L-Università ta' Malta
Faculty of Information & Communication Technology
Department of Artificial Intelligence

ARI5012:

Data Analysis Techniques

Individual Project

Nathan Portelli
M.Sc. in AI

Task 1: How many questions exist in the survey?

```python
# List of columns, each of which represent the options for each question
columns = data.columns

# Unique question prefixes
unique_questions = set([col.split('_')[0] for col in columns])
print("Unique Questions:", unique_questions)
# Total number of unique questions
print("Number of questions in the survey:", len(unique_questions))
```

```
Unique Questions: {'q6', 'q2', 'q8', 'q9', 'q4', 'q10', 'q3', 'q5', 'q7', 'q1'}
Number of questions in the survey: 10
```

Different Questions

Question No. Response No.

q1_1	q2_1	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	q3_9	q3_10	q7_1	q7_2	q7_3	q7_4	q7_5	q7_6	q7_7	q7_8	q7_9	q7_10
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	1	1

Task 1: How many respondents are in the survey?

```python
# Number of rows (excluding the header)
num_respondents = len(data)

print("Number of respondents in the survey:", num_respondents
```

```
Number of respondents in the survey: 2288
```

Different Questions

Question No.

Response No.

Unique Respondents

q1_1	q2_1	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	q3_9	q3_10	q7_1	q7_2	q7_3	q7_4	q7_5	q7_6	q7_7	q7_8	q7_9	q7_10
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	1	1

Task 1: What are the different question types based on their selection options?

```python
question_types = {}  # Question types for all questions
selection_types = {}  # Determine single/multiple selection type for each question

# Checking if single or multiple options
for col in data.columns:
    question_prefix = col.split('_')[0]
    # If prefix is not in dictionary, assume a single-option question
    if question_prefix not in selection_types:
        selection_types[question_prefix] = 'Single'
    else:
        selection_types[question_prefix] = 'Multiple'
```

```python
# Checking if single or multiple responses
for col in data.columns:
    question_prefix = col.split('_')[0]
    if question_prefix not in question_types:
        question_types[question_prefix] = set()

    # Determine if single or multiple-response questions
    question_columns = [col for col in data.columns if col.startswith(question_prefix)]

    responses_sum = data[question_columns].sum(axis=1)
    responses_max = responses_sum.max()
    if responses_max == 1:
        question_types[question_prefix].add("Single")
    else:
        question_types[question_prefix].add("Multiple")
```

```
Question Types:
q1: Single options
q2: Single options
q3: Multiple options & Single responses
q4: Multiple options & Single responses
q5: Multiple options & Single responses
q6: Multiple options & Multiple responses
q7: Multiple options & Multiple responses
q8: Multiple options & Multiple responses
q9: Multiple options & Multiple responses
q10: Multiple options & Multiple responses
```

Single Response

Single Options

Multiple Options

Multiple Responses

Unique Respondents

q1_1	q2_1	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	q3_9	q3_10	q7_1	q7_2	q7_3	q7_4	q7_5	q7_6	q7_7	q7_8	q7_9	q7_10
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	1	1

Task 2: Which performance measures would you use?

Binary Classification Problems	*Questions 1 & 2*	Accuracy, Precision, Recall, F1-Score, AUC-ROC
Multi-Class Classification Problems	*Questions 3, 4 & 5*	Accuracy, Precision, Recall, F1-Score (Micro-Average), Balanced Accuracy
Multi-Label Classification Problems	*Questions 6 – 10*	Hamming Loss, Precision at k, Recall at k, F1-score (Micro-Average and Macro-Average), Subset Accuracy

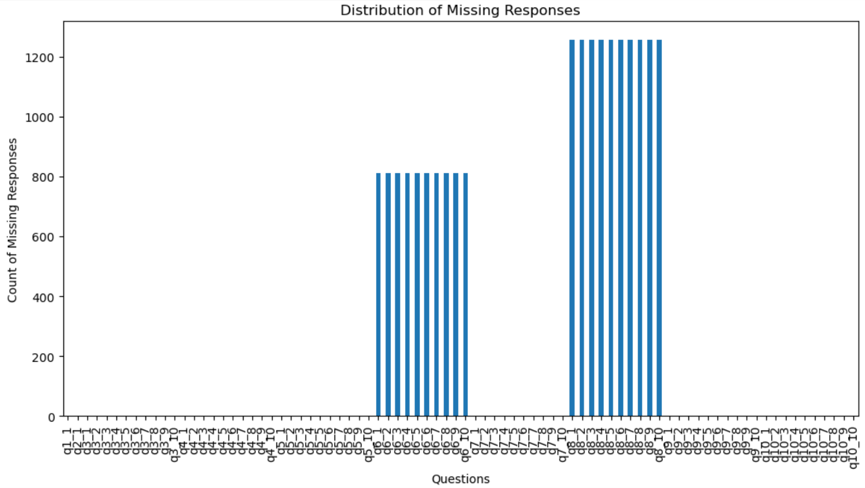
q1_1	q2_1	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	q3_9	q3_10	q7_1	q7_2	q7_3	q7_4	q7_5	q7_6	q7_7	q7_8	q7_9	q7_10
1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
1	1	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	1	1

Task 3: Implement a suitable algorithm for this task.

