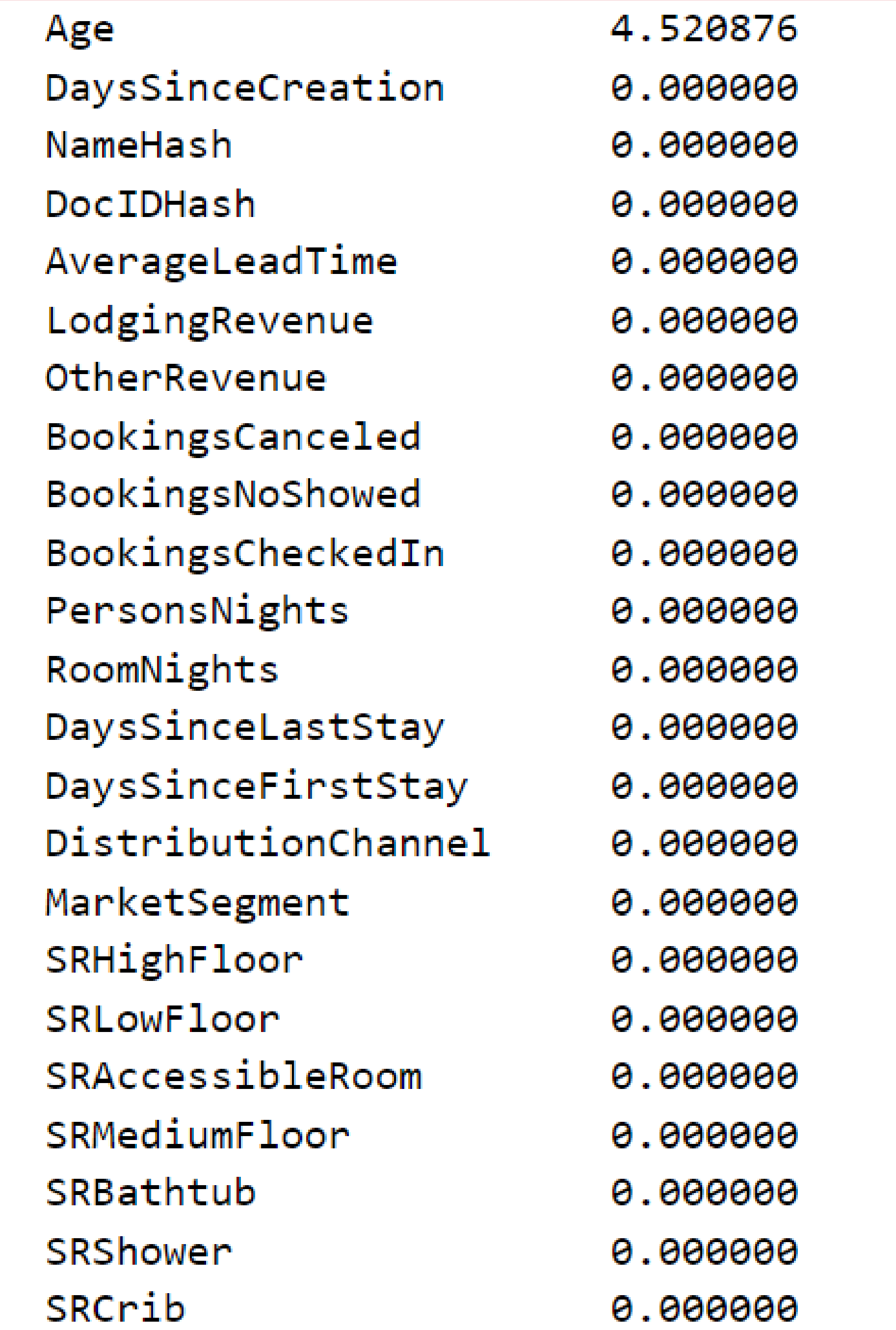
AGE CUSTOMER'S : AGE (IN YEARS) AT THE LAST DAY OF THE EXTRACTION PERIOD

BOOKINGS CANCELED : NUMBER OF BOOKINGS THE CUSTOMER MADE BUT SUBSEQUENTLY CANCELED (THE COSTUMER INFORMED THE HOTEL HE/SHE WOULD

