DSM100 Coursework 1: Development and Implementation of a Bayesian Network

Submitted for partial fulfilment for the Artificial Intelligence course.

By

Hendrik Matthys van Rooyen

University of London

November - December 2023

CONTENTS

CHAPTER 1 INTRODUCTION

This project aims to explore the design and implementation of Decision Support Systems using a Bayesian Network. For this project, the scenario of Obesity or Cardiovascular Disease diagnosis support is used as background.

1.1 DSS

Decision Support Systems utilize artificial intelligence to enhance decision-making in various sectors like healthcare and logistics (Poniecki-Klotz, 2023) (‘AI Decision Support Systems: A Guide to IDSS | Symanto’, 2022). By analysing data from multiple sources, DSS provide actionable insights, significantly improving organizational operations and problem-solving. In healthcare, AI-driven decision-support systems can detect overlooked medical information, allowing for interactive and clarified decision-making processes. (Bajgain *et al.*, 2023)

1.2 BAYESIAN NETWORKS

Bayesian Networks (BNs) in DSS, crucial for reasoning under uncertainty, use Bayes' theorem to update decisions with new data. They are particularly effective in uncertain environments like healthcare and environmental management. (Bajgain *et al.*, 2023)

CHAPTER 2 SPECIFICATION OF THE DSS

The DSS of concern in this project is aimed at assisting the diagnosis of Obesity or Cardiovascular Disease (CVD). It uses a Bayesian Network, which is a type of probabilistic graphical models. This allows the model to efficiently calculate probabilities. The system is built from the data provided one the Kaggle dataset: “Obesity or CVD risk (Classify/Regressor/Cluster)”. (*Obesity or CVD risk (Classify/Regressor/Cluster)*, no date)

As mentioned, the core of the DSS is a Bayesian network, which is trained to identify patterns and correlations in the aspects of the medical data, but those indicative of CVD is of specific interest.

The data, and the output of the model indicates a range of states:

- Insufficient Weight
- Normal Weight
- Overweight Level I – II
- Obesity Type I – III

These states would be calculated based on aspects such as, Gender, Age, Height and Weight, as well as some others relating to various health aspects.

2.1 DSS TASKS/OBJECTIVES

- Analize input health data of patients to determine the risk of obesity.
- Provide probability scores of determined status, to help medical professionals evaluate overall risk, or progression towards Obesity.
- Conceptually the DSS could be used to make recommendations or alert a patient of risk.

Through these the DSS can increase the accuracy and decrease the complexity of CVD diagnosis and serve as an early detection measure.

2.2 CONSTRAINTS

- The quality of the DSS is dependent on the quality and amount of input data, as well as the availability of training data.
- The privacy of a patient’s data in a model such as this, in a production environment, is of great importance.
- The DSS should be used with the support of a medical professional.
- The data and model may need to be updated from time to time.

(Heyen and Salloch, 2021)

2.3 STAKEHOLDERS

- Healthcare providers and medical institutions.
- Patients and their families.
- Health insurance companies.
- Medical researchers.

2.4 USERS

- Doctors and medical professionals
- Medical researchers

2.5 INTEGRATIONS

- It may be useful to integrate the DSS with existing digital health records.
- Medical diagnostic tools

2.6 DATA REQUIREMENTS

- Comprehensive patient health records, including lifestyle, genetic, and clinical data. Though for this project the limited Kaggle dataset will suffice.
- Regular updates to the data and model, as new information and patient data comes available.

2.7 KNOWLEDGE REQUIREMENTS

- For the sake of developing the model, some expertise in Bayesian networks, and data science is required.
- An understanding of the medical terms and factors would assist in the interpretation of the data.
- Knowledge and awareness of the ethical concerns and legal standards is required in the handling of the medical data.

CHAPTER 3 CONCEPTUAL DESIGN OF THE DSS

3.1 DSS COMPONENTS

- **Data Collection/Ingestion:** Manage the collection or retrieval of the data from the provider.
- **Preprocessing:** Clean the retrieved data, balance the training data, bin the data as required, and prepare for training.
- **Bayesian Network:**
    - Estimate the Directed Acyclic Graph (e.g. Hill Climb Search) or use a fixed graph from supplied node relationships.
    - Estimate the Bayesian Network model.
    - Evaluate the accuracy.
- **User Interface:** An interaction-point for the user, for the project, it is limited to the Python scripts, but could realistically include web based or application-based interfaces. The interfaces would allow you to supply the available medical information and receive an estimate of your CVD status.

Conceptual:

- **Integration Interface:** Integrations with existing systems to pull the data from.
- **Data Storage:** A hosted relational database or other storage space for data. For the project it is a CSV file.
- **Reporting/Allert System:** Integration with medical diagnosis software, health apps, or emailing systems.

3.1.1 INPUT DATA

- Patient demographics (Gender, Age, Height).
- Lifestyle data.
    - Consumption of high caloric food (FAVC)
    - Consumption of vegetables (FCVC)
    - Number of main meals (NCP)
    - Consumption of food between meals (CAEC)
    - Consumption of water daily (CH20)
    - Consumption of alcohol (CALC)
    - Smoker or not
    - Physical activity frequency
    - Time using technology devices.
    - Transportation used.
- Genetic information.
    - Family member suffered or suffers from overweight.
- Clinical measurements (Weight).

(*Obesity or CVD risk (Classify/Regressor/Cluster)*, 2023)

3.2 KNOWLEDGE SOURCES

- Medical data and literature on obesity and CVD
- Statistical modeling knowledge
- Historical or sample data for training the Bayesian Network

CHAPTER 4 BAYESIAN NETWORK REPRESENTATION

4.1 BACKGROUND KNOWLEDGE

As previously mentioned, the Bayesian network for this DSS is built upon medical data and statistical knowledge concerning obesity and CVD. From the data some key factors are provided:

- Patient demographics
- Lifestyle data
- Genetic information
- Clinical measurements

Alongside these a diagnosis is provided on the range of:

- Insufficient Weight
- Normal Weight
- Overweight Level I – II
- Obesity Type I – III

(*Obesity or CVD risk (Classify/Regressor/Cluster)*, 2023)

The links between these can be determined through expert medical knowledge, or in the case of the project, be estimated based on the available dataset. These links represent the relationships between different factors, which may not be directly related, but directly or through their relationship with other nodes be linked to the diagnosis.

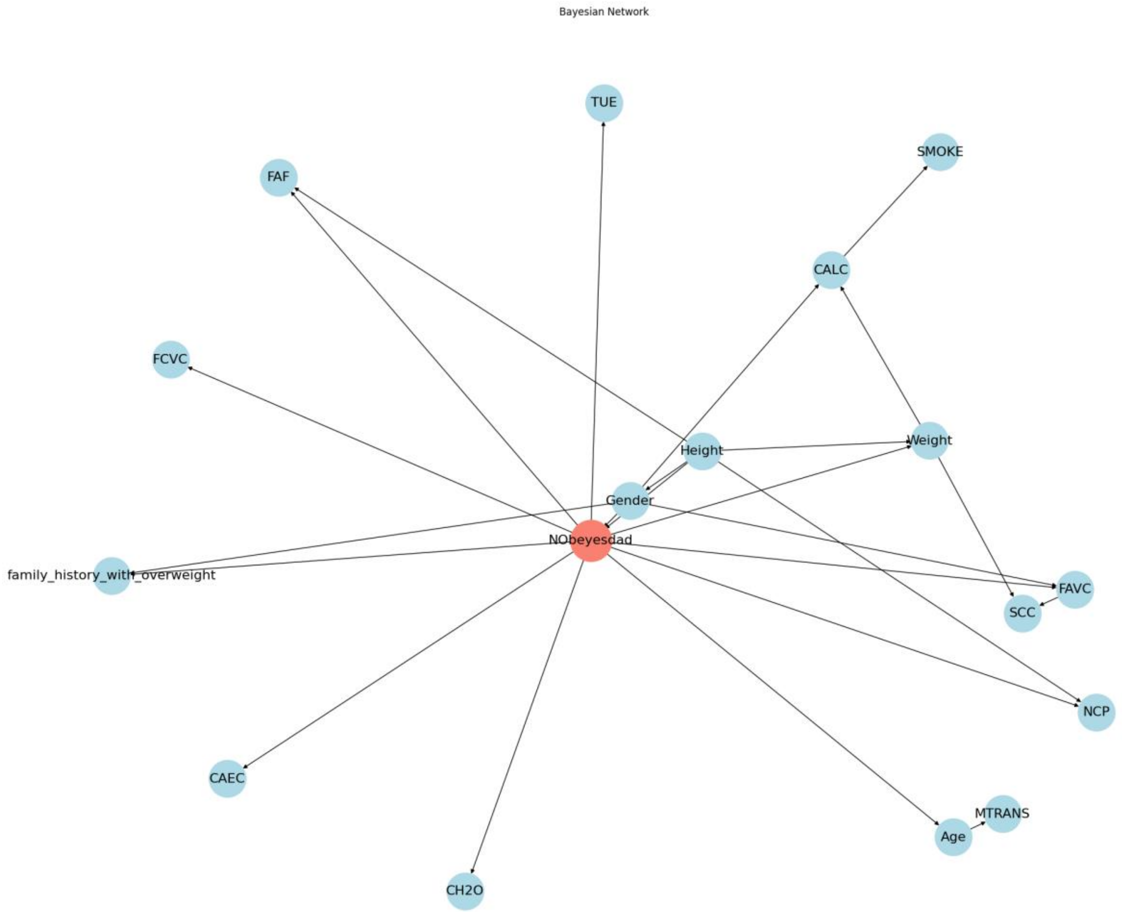
4.2 ESTIMATION OF ASSOCIATED PROBABILITIES