Missing Values

```
# Amount of missing values per column in explanatory, grouped by question
missing_explanatory_grouped = explanatory.filter(like='q').isnull().sum().groupby(lambda x: x.split('_')[0]).sum()

# Amount of missing values per column in response, grouped by question
missing_response_grouped = response.filter(like='q').isnull().sum().groupby(lambda x: x.split('_')[0]).sum()

print("Missing Values in Explanatory Data (Grouped by Question):")
print(missing_explanatory_grouped)

print("\nMissing Values in Response Data (Grouped by Question):")
print(missing_response_grouped)
```

```
Missing Values in Explanatory Data (Grouped by Question):
q1          0
q10         0
q2          0
q3          0
q4          0
q6       8110
q7          0
q8      12550
q9          0
dtype: int64

Missing Values in Response Data (Grouped by Question):
q5    0
dtype: int64
```



Task 3: Implement a suitable algorithm for this task.

Deletion

List-Wise Deletion

Column-Wise Deletion

Missing Responses are

Missing Completely At Random (MCAR)

Missing At Random (MAR)

Missing Not At Random (MNAR)

Imputation

Simple Imputation

Mean Imputation

Mode Imputation

K-Nearest Neighbours (KNN)

Regression Imputation

Hot-Deck Imputation

Response Distribution when q6 is not picked:

	q1_1	q2_1	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	...	q10_1	\
0.0	0.0	804	1215	1439	1308	1339	1388	1257	1293	1384	...	1300	
1.0	1477.0	673	262	38	169	138	89	220	184	93	...	177	

	q10_2	q10_3	q10_4	q10_5	q10_6	q10_7	q10_8	q10_9	q10_10
0.0	1221	955	1174	826	1066	809	1073	646	1314
1.0	256	522	303	651	411	668	404	831	163

Response Distribution when q8 is not picked:

	q1_1	q2_1	q3_1	q3_2	q3_3	q3_4	q3_5	q3_6	q3_7	q3_8	...	q10_1	\
0.0	360	0.0	792	963	975	1010	928	914	848	943	...	787	
1.0	673	1033.0	241	70	58	23	105	119	185	90	...	246	

	q10_2	q10_3	q10_4	q10_5	q10_6	q10_7	q10_8	q10_9	q10_10
0.0	958	773	564	390	513	815	966	546	941
1.0	75	260	469	643	520	218	67	487	92

Task 3: Implement and evaluate a suitable algorithm for this task.

Preliminary Steps

Training & Testing Split (70/30)

Hyperparameter Grids

Implementation

Decision Tree

- Criterion: Entropy
- Max. Depth: 30
- Min. Sample Leaf: 1
- Min. Sample Split: 2

Neural Network

- Activation: ReLU
- Alpha: 0.01
- Hidden Layer Sizes: (50, 50)
- Learning Rate: Constant
- Solver: adam