DATA CLEANING

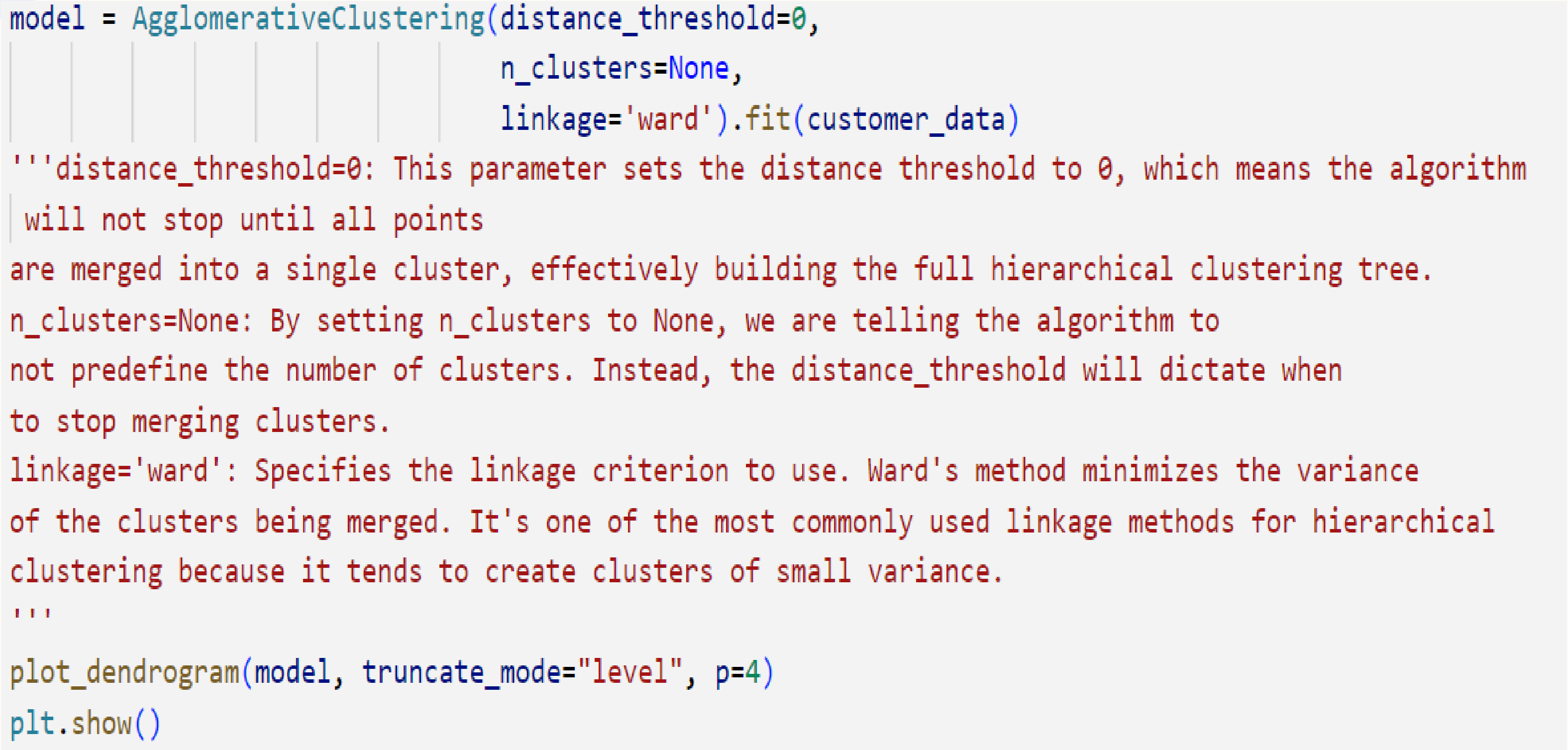


```
#read the data file
data = pd.read_excel("D:\datamining\HotelCustomersDataset.xlsx")

#print the data
print(data)
print(data.info())

#percentage of missing values in each column
print(data.isna().sum()/data.shape[0] * 100)

#drop the null or NaN values and reset the index
data = data.dropna().reset_index(drop=True)
print(data)

#print the number of unique values in each column
data.nunique()

#drop the not needed columns
data = data.drop(columns=['NameHash','DocIDHash','DistributionChannel','MarketSegment'])

#drop the duplicates
data.drop_duplicates(inplace=True)
print(data.info())

df = data.copy()
```