The probabilities of each node-relationship/edge would be estimated based on the historic or sample-data, however, the probabilities could also be determined through expert knowledge. The relationships can also be updated with time as new data or knowledge becomes available.

4.3 CONSTRUCTION OF THE BAYESIAN NETWORK

As the network structure can vary between estimations, it may be difficult to determine the structure beforehand but, as is the nature of the Directed Acyclic Graph, each property in the dataset would be represented by a node in the network, with the probabilities of each node quantifying the likelihood of each state, given the state of its parents.

A graphical representation of the graph can be seen below:



CHAPTER 5 QUERYING THE BAYESIAN NETWORK

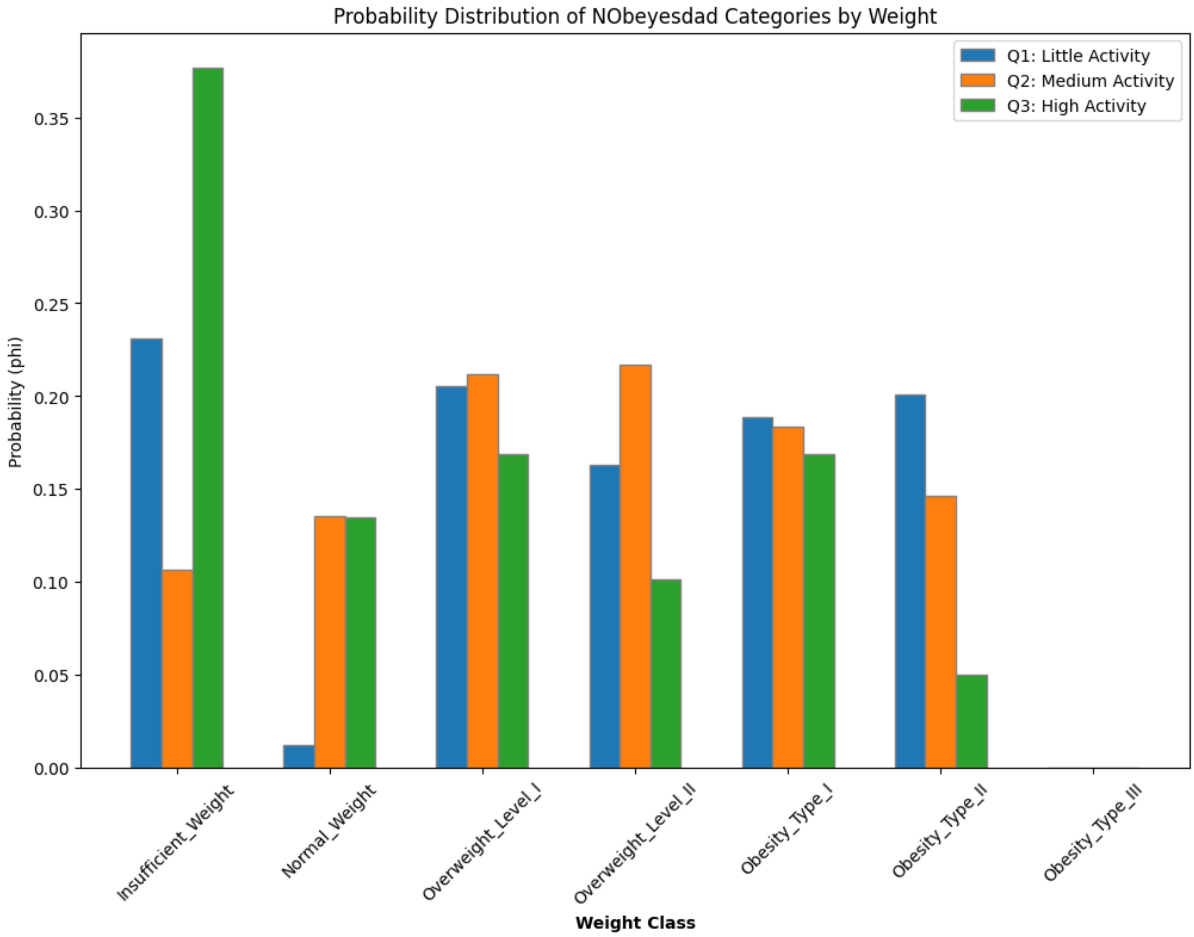
5.1 QUESTION 1

What is the impact of technology use, alongside physical activity on obesity?

For the primary question we will investigate the probability of each weight class given that the person spends a large amount of time on technology devices, with little physical activity.

We will then investigate the difference with an increased physical activity,

And lastly, investigate the effect of maximal physical activity.



From the above we can see that, spending a large amount of time using technology devices (likely sitting at a desk or couch), while doing little physical activity has very little chance (0.0117) to result in normal weight, while causing having a 23% chance of being underweight, and a cumulative 75.7% chance of ranging from Overweight Level I to Obesity Type II.

A moderate increase in physical activity increases your probability of being of Normal Weight to 13.5%, decreases your probability of Obesity Type II, as well as your probability of having Insufficient Weight.

High physical activity can decrease your probability of Obesity Type II to 5% and your cumulative chance to be overweight or obese to 48.9%. It does, however, cause a significant probability of having Insufficient Weight.