Results

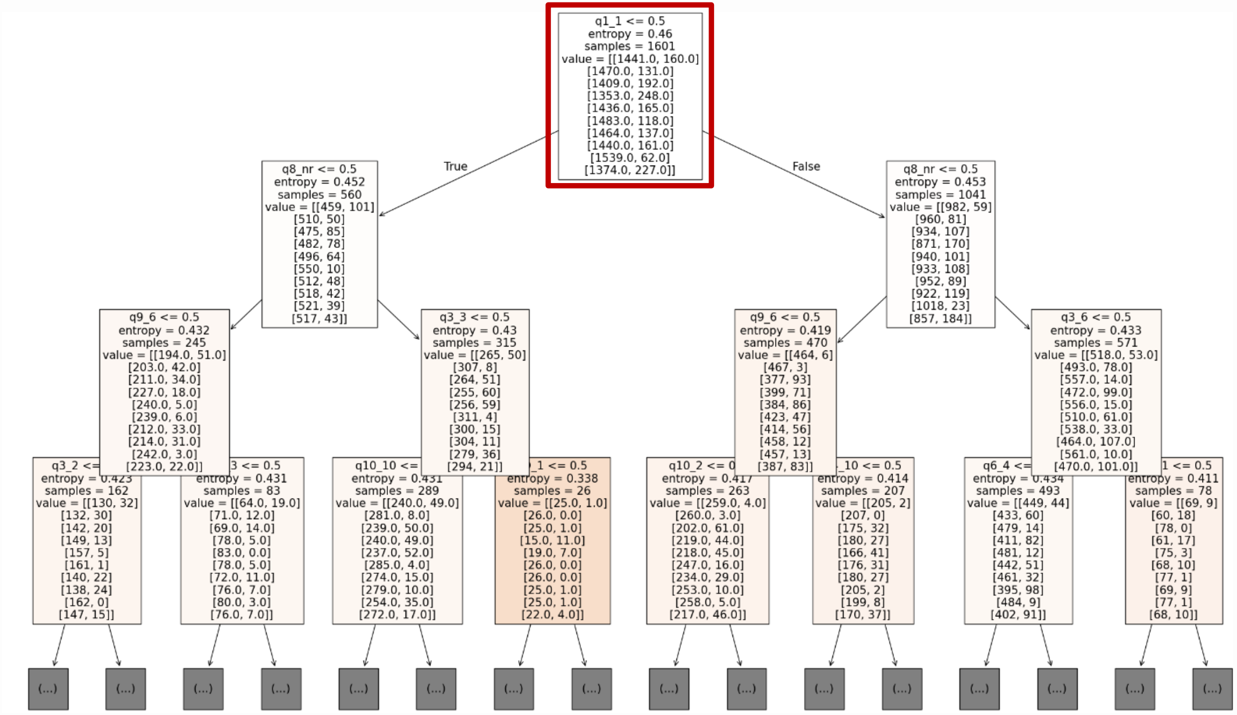
Decision Tree

- Accuracy: 0.14119
- Precision: 0.15267
- Recall: 0.14119
- F1-Score: 0.14464

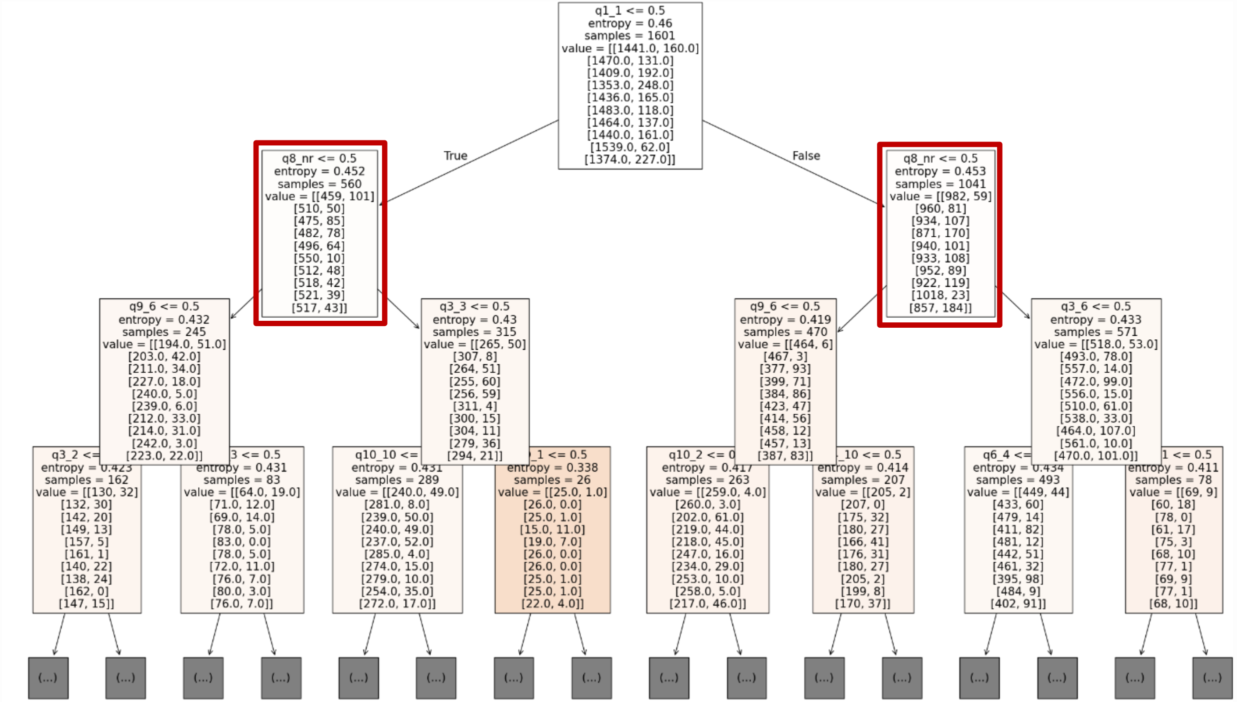
Neural Network

- Accuracy: 0.04949
- Precision: 0.12822
- Recall: 0.06550
- F1-Score: 0.08273

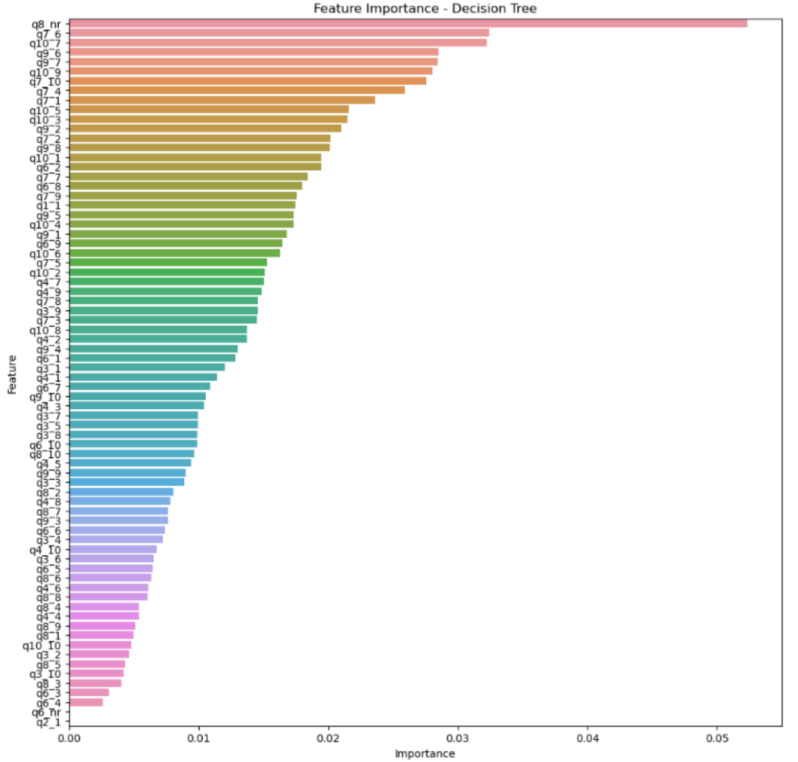
Task 3: Interpret the trained model and results.