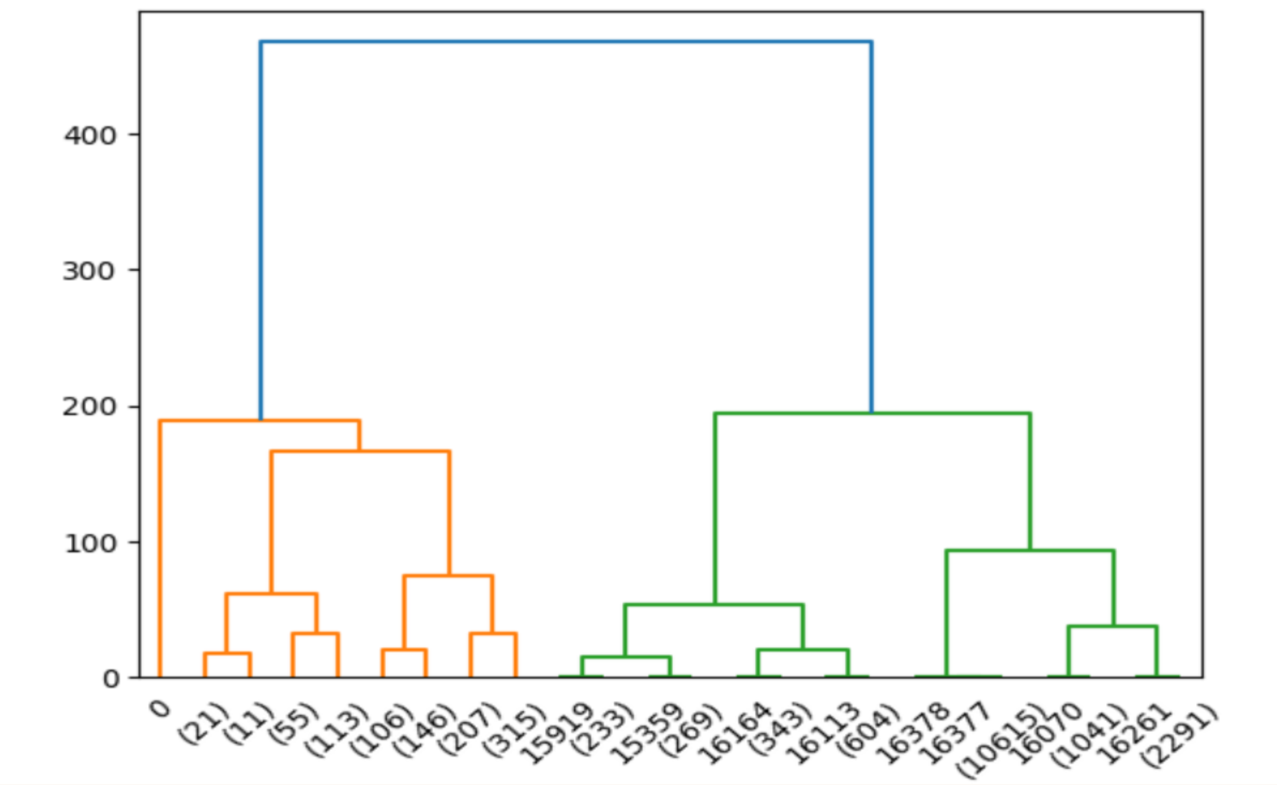
```
Age                    4.520876
DaysSinceCreation      0.000000
NameHash               0.000000
DocIDHash              0.000000
AverageLeadTime        0.000000
LodgingRevenue         0.000000
OtherRevenue           0.000000
BookingsCanceled       0.000000
BookingsNoShowed       0.000000
BookingsCheckedIn      0.000000
PersonsNights          0.000000
RoomNights             0.000000
DaysSinceLastStay      0.000000
DaysSinceFirstStay     0.000000
DistributionChannel    0.000000
MarketSegment          0.000000
SRHighFloor            0.000000
SRLowFloor             0.000000
SRAccessibleRoom       0.000000
SRMediumFloor          0.000000
SRBathtub              0.000000
SRShower               0.000000
SRCrib                 0.000000
```