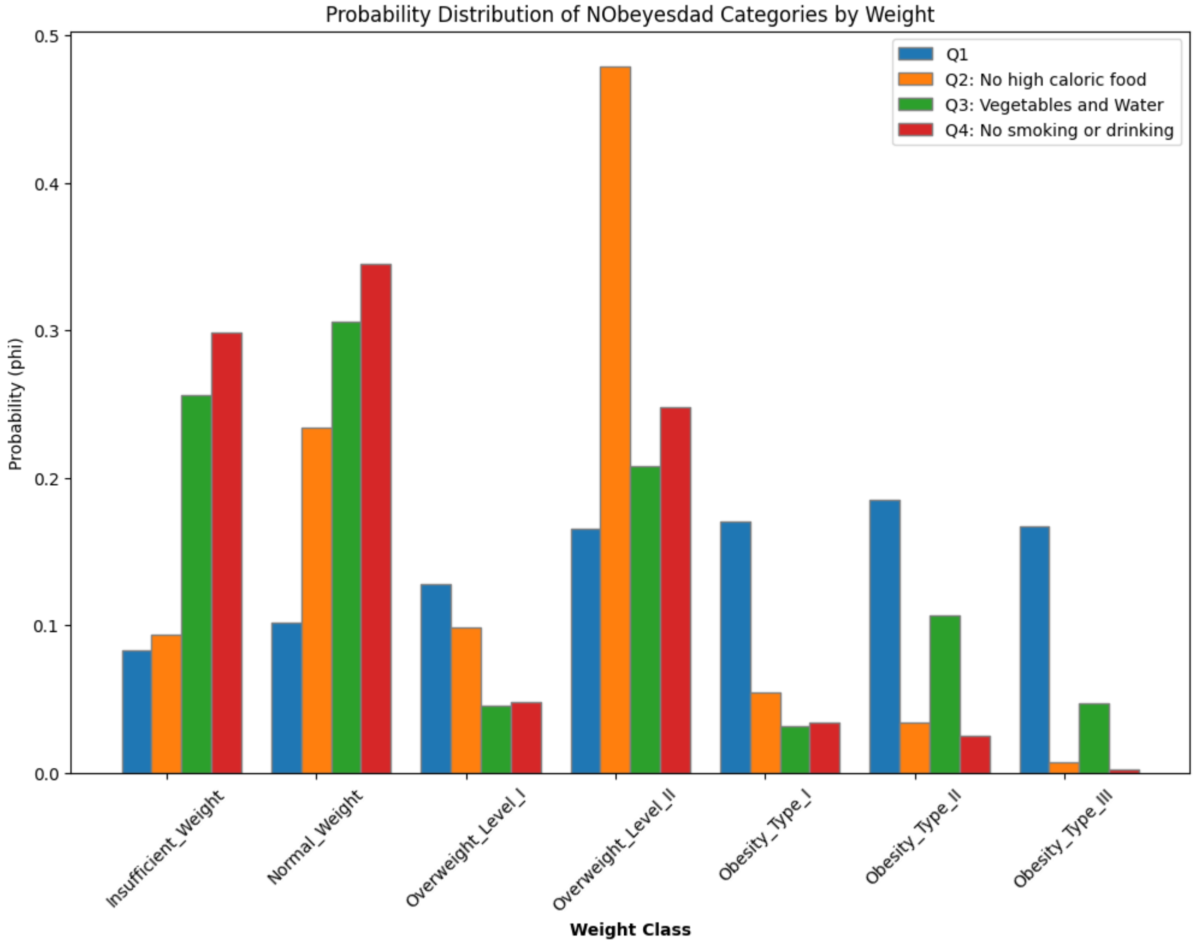
5.2 QUESTION 2

How does a low caloric diet impact CVD risk when you have a genetic predisposition (Family member suffered or suffers from overweight)?

Firstly, the effect of having a Family member who suffered or suffers from overweight has on how likely you are to fall into a given weight class is investigated.

Then it will be investigated how a increasingly healthy diet affects the outcomes:

- Not consuming high caloric food
- Consuming healthy amounts of vegetables and water
- Not smoking or drinking



From the above it is shown that having a Family member who suffered or suffers from being overweight does indicate risk of being overweight or obese, with a person having only a probability of 10.1% of being Normal Weight, and 8.3% having Insufficient Weight.

Resulting in a cumulative 81.6% probability of being on the range of Overweight Level I to Obesity Type III, whit the highest being Obesity Type II at 18.5%.

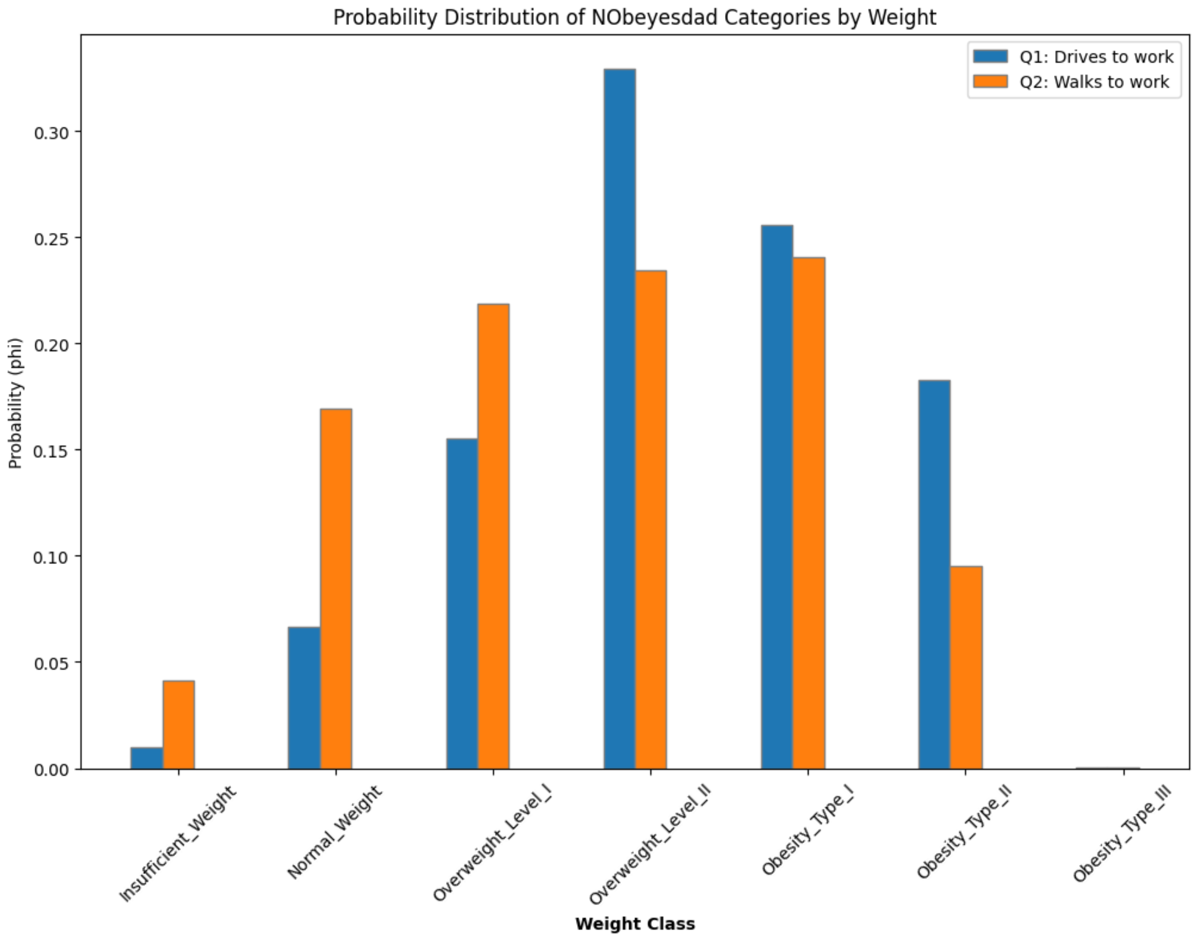
The follow up questions then indicate that you can significantly reduce your probability of Obesity Type I - III through cutting high caloric food, consuming healthy amounts of (Poniecki-Klotz, 2023)Cutting high caloric food alone already increases the probability of being of Normal weight to 23.4%, a significant increase in Overweight level II can be seen, but this can be justified by the cumulative decrease of the probability for Obesity Level I - III from 52.2% to 9.5%.

Implementing the healthy diet fully can decrease the probability of Obesity Type I - III further to 6.1% and increase the probability to be of normal weight or lower to 64.4%

5.3 QUESTION 3

How does smoking and alcohol consumption, alongside mode of transportation correlate to obesity risk?

For this question, the effect of walking to work has, compared to driving, for a person who smokes and drinks alcohol frequently, on their weight classification.



From the above we can evaluate that smoking and drinking but driving to work leans towards being overweight or obese, with a cumulative probability of 92.3.9%, the highest of which is Overweight Level II at 23.5%.

Switching mode of transport to walking increases the probability of being of normal weight form 6.7%, to 16.9%. And decreases the cumulative probability of being overweight or obese to 78.9%.