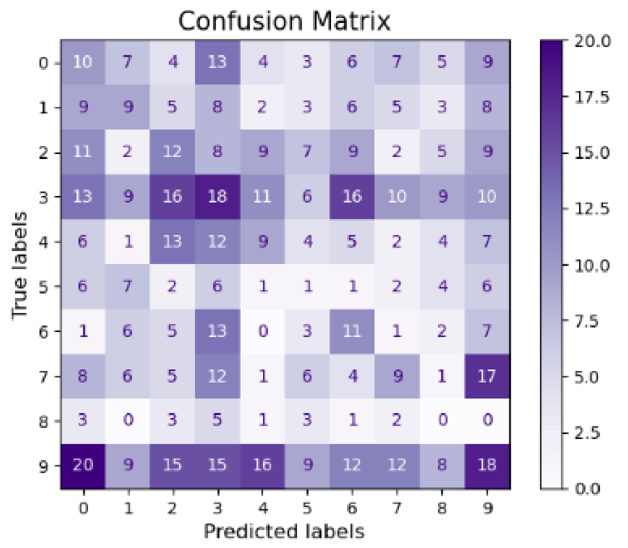


Task 3: Interpret the trained model and results.



Task 3: Interpret the trained model and results.





Task 3: Given that Question 5 responses are semantically ordered.

```
response_modified = response.copy()

for i in range(1, 11):
    col_name = f'q5_{i}'
    response_modified[col_name] = response_modified[col_name].apply(lambda x: i if x == 1 else x)

# Sum of all columns q5_1 ... q5_10
response_modified['q5'] = response_modified.sum(axis=1)
# Only keeping the new q5 column, deleting q5_1 ... q5_10
response_modified = response_modified[['q5']]

response_modified.head()
```

	q5
0	5.0
1	8.0
2	7.0
3	1.0
4	1.0

Implementation

Decision Tree
- Criterion: Gini
- Max. Depth: 10
- Min. Sample Leaf: 4
- Min. Sample Split: 5

Results

Ordinal Classification Problem
- MAE: 3.38427
- MSE: 18.29549
- Weighted Kappa: 0.01708

Task 4: Create groups of similar respondents by implementing suitable algorithms.

K-Means

- Simple and efficient
- Well-suited for quantitative variables
- Partitions data into K-clusters by minimizing within-cluster variance
- Simple to implement
- Suitable for large datasets
- Requires the number of clusters to be specified in advance
- Sensitive to initial cluster centroids

K-Medoids

- Works well with categorical variables
- More robust against outliers
- Suitable for mixed data types
- More computationally intensive
- Requires the number of clusters to be specified in advance
- Sensitive to the choice of distance metric

Task 4: Create groups of similar respondents by implementing suitable algorithms.

K-Means

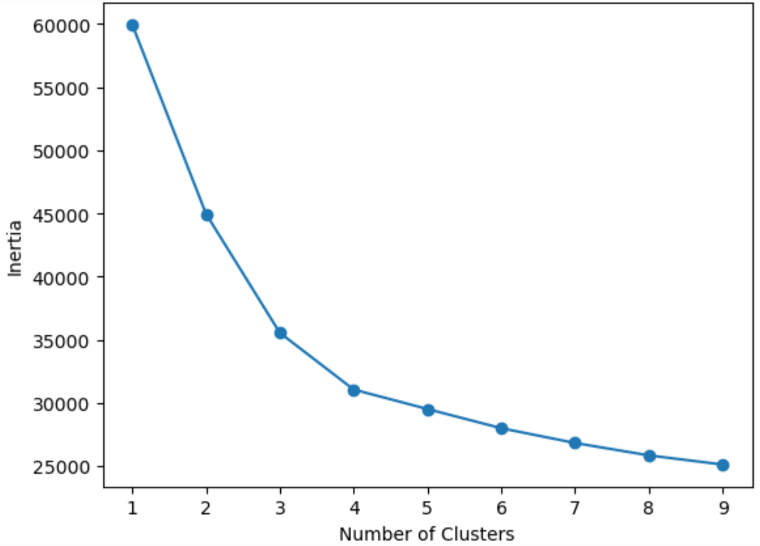
```
def elbow_method(data):
    means = []
    inertias = []

    for k in range(1, 10):
        kmeans = KMeans(n_clusters=k, n_init=10)
        kmeans.fit(data)
        means.append(k)
        inertias.append(kmeans.inertia_)

    plt.plot(means, inertias, marker='o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.show()
```