HIERARCHICAL CLUSTERING: AGGLOMERATIVE CLUSTERING

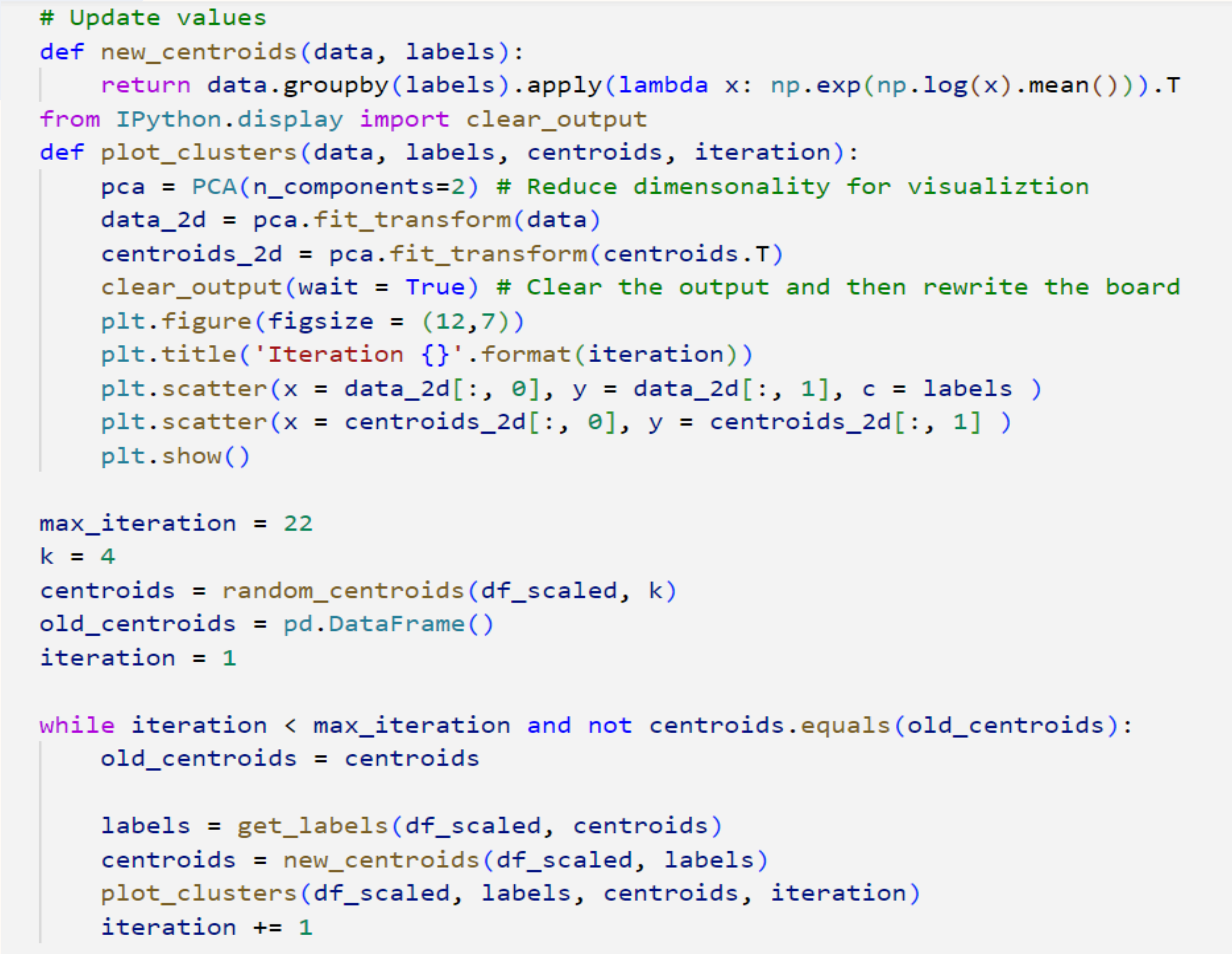


```
model = AgglomerativeClustering(distance_threshold=0,
                               n_clusters=None,
                               linkage='ward').fit(customer_data)
'''distance_threshold=0: This parameter sets the distance threshold to 0, which means the algorithm
 will not stop until all points
are merged into a single cluster, effectively building the full hierarchical clustering tree.
n_clusters=None: By setting n_clusters to None, we are telling the algorithm to
not predefine the number of clusters. Instead, the distance_threshold will dictate when
to stop merging clusters.
linkage='ward': Specifies the linkage criterion to use. Ward's method minimizes the variance
of the clusters being merged. It's one of the most commonly used linkage methods for hierarchical
clustering because it tends to create clusters of small variance.
'''
plot_dendrogram(model, truncate_mode="level", p=4)
plt.show()
```