CHAPTER 6 CONCLUSION

This project successfully demonstrates the potent application of Bayesian Networks within a Decision Support System, specifically tailored for the diagnosis of Obesity or Cardiovascular Disease.

The model's ability to integrate and analyze various health data points, ranging from patient demographics to lifestyle and genetic information has proven effective. The results generated by the Bayesian Network in assessing the risk of Obesity and Cardiovascular Disease, highlight the system's potential as a valuable tool for medical professionals.

The exploration of different scenarios through querying the Bayesian Network offers insightful revelations, such as the effect of technology use, physical activity, diet, and genetic predispositions, on health outcomes. These insights not only validate the system's capability in handling complex queries but also illustrate its practical utility in real-world healthcare settings.

In conclusion, this project exemplifies the practical application of artificial intelligence in healthcare, specifically through the lens of Bayesian Networks in a Decision Support System. The findings and methodologies discussed provide a solid foundation for future research and development in this domain, with a clear path towards enhancing healthcare delivery and patient outcomes. Further work could also include the full integration of a live system, as well as the development of a user interface.

REFERENCES

‘AI Decision Support Systems: A Guide to IDSS | Symanto’ (2022), 13 October. Available at: https://www.symanto.com/blog/artificial-intelligence-decision-support-systems/ (Accessed: 9 December 2023).

Bajgain, B. *et al.* (2023) ‘Determinants of implementing artificial intelligence-based clinical decision support tools in healthcare: a scoping review protocol’, *BMJ Open*, 13(2), p. e068373. Available at: https://doi.org/10.1136/bmjopen-2022-068373.

Heyen, N.B. and Salloch, S. (2021) ‘The ethics of machine learning-based clinical decision support: an analysis through the lens of professionalisation theory’, *BMC Medical Ethics*, 22(1), p. 112. Available at: https://doi.org/10.1186/s12910-021-00679-3.

Obesity or CVD risk (Classify/Regressor/Cluster) (no date). Available at: https://www.kaggle.com/datasets/aravindpcoder/obesity-or-cvd-risk-classifyregressorcluster (Accessed: 9 December 2023).

Poniecki-Klotz, B. (2023) ‘Unleashing the Power of AI in Decision Support Systems’, *Ubuntu AI*, 27 November. Available at: https://medium.com/ubuntu-ai/unleashing-the-power-of-ai-in-decision-support-systems-cb95c7594177 (Accessed: 9 December 2023).

Word Count: 1860

APPENDICES

Coursework 1 - Addendum 1:

Submitted for the partial fulfilment of the DSM100 course

By Hendrik Matthys van Rooyen

230221176

Decision Support System for CVD risk evaluation using Bayesian Network

---

In [ ]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTENC
from sklearn.utils import shuffle

from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split

import networkx as nx
import matplotlib.pyplot as plt

from pgmpy.estimators import HillClimbSearch, BicScore
from pgmpy.models import BayesianNetwork
from pgmpy.estimators import MaximumLikelihoodEstimator
```

Data Preprocessing

In [ ]:
```
data = pd.read_csv('ObesityDataSet.csv')
data
```

Out[ ]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	
0	Female	21.000000	1.620000	64.000000	yes	no	2.0	3.0	So
1	Female	21.000000	1.520000	56.000000	yes	no	3.0	3.0	So
2	Male	23.000000	1.800000	77.000000	yes	no	2.0	3.0	So
3	Male	27.000000	1.800000	87.000000	no	no	3.0	3.0	So
4	Male	22.000000	1.780000	89.800000	no	no	2.0	1.0	So
...	...	...	...	...	...	...	...	...	
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.0	3.0	So
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.0	3.0	So
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.0	3.0	So
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.0	3.0	So
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.0	3.0	So

2111 rows × 17 columns

In [ ]:
```
print(data['NObeyesdad'].value_counts())
```

```
NObeyesdad
Obesity_Type_I         351
Obesity_Type_III       324
Obesity_Type_II        297
Overweight_Level_I     290
Overweight_Level_II    290
Normal_Weight          287
Insufficient_Weight    272
Name: count, dtype: int64
```

In [ ]:
```
print(data['CALC'].value_counts())
print(data['MTRANS'].value_counts())
```

```
CALC
Sometimes     1401
no             639
Frequently      70
Always           1
Name: count, dtype: int64
MTRANS
Public_Transportation    1580
Automobile                457
Walking                    56
Motorbike                  11
Bike                        7
Name: count, dtype: int64
```

In [ ]:
```
data['CALC'] = data['CALC'].replace('Always', 'Frequently')
data['MTRANS'] = data['MTRANS'].replace('Bike', 'Walking')
```

In [ ]:
```
def encode_categorical_columns(dataset):
    # Identify categorical columns
    categorical_cols = dataset.select_dtypes(include=['object']).columns.tolist()

    label_encoders = {}
    for col in categorical_cols:
        if dataset[col].dtype == 'object':
            le = LabelEncoder()
            dataset[col] = le.fit_transform(dataset[col])
            label_encoders[col] = le
    return dataset, label_encoders

def balance_dataset(dataset, target):

    X = dataset.copy()
    X = X.drop(target, axis=1)

    # Identify categorical columns (assuming object type columns are categorical)
    categorical_cols = X.select_dtypes(include=['object']).columns.tolist()

    # Encode categorical columns
    X, label_encoders = encode_categorical_columns(X)

    # Indices of categorical columns
    categorical_features_indices = [X.columns.tolist().index(col) for col in categorical_cols]

    # Apply SMOTENC
    smotenc = SMOTENC(categorical_features=categorical_features_indices, random_state=0)
    X_resampled, y_resampled = smotenc.fit_resample(X, dataset[target])

    # Combine back into a DataFrame
    resampled_data = pd.DataFrame(X_resampled, columns=X.columns)
    resampled_data[target] = y_resampled

    return (shuffle(resampled_data, random_state=0), label_encoders)  # Shuffle the dataset
```

In [ ]:
```
(dataset, label_encoders) = balance_dataset(data, 'NObeyesdad')
```

In [ ]:
```
# Binning each column and storing the bin edges
data_binned = dataset.copy()
bins_dict = {}

for column in data_binned.columns:
    if data_binned[column].dtype == 'float64':
        data_binned[column], bins = pd.qcut(data_binned[column], q=5, retbins=True, duplicates
        bins_dict[column] = bins
```

In [ ]:
```
data_binned.dtypes
```

Out[ ]:
```
Gender                              int32
Age                              category
Height                           category
Weight                           category
family_history_with_overweight      int32
FAVC                                int32
FCVC                             category
NCP                              category
CAEC                                int32
SMOKE                               int32
CH2O                             category
SCC                                 int32
FAF                              category
TUE                              category
CALC                                int32
MTRANS                              int32
NObeyesdad                         object
dtype: object
```

In [ ]:
```python
# Split data into training and test sets for model evaluation
train_data, test_data = train_test_split(data_binned, test_size=0.2, random_state=42)
```

In [ ]:
```python
print(len(train_data['MTRANS'].value_counts()))
print(len(test_data['MTRANS'].value_counts()))
```

```
4
4
```

In [ ]:
```python
test_data
```

Out[ ]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC
1143	1	(23.807, 28.342]	(1.676, 1.735]	(77.355, 89.427]	1	1	(0.999, 2.0]	(2.131, 3.0]	2
619	0	(13.999, 19.006]	(1.449, 1.619]	(38.999, 60.0]	0	1	(0.999, 2.0]	(0.999, 2.131]	2
554	1	(13.999, 19.006]	(1.676, 1.735]	(38.999, 60.0]	0	1	(0.999, 2.0]	(0.999, 2.131]	2
1742	1	(23.807, 28.342]	(1.785, 1.98]	(111.836, 173.0]	1	1	(2.743, 3.0]	(2.131, 3.0]	2
2381	0	(28.342, 61.0]	(1.619, 1.676]	(60.0, 77.355]	1	1	(2.0, 2.052]	(2.131, 3.0]	2
...	...	...	...	...	...	...	...	...	...
773	0	(21.323, 23.807]	(1.449, 1.619]	(60.0, 77.355]	1	1	(2.052, 2.743]	(2.131, 3.0]	2
1526	1	(28.342, 61.0]	(1.785, 1.98]	(111.836, 173.0]	1	1	(2.743, 3.0]	(2.131, 3.0]	2
427	0	(19.006, 21.323]	(1.619, 1.676]	(38.999, 60.0]	0	1	(2.743, 3.0]	(2.131, 3.0]	3
909	1	(21.323, 23.807]	(1.676, 1.735]	(60.0, 77.355]	1	1	(0.999, 2.0]	(2.131, 3.0]	2
560	1	(13.999, 19.006]	(1.735, 1.785]	(38.999, 60.0]	1	1	(0.999, 2.0]	(3.0, 4.0]	2

492 rows × 17 columns

In [ ]:
```
print(train_data['Age'].value_counts())
print(test_data['Age'].value_counts())
```