K-Medoids

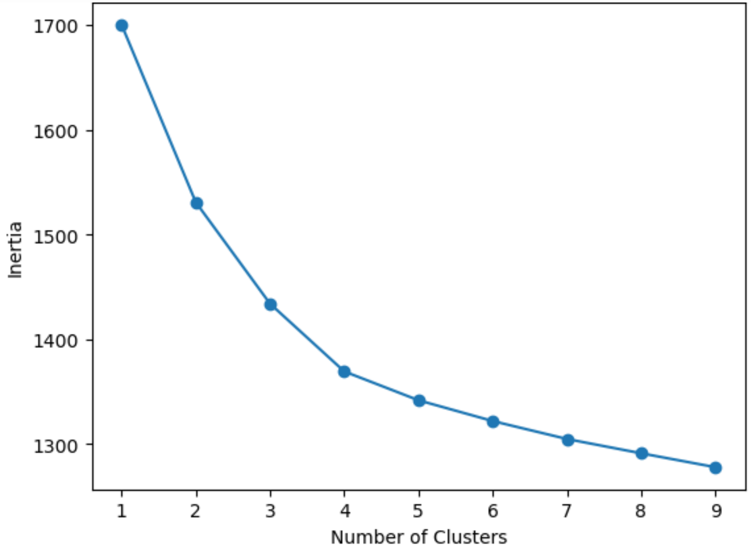
```
def elbow_method(data):
    means = []
    inertias = []

    for k in range(1, 10):
        kmedoids = KMedoids(n_clusters=k, metric='jaccard', method='pam', max_iter=500, random_state=1)
        kmedoids.fit(data)
        means.append(k)
        inertias.append(kmedoids.inertia_)

    plt.plot(means, inertias, marker='o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.show()
```

Task 4: Create groups of similar respondents by implementing suitable algorithms.

K-Means

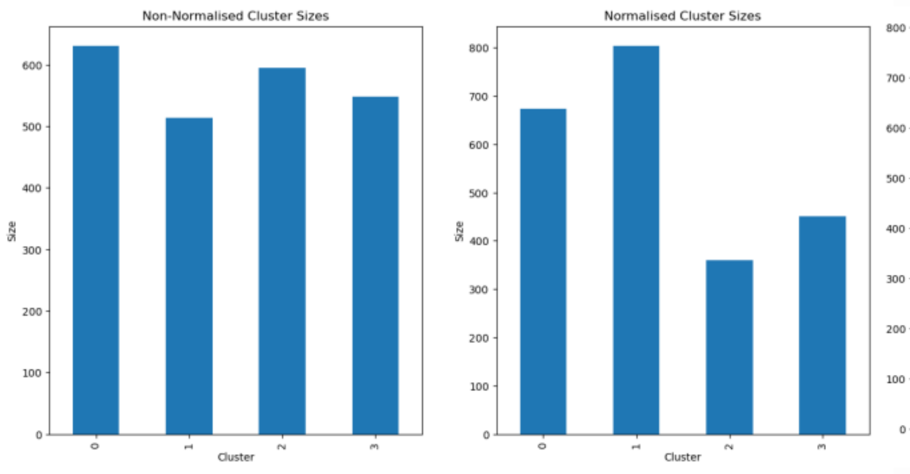


K-Medoids

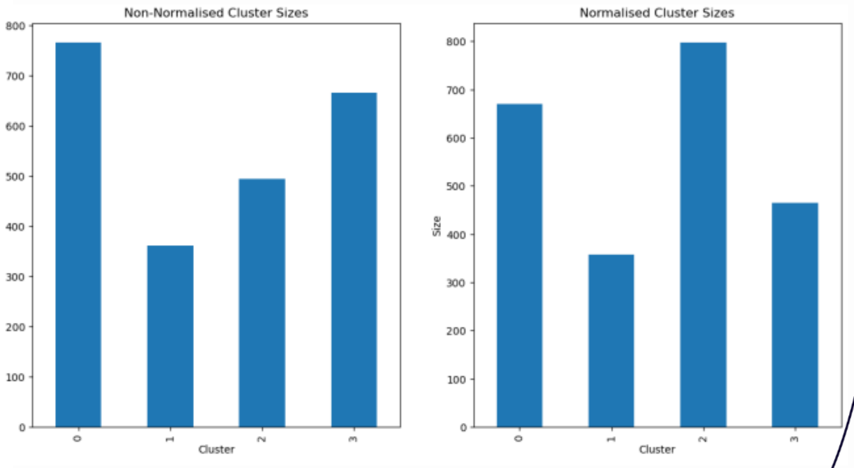


Task 4: Create groups of similar respondents by implementing suitable algorithms.

K-Means



K-Medoids



Task 4: Evaluate whether the clusters you created are good enough.