VISUALIZE RESULTS

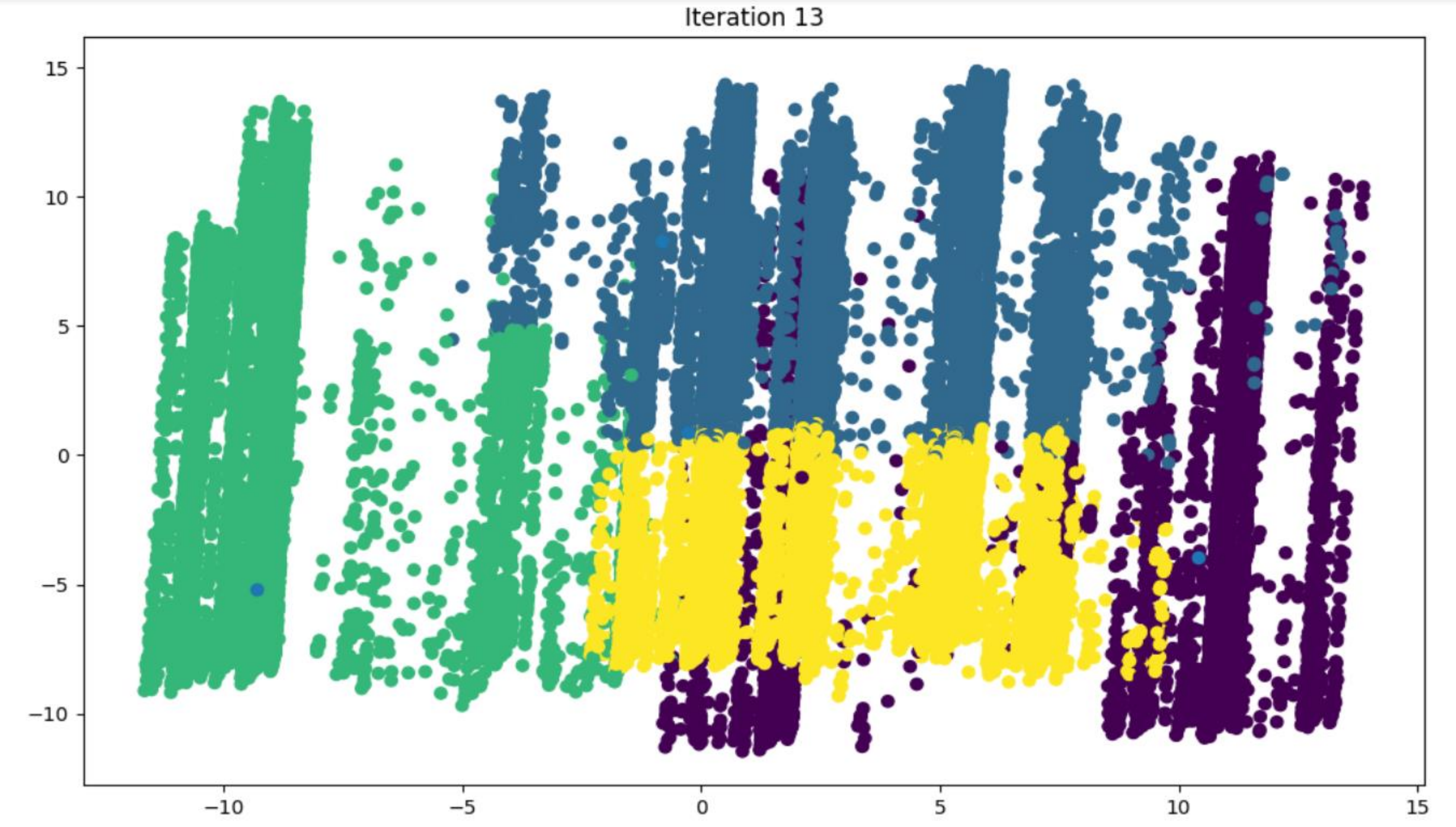


K-MEANS CLUSTERING

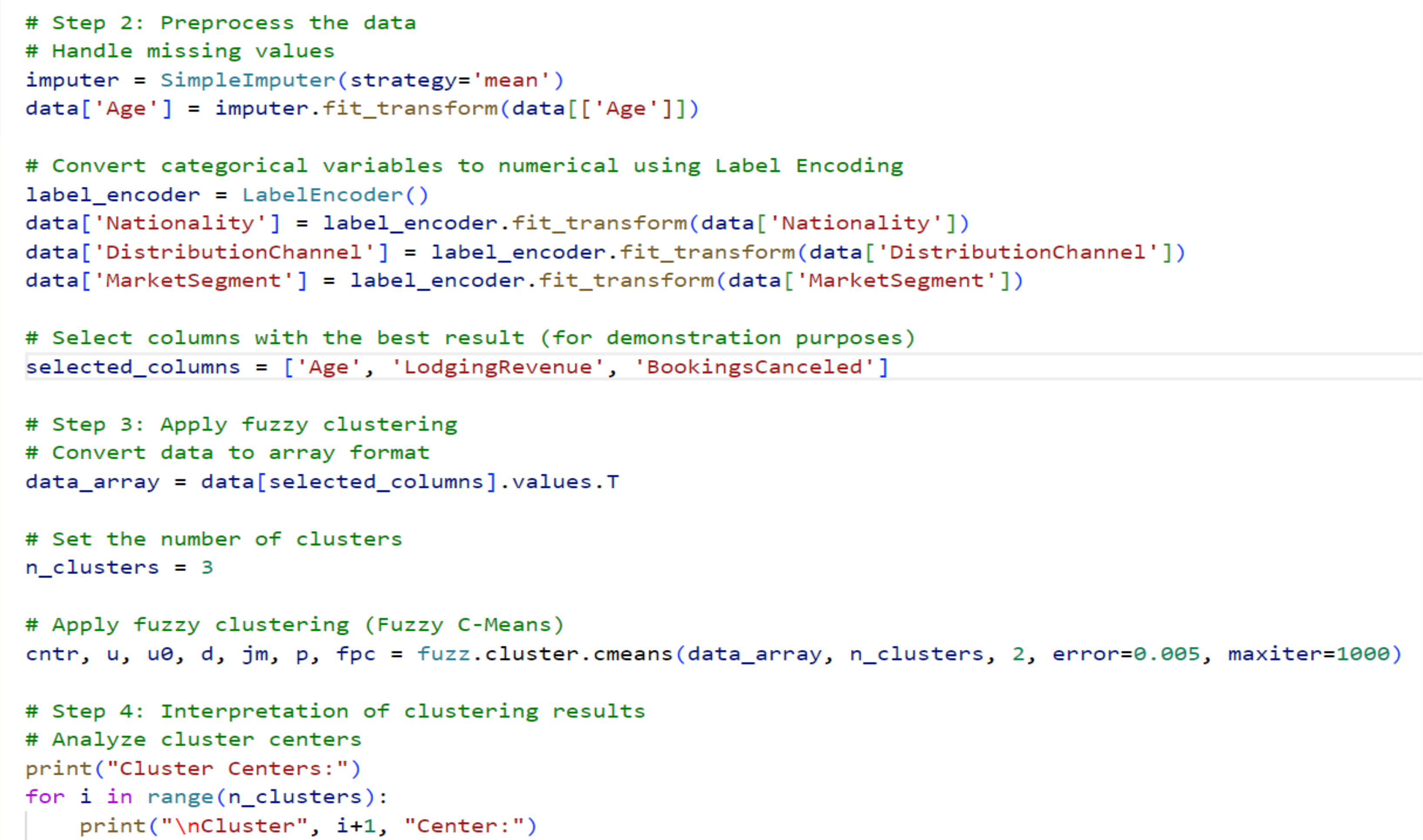


```
# Update values
def new_centroids(data, labels):
    return data.groupby(labels).apply(lambda x: np.exp(np.log(x).mean())).T
from IPython.display import clear_output
def plot_clusters(data, labels, centroids, iteration):
    pca = PCA(n_components=2) # Reduce dimensonality for visualiztion
    data_2d = pca.fit_transform(data)
    centroids_2d = pca.fit_transform(centroids.T)
    clear_output(wait = True) # Clear the output and then rewrite the board
    plt.figure(figsize = (12,7))
    plt.title('Iteration {}'.format(iteration))
    plt.scatter(x = data_2d[:, 0], y = data_2d[:, 1], c = labels )
    plt.scatter(x = centroids_2d[:, 0], y = centroids_2d[:, 1] )
    plt.show()

max_iteration = 22
k = 4
centroids = random_centroids(df_scaled, k)
old_centroids = pd.DataFrame()
iteration = 1

while iteration < max_iteration and not centroids.equals(old_centroids):
    old_centroids = centroids

    labels = get_labels(df_scaled, centroids)
    centroids = new_centroids(df_scaled, labels)
    plot_clusters(df_scaled, labels, centroids, iteration)
    iteration += 1
```