```
Age
(23.807, 28.342]    398
(13.999, 19.006]    395
(19.006, 21.323]    394
(28.342, 61.0]      393
(21.323, 23.807]    385
Name: count, dtype: int64
Age
(21.323, 23.807]    106
(28.342, 61.0]       99
(13.999, 19.006]     97
(19.006, 21.323]     97
(23.807, 28.342]     93
Name: count, dtype: int64
```

Train Bayesian Network

In [ ]:
```
best_model = HillClimbSearch(train_data).estimate(scoring_method=BicScore(train_data))
```

In [ ]:
```
list(best_model.edges())
```

Out[ ]:
```
[('Gender', 'NObeyesdad'),
 ('Gender', 'Height'),
 ('Gender', 'CALC'),
 ('Gender', 'family_history_with_overweight'),
 ('Age', 'MTRANS'),
 ('Height', 'Weight'),
 ('Height', 'NCP'),
 ('Height', 'FAVC'),
 ('Weight', 'CALC'),
 ('Weight', 'CH2O'),
 ('Weight', 'SCC'),
 ('FAVC', 'NObeyesdad'),
 ('CALC', 'SMOKE'),
 ('NObeyesdad', 'Age'),
 ('NObeyesdad', 'FCVC'),
 ('NObeyesdad', 'family_history_with_overweight'),
 ('NObeyesdad', 'CAEC'),
 ('NObeyesdad', 'NCP'),
 ('NObeyesdad', 'FAF'),
 ('NObeyesdad', 'TUE'),
 ('NObeyesdad', 'Weight')]
```

In [ ]:
```
model = BayesianNetwork(best_model.edges())
model.fit(train_data, estimator=MaximumLikelihoodEstimator)
```

In [ ]:

```
# Assuming your test_data has both features and the target variable
# Splitting test_data into X_test (features) and y_test (target)
X_test = test_data.drop(columns=['NObeyesdad'])  # replace 'target_column' with your actual ta
y_test = test_data['NObeyesdad']

# Generate predictions for the test set
predictions = model.predict(X_test)

# In case of probabilistic outcomes, convert probabilities to concrete predictions
# This step depends on the nature of your data and model

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Optionally, print a confusion matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)
```

```
  0%|          | 0/438 [00:00<?, ?it/s]100%|██████████| 438/438 [00:00<00:00, 1360.73it/s]
Accuracy: 0.8536585365853658
Confusion Matrix:
 [[71  8  0  0  0  0  0]
 [ 5 52  0  0  0  5  1]
 [ 0  0 50  3  0  7 11]
 [ 0  0  3 51  0  0  1]
 [ 0  0  0  0 81  0  0]
 [ 1  5  0  0  0 56  8]
 [ 0  1  9  0  0  4 59]]
```

In [ ]:

```
print(train_data['MTRANS'].value_counts())
print(test_data['MTRANS'].value_counts())
```

```
MTRANS
2    1481
0     428
3      48
1       8
Name: count, dtype: int64
MTRANS
2    371
0    102
3     16
1      3
Name: count, dtype: int64
```

In [ ]:

```
# Display CPDs
for cpd in model.get_cpds():
    print("CPD of {0}:".format(cpd.variable))
    print(cpd)
```

CPD of Gender:

Gender(0)	0.482443
Gender(1)	0.517557

CPD of NObeyesdad:

FAVC	...	FAVC(1)
Gender	...	Gender(1)
NObeyesdad(Insufficient_Weight)	...	0.11336898395721925
NObeyesdad(Normal_Weight)	...	0.11657754010695187
NObeyesdad(Obesity_Type_I)	...	0.15935828877005348
NObeyesdad(Obesity_Type_II)	...	0.3090909090909091
NObeyesdad(Obesity_Type_III)	...	0.0010695187165775401
NObeyesdad(Overweight_Level_I)	...	0.13903743315508021
NObeyesdad(Overweight_Level_II)	...	0.16149732620320856

CPD of Height:

Gender	Gender(0)	Gender(1)
Height((1.449, 1.619])	0.39345991561181437	0.025565388397246803
Height((1.619, 1.676])	0.310126582278481	0.09636184857423795
Height((1.676, 1.735])	0.16139240506329114	0.22517207472959685
Height((1.735, 1.785])	0.10443037974683544	0.29793510324483774
Height((1.785, 1.98])	0.03059071729957806	0.35496558505408066

CPD of CALC:

Gender	Gender(0)	...	Gender(1)
Weight	Weight((38.999, 60.0])	...	Weight((111.836, 173.0])
CALC(0)	0.010416666666666666	...	0.008333333333333333
CALC(1)	0.5902777777777778	...	0.9708333333333333
CALC(2)	0.3993055555555556	...	0.020833333333333332

CPD of family_history_with_overweight:

Gender	...	Gender(1)
NObeyesdad	...	NObeyesdad(Overweight_Level_II)
family_history_with_overweight(0)	...	0.06936416184971098
family_history_with_overweight(1)	...	0.930635838150289

CPD of Age:

NObeyesdad	...	NObeyesdad(Overweight_Level_II)

Age((13.999, 19.006])	...	0.1079136690647482
Age((19.006, 21.323])	...	0.15827338129496402
Age((21.323, 23.807])	...	0.21223021582733814
Age((23.807, 28.342])	...	0.1366906474820144
Age((28.342, 61.0])	...	0.38489208633093525

CPD of MTRANS:

Age	Age((13.999, 19.006])	...	Age((28.342, 61.0])
MTRANS(0)	0.12658227848101267	...	0.7786259541984732
MTRANS(1)	0.002531645569620253	...	0.002544529262086514
MTRANS(2)	0.8253164556962025	...	0.2010178117048346
MTRANS(3)	0.04556962025316456	...	0.017811704834605598

CPD of Weight:

Height	...	Height((1.785, 1.98])
NObeyesdad	...	NObeyesdad(Overweight_Level_II)
Weight((38.999, 60.0])	...	0.0
Weight((60.0, 77.355])	...	0.0
Weight((77.355, 89.427])	...	0.019230769230769232
Weight((89.427, 111.836])	...	0.9807692307692307
Weight((111.836, 173.0])	...	0.0

CPD of NCP:

Height	...	Height((1.785, 1.98])
NObeyesdad	...	NObeyesdad(Overweight_Level_II)
NCP((0.999, 2.131])	...	0.5
NCP((2.131, 3.0])	...	0.5
NCP((3.0, 4.0])	...	0.0

CPD of FAVC:

Height	Height((1.449, 1.619])	...	Height((1.785, 1.98])
FAVC(0)	0.2556390977443609	...	0.04358974358974359
FAVC(1)	0.7443609022556391	...	0.9564102564102565

CPD of CH2O:

Weight	...	Weight((111.836, 173.0])
CH2O((0.999, 1.386])	...	0.0472636815920398
CH2O((1.386, 2.0])	...	0.23880597014925373

CH2O((2.0, 2.036])	...	0.06965174129353234
CH2O((2.036, 2.619])	...	0.3681592039800995
CH2O((2.619, 3.0])	...	0.27611940298507465

CPD of SCC:

Weight	Weight((38.999, 60.0])	...	Weight((111.836, 173.0])
SCC(0)	0.9073634204275535	...	0.9950248756218906
SCC(1)	0.09263657957244656	...	0.004975124378109453

CPD of SMOKE:

CALC	CALC(0)	CALC(1)	CALC(2)
SMOKE(0)	0.8888888888888888	0.9813664596273292	0.99185667752443
SMOKE(1)	0.1111111111111111	0.018633540372670808	0.008143322475570033

CPD of FCVC:

NObeyesdad	...	NObeyesdad(Overweight_Level_II)
FCVC((0.999, 2.0])	...	0.5035971223021583
FCVC((2.0, 2.052])	...	0.046762589928057555
FCVC((2.052, 2.743])	...	0.21942446043165467
FCVC((2.743, 3.0])	...	0.2302158273381295

CPD of CAEC:

NObeyesdad	...	NObeyesdad(Overweight_Level_II)
CAEC(0)	...	0.007194244604316547
CAEC(1)	...	0.039568345323741004
CAEC(2)	...	0.9496402877697842
CAEC(3)	...	0.0035971223021582736

CPD of FAF:

NObeyesdad	...	NObeyesdad(Overweight_Level_II)
FAF((-0.001, 0.0296])	...	0.19784172661870503
FAF((0.0296, 0.802])	...	0.20863309352517986
FAF((0.802, 1.075])	...	0.2805755395683453
FAF((1.075, 1.947])	...	0.17266187050359713
FAF((1.947, 3.0])	...	0.14028776978417265

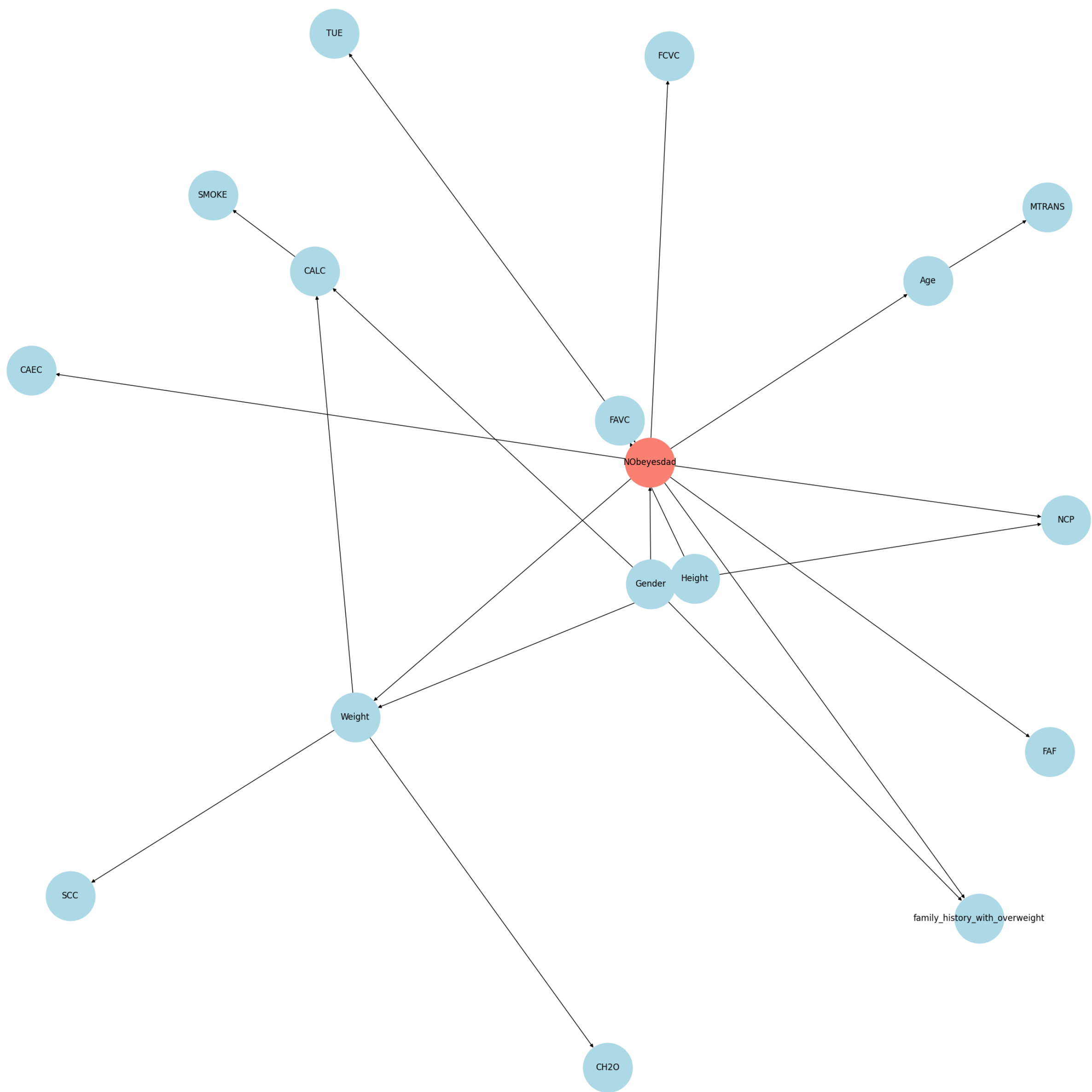
CPD of TUE:

NObeyesdad	...	NObeyesdad(Overweight_Level_II)
TUE((-0.001, 0.369])	...	0.34532374100719426

```
| TUE((0.369, 0.836]) | ... | 0.20503597122302158              |
+---------------------+-----+----------------------------------+
| TUE((0.836, 1.0])   | ... | 0.22661870503597123              |
+---------------------+-----+----------------------------------+
| TUE((1.0, 2.0])     | ... | 0.22302158273381295              |
+---------------------+-----+----------------------------------+
```

In [ ]:

```
# Create a NetworkX graph from the Bayesian Network
nx_graph = nx.DiGraph()
nx_graph.add_nodes_from(model.nodes())
nx_graph.add_edges_from(model.edges())

# Visualize the network structure
pos = nx.spring_layout(nx_graph, seed= 12)

# Annotate the graph
plt.figure(figsize=(25, 25))
nx.draw(nx_graph, pos, with_labels=True, node_size=5000, node_color='lightblue')
nx.draw_networkx_nodes(nx_graph, pos, nodelist=['NObeyesdad'], node_size=5000, node_color="sal
plt.show()
```



Inference Computation

In [ ]:
```
def truncate(number, decimals=0):
    factor = 10 ** decimals
    return int(number * factor) / factor

def get_encoded_label(label_encoders, bins, column, value):
    if column in bins:
        bin_edges = bins[column]
        for i in range(len(bin_edges) - 1):
            if bin_edges[i] <= value < bin_edges[i + 1]:
                if i == 0 :
                    return pd.Interval(left=truncate(bin_edges[i]-0.00001,3), right=round(bin_
                else:
                    return pd.Interval(left=round(bin_edges[i],3), right=round(bin_edges[i + 1
    return label_encoders[column].transform([value])[0]

def render_questions(question_dict):
    # Convert DiscreteFactor objects to dictionaries
    for weight in question_dict:
        factor = question_dict[weight]
        question_dict[weight] = {state: prob for state, prob in zip(factor.state_names['NObeye

    # Manually specify the order of classes
    ordered_classes = ['Insufficient_Weight', 'Normal_Weight', 'Overweight_Level_I', 'Overweig

    # Prepare data for plotting
    weights = sorted(question_dict.keys())
    n_classes = len(ordered_classes)
    n_weights = len(weights)