K-Means

Silhouette Coefficient

– Normalised:	0.19586
– Non-Normalised:	0.13772

Davies-Boudin Index

– Normalised:	1.49233
– Non-Normalised:	2.13013

K-Medoids

Silhouette Coefficient

– Normalised:	0.11805
– Non-Normalised:	0.11298

Davies-Boudin Index

– Normalised:	2.35494
– Non-Normalised:	2.44112

Task 4: Compare and select one of them

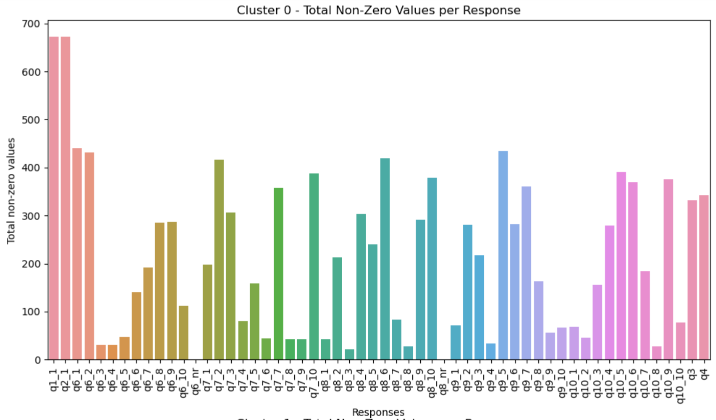
K-Means

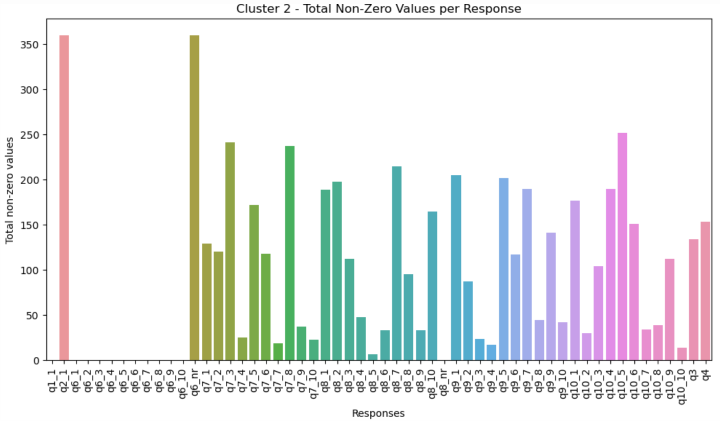
- Typically applied to numerical data types.
- Sensitive to outliers.
- Less computationally expensive.
- Preferred for large datasets.
- Centroids may not accurately represent clusters.
- Less robust to outliers.

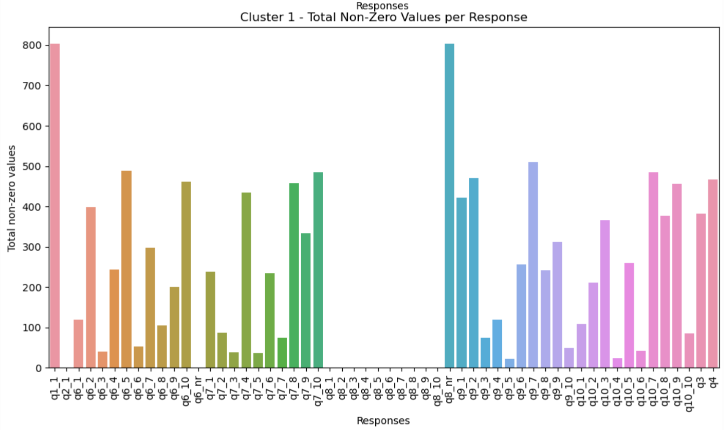
K-Medoids

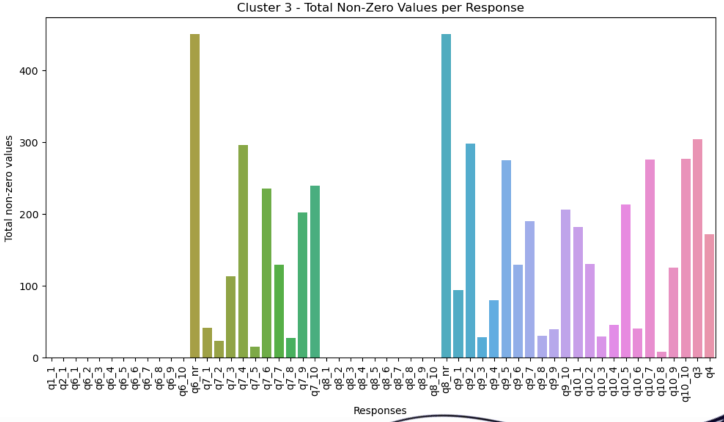
- Can handle categorical data & mixed data types.
- Less sensitive to outliers.
- More computationally expensive
- Easier to interpret
- More robust to outliers.

Task 4: Describe the groups you identified.