VISUALIZE RESULTS

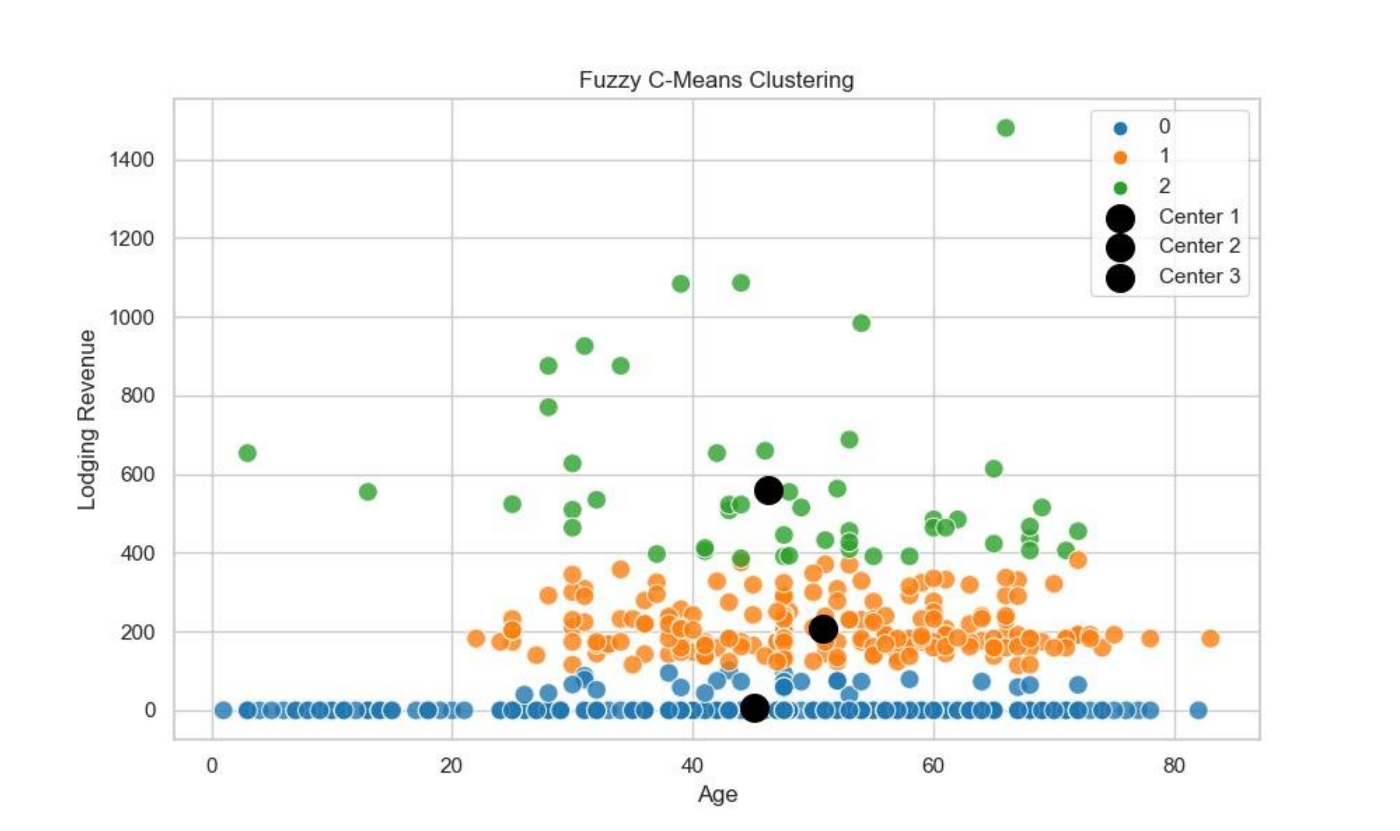


FUZZY CLUSTERING



```
# Step 2: Preprocess the data
# Handle missing values
imputer = SimpleImputer(strategy='mean')
data['Age'] = imputer.fit_transform(data[['Age']])

# Convert categorical variables to numerical using Label Encoding
label_encoder = LabelEncoder()
data['Nationality'] = label_encoder.fit_transform(data['Nationality'])
data['DistributionChannel'] = label_encoder.fit_transform(data['DistributionChannel'])
data['MarketSegment'] = label_encoder.fit_transform(data['MarketSegment'])

# Select columns with the best result (for demonstration purposes)
selected_columns = ['Age', 'LodgingRevenue', 'BookingsCanceled']

# Step 3: Apply fuzzy clustering
# Convert data to array format
data_array = data[selected_columns].values.T

# Set the number of clusters
n_clusters = 3

# Apply fuzzy clustering (Fuzzy C-Means)
cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(data_array, n_clusters, 2, error=0.005, maxiter=1000)

# Step 4: Interpretation of clustering results
# Analyze cluster centers
print("Cluster Centers:")
for i in range(n_clusters):
    print("\nCluster", i+1, "Center:")
```

VISUALIZE RESULTS



THANK YOU

Mohamed Mostafa	2205051
Abd al-Rahman Mohamed	2205078
Ahmed Bassuny	2205180
Abd al-Rahman Mostafa	2205206
Mahmoud amir	2205148