    # Data for plotting
    barWidth = 0.2
    r = np.arange(n_classes)

    # Plotting
    plt.figure(figsize=(12, 8))

    for i, weight in enumerate(weights):
        probabilities = [question_dict[weight].get(cls, 0) for cls in ordered_classes]
        bars = plt.bar(r + i * barWidth, probabilities, width=barWidth, edgecolor='grey', labe

    plt.xlabel('Weight Class', fontweight='bold')
    plt.xticks([r + barWidth for r in range(n_classes)], ordered_classes, rotation=45)
    plt.ylabel('Probability (phi)')
    plt.title('Probability Distribution of NObeyesdad Categories by Weight')

    plt.legend()
    plt.show()
```

Testing the Encoder:

In [ ]:
```
print(get_encoded_label(label_encoders,bins_dict,'family_history_with_overweight', 'yes'))
print(get_encoded_label(label_encoders,bins_dict,'Weight', 99))
```

```
1
(89.427, 111.836]
```

In [ ]:
```
from pgmpy.inference import VariableElimination

inference = VariableElimination(model)
```

In [ ]:
```
q = inference.query(variables=['NObeyesdad'], evidence={'family_history_with_overweight': get_
print(q)
```

```
+----------------------------------+--------------------+
| NObeyesdad                       |   phi(NObeyesdad) |
+==================================+====================+
| NObeyesdad(Insufficient_Weight) |             0.0829 |
+----------------------------------+--------------------+
| NObeyesdad(Normal_Weight)        |             0.1016 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)       |             0.1700 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)      |             0.1852 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)     |             0.1670 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)   |             0.1277 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II) |             0.1656 |
+----------------------------------+--------------------+
```

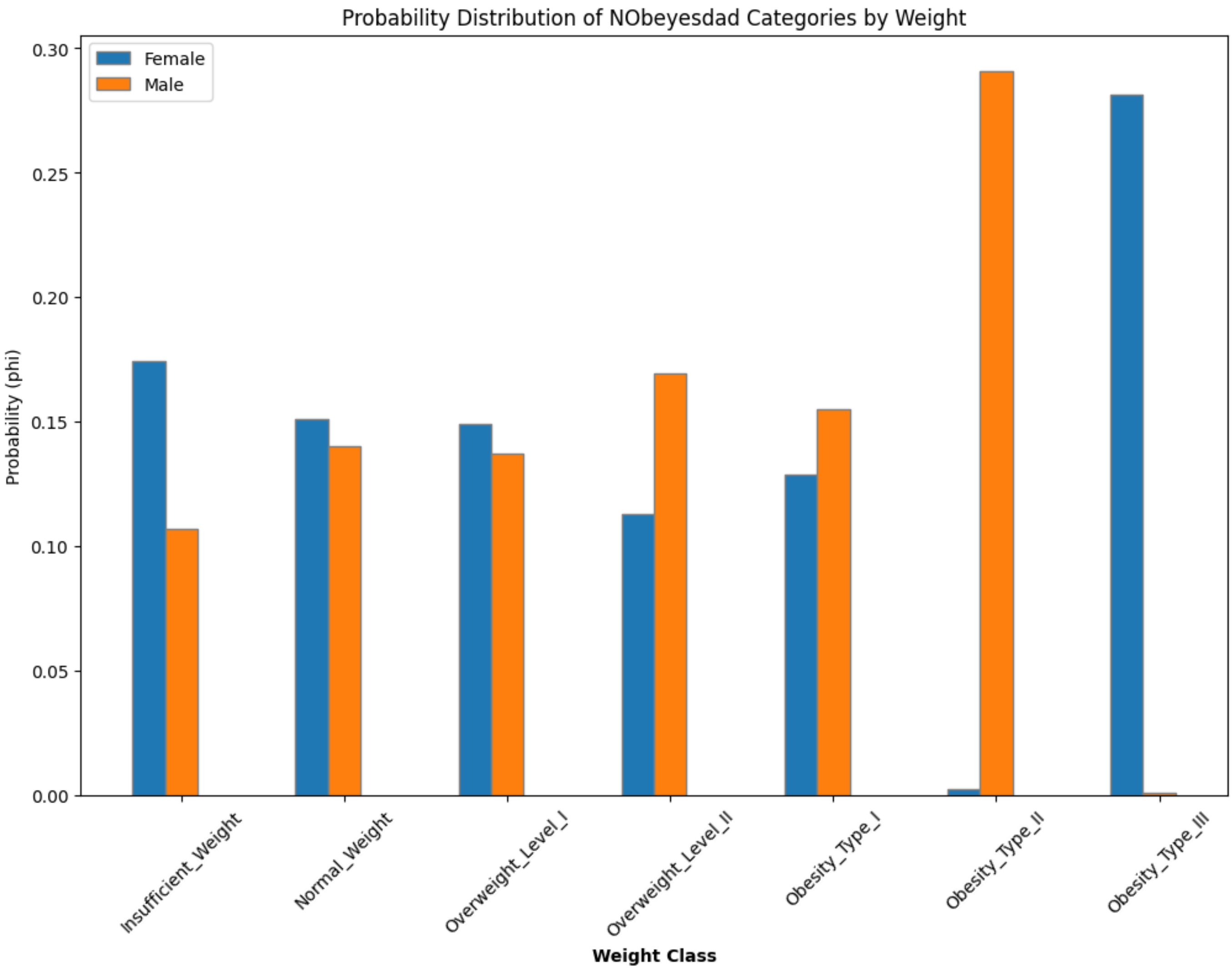
Testing the fallback for the binned variables:

In [ ]:
```
q = inference.query(variables=['NObeyesdad'], evidence={'Weight': pd.Interval(left=38.999, rig

print(q)
```

```
+----------------------------------+--------------------+
| NObeyesdad                       |   phi(NObeyesdad) |
+==================================+====================+
| NObeyesdad(Insufficient_Weight) |             0.6845 |
+----------------------------------+--------------------+
| NObeyesdad(Normal_Weight)        |             0.2961 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)       |             0.0000 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)      |             0.0000 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)     |             0.0000 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)   |             0.0163 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II) |             0.0032 |
+----------------------------------+--------------------+
```

Testing the result rendering on a simple question of the difference between the likely weight classes per gender:

In [ ]:
```
# Sample data from your queries
results = {
    'Male': inference.query(variables=['NObeyesdad'], evidence={
    'Gender': get_encoded_label(label_encoders,bins_dict,'Gender', 'Male')
    }),
    'Female': inference.query(variables=['NObeyesdad'], evidence={
    'Gender': get_encoded_label(label_encoders,bins_dict,'Gender', 'Female')
    })
}

render_questions(results)
```



What is the impact of technology use, alongside physical activity on obesity?

For the primary question we will investigate the probability of each weight class given that the person spends a large amount of time on technology devices, with little physical activity.

We will then investigate the difference with an increased physical activity,

And lastly, investigate the effect of maximal physical activity.

In [ ]:
```
print(data_binned['FAF'].value_counts())
print(data_binned['TUE'].value_counts())
```

```
FAF
(-0.001, 0.0296]    492
(1.947, 3.0]        492
(0.0296, 0.802]     491
(0.802, 1.075]      491
(1.075, 1.947]      491
Name: count, dtype: int64
TUE
(-0.001, 0.369]    983
(0.836, 1.0]       506
(0.369, 0.836]     491
(1.0, 2.0]         477
Name: count, dtype: int64
```

In [ ]:
```
q1_1 = inference.query(variables=['NObeyesdad'], evidence={
    'TUE': get_encoded_label(label_encoders,bins_dict,'TUE', 1.9), # Time using technology dev
    'FAF': pd.Interval(left=0.0296, right=0.802) # Physical activity frequency
    })
print(q1_1)
```