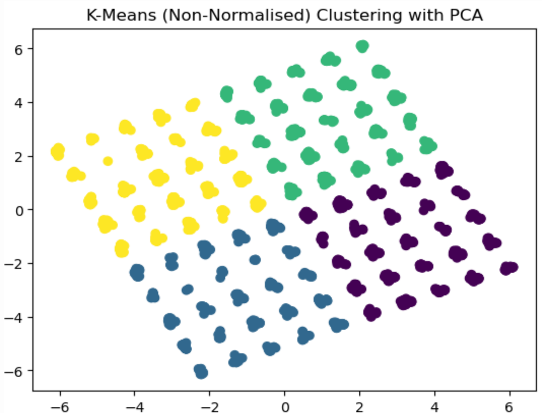


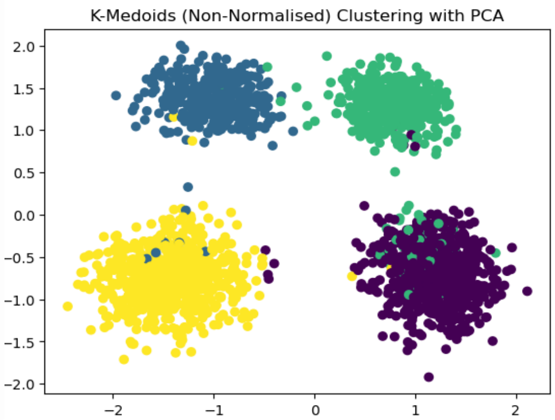




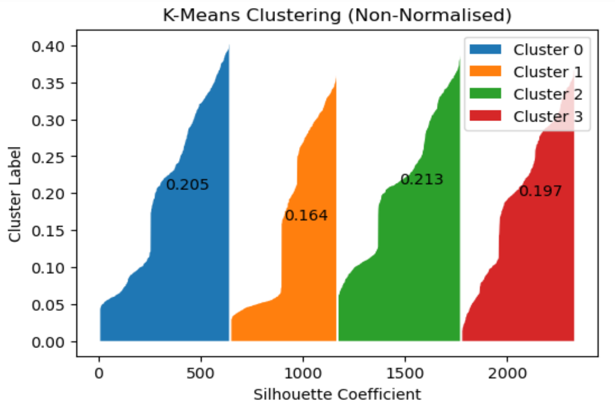


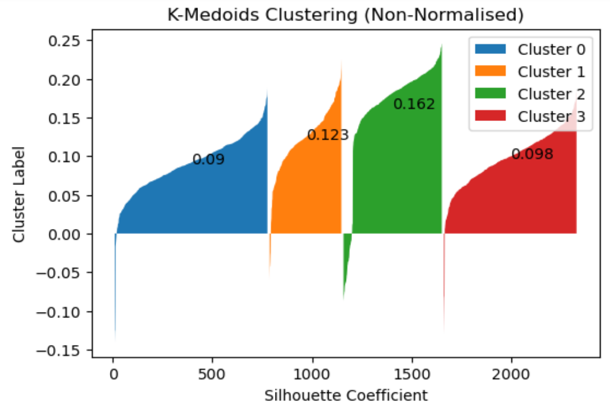
Task 4: Create visualisations for the clustering results.

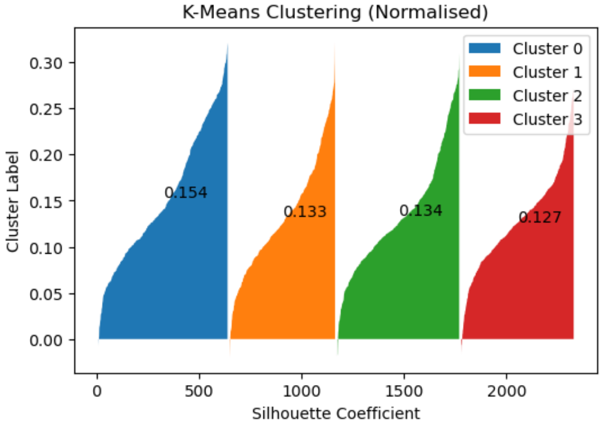


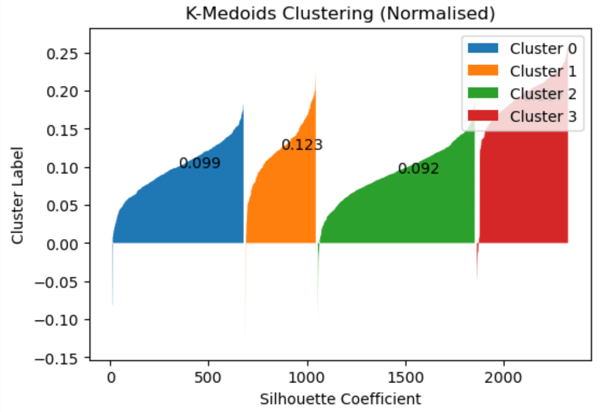


Task 4: Create visualisations for the clustering results.









Task 4: Assigning a new respondent to existing group.

```
# The new respondent's data will include the formatting done in Task 3.1.1 and 4.1
new_respondent = [1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0
```

```
# Fit the scaler on all the features
scaler.fit(explanatory_kmeans_normalised_nolabel)

new_respondent_normalised = scaler.transform([new_respondent])  # Normalising new response

# Finding the distances to each centroid
distances = np.linalg.norm(kmeans_normalised.cluster_centers_ - new_respondent_normalised, axis=1)
closest_cluster = np.argmin(distances)  # Getting the closest cluster

print(f'The New Respondent is assigned to Cluster {closest_cluster}')
```

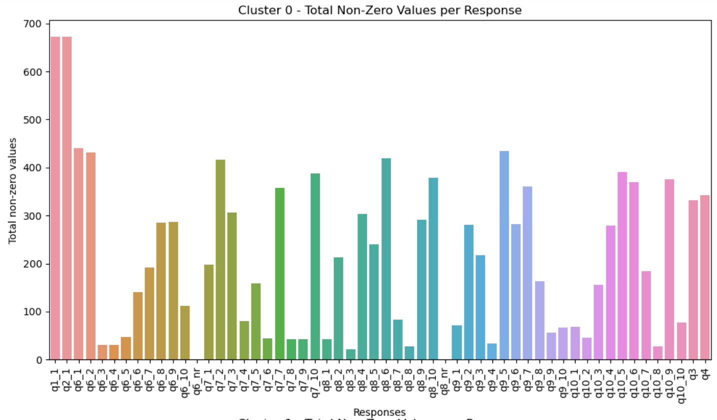
```
The New Respondent is assigned to Cluster 1
```

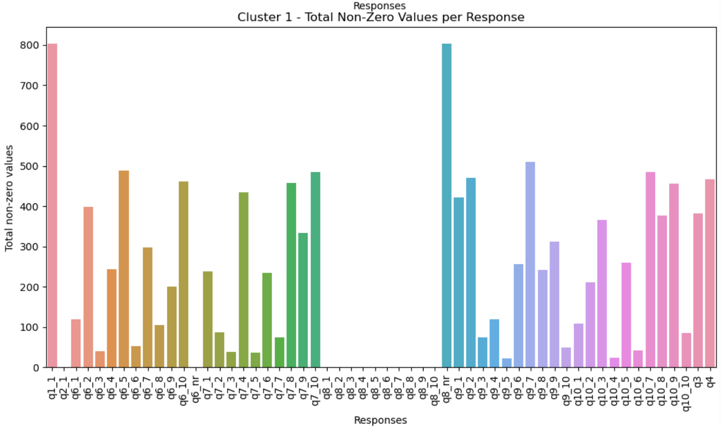
```
second_respondent = [1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0
```

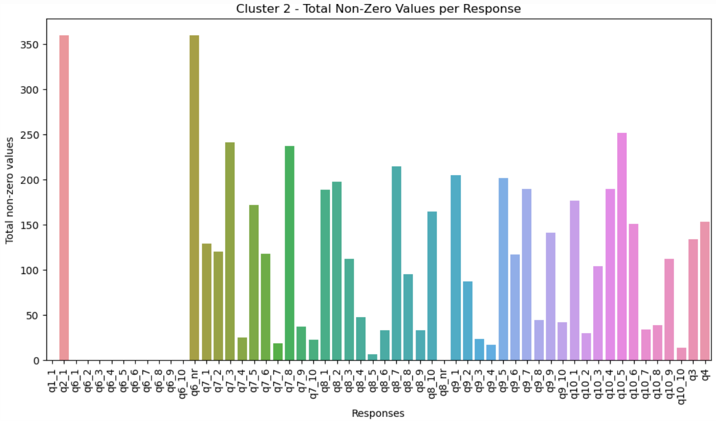
```
# Fit the scaler on all the features
scaler.fit(explanatory_kmeans_normalised_nolabel)

new_respondent_normalised = scaler.transform([second_respondent])  # Normalising second response

# Finding the distances to each centroid
distances = np.linalg.norm(kmeans_normalised.cluster_centers_ - new_respondent_normalised, axis=1)
closest_cluster = np.argmin(distances)  # Getting the closest cluster

print(f'The Second Respondent is assigned to Cluster {closest_cluster}')
```

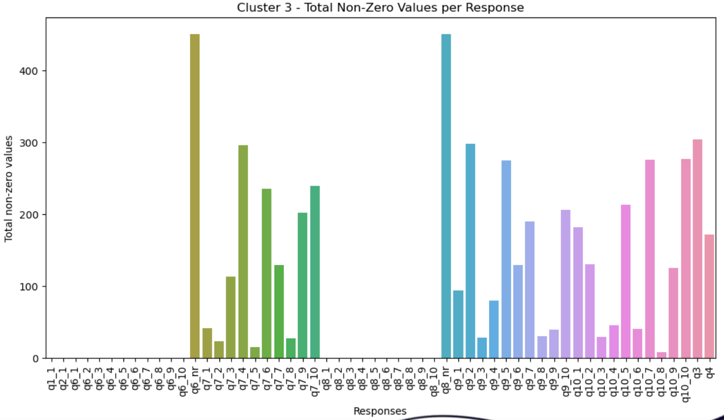
```
The Second Respondent is assigned to Cluster 0
```

Task 4: Describe the groups you identified.









Task 5: Explain whether the clustering information could be used to build more accurate models for Task 3.

Feature Engineering

- Use cluster assignments as additional features.
- Serve as proxies for underlying patterns within the dataset that individual responses might not capture.

Ensemble Methods

- Train separate predictive models for each of the four clusters identified in Task 4.
- Potentially improve prediction accuracy for each subgroup.

Cluster-Specific Sampling & Analysis

- Use statistics, visualisation, or model interpretation to identify unique patterns between survey responses and the target variable within each cluster.
- Use clustering as stratification criteria to ensure adequate representation of each cluster, preventing data imbalance and bias.

L-Università ta' Malta
Faculty of Information & Communication Technology
Department of Artificial Intelligence

ARI5012:

Data Analysis Techniques

Individual Project

Nathan Portelli
M.Sc. in AI