```
+----------------------------------+--------------------+
| NObeyesdad                       |   phi(NObeyesdad) |
+==================================+====================+
| NObeyesdad(Insufficient_Weight) |             0.2311 |
+----------------------------------+--------------------+
| NObeyesdad(Normal_Weight)        |             0.0117 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)       |             0.1887 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)      |             0.2008 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)     |             0.0000 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)   |             0.2051 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II) |             0.1626 |
+----------------------------------+--------------------+
```

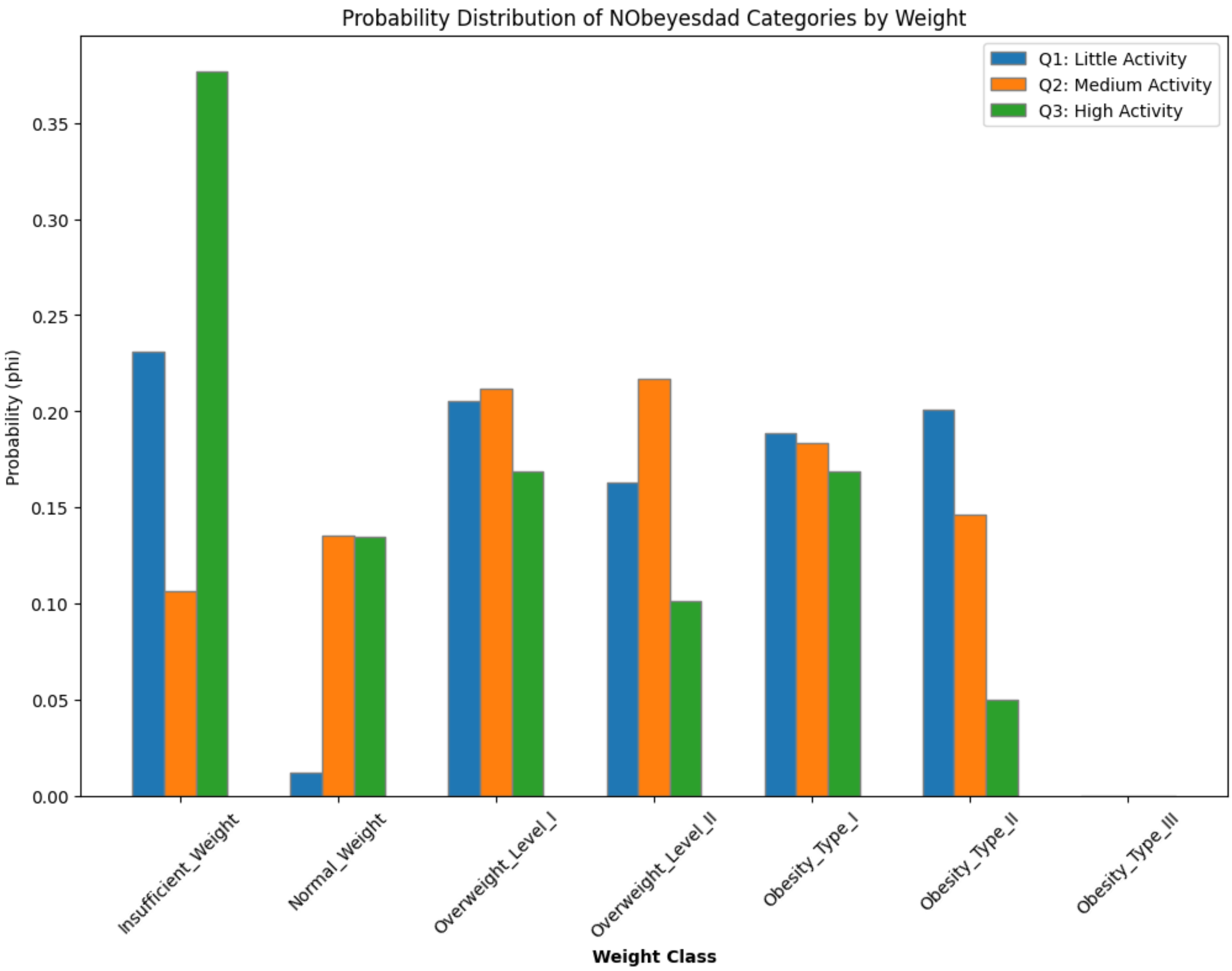
In [ ]:
```
q1_2 = inference.query(variables=['NObeyesdad'], evidence={
    'TUE': get_encoded_label(label_encoders,bins_dict,'TUE', 1.9), # Time using technology dev
    'FAF': get_encoded_label(label_encoders,bins_dict,'FAF', 1) # Physical activity frequency
    })
print(q1_2)
```

```
+----------------------------------+--------------------+
| NObeyesdad                       |   phi(NObeyesdad) |
+==================================+====================+
| NObeyesdad(Insufficient_Weight) |             0.1066 |
+----------------------------------+--------------------+
| NObeyesdad(Normal_Weight)        |             0.1351 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)       |             0.1837 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)      |             0.1460 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)     |             0.0000 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)   |             0.2119 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II) |             0.2167 |
+----------------------------------+--------------------+
```

In [ ]:
```
q1_3 = inference.query(variables=['NObeyesdad'], evidence={
    'TUE': get_encoded_label(label_encoders,bins_dict,'TUE', 1.9), # Time using technology dev
    'FAF': get_encoded_label(label_encoders,bins_dict,'FAF', 2.9) # Physical activity frequenc
    })
print(q1_3)
```

NObeyesdad	phi(NObeyesdad)
NObeyesdad(Insufficient_Weight)	0.3769
NObeyesdad(Normal_Weight)	0.1346
NObeyesdad(Obesity_Type_I)	0.1687
NObeyesdad(Obesity_Type_II)	0.0497
NObeyesdad(Obesity_Type_III)	0.0000
NObeyesdad(Overweight_Level_I)	0.1688
NObeyesdad(Overweight_Level_II)	0.1014

In [ ]:
```
results = {
    'Q1: Little Activity': q1_1,
    'Q2: Medium Activity': q1_2,
    'Q3: High Activity': q1_3
}

render_questions(results)
```



From the above we can see that, spending a large amount of time using technology devices (likely sitting at a desk or couch), while doing little physical activity has very little chance (0.0117) to result in normal weight, while causing having a 23% chance of being underweight, and a cumulative 75.7% chance of ranging from Overweight Level I to Obesity Type II.

A moderate increase in physical activity increases your probability of being of Normal Weight to 13.5%, decreases your probability of Obesity Type II, as well as your probability of having Insufficient Weight.

High physical activity can decrease your probability of Obesity Type II to 5% and your cumulative chance to be overweight or obese to 48.9%. It does however cause a significant probability of having Insufficient Weight.

How does a healthy diet impact CVD risk when you have a genetic predisposition (Family member suffered or suffers from overweight)?

Firstly the effect of having a Family member whom suffered or suffers from overweight has on how likely you are to fall into a given weight class is investigated.

Then it will be investigated how a increasingly healthy diet affects the outcomes:

- Not consuming high caloric food
- Consuming healthy amounts of vegetables and water
- Not smoking or drinking

In [ ]:

```
print(data['family_history_with_overweight'].value_counts())
print(data['FAVC'].value_counts())
print(data_binned['FCVC'].value_counts())
print(data_binned['CH2O'].value_counts())
print(data['SMOKE'].value_counts())
print(data['CALC'].value_counts())
```

```
family_history_with_overweight
yes    1726
no      385
Name: count, dtype: int64
FAVC
yes    1866
no      245
Name: count, dtype: int64
FCVC
(2.743, 3.0]      983
(0.999, 2.0]      915
(2.052, 2.743]    491
(2.0, 2.052]       68
Name: count, dtype: int64
CH2O
(1.386, 2.0]      921
(0.999, 1.386]    492
(2.619, 3.0]      492
(2.036, 2.619]    491
(2.0, 2.036]       61
Name: count, dtype: int64
SMOKE
no     2067
yes      44
Name: count, dtype: int64
CALC
Sometimes     1401
no             639
Frequently      71
Name: count, dtype: int64
```

In [ ]:

```
q2_1 = inference.query(variables=['NObeyesdad'], evidence={
    'family_history_with_overweight': get_encoded_label(label_encoders,bins_dict,'family_histo
    })
print(q2_1)
```

```
+-----------------------------------+-------------------+
| NObeyesdad                        |   phi(NObeyesdad) |
+===================================+===================+
| NObeyesdad(Insufficient_Weight)   |            0.0829 |
+-----------------------------------+-------------------+
| NObeyesdad(Normal_Weight)         |            0.1016 |
+-----------------------------------+-------------------+
| NObeyesdad(Obesity_Type_I)        |            0.1700 |
+-----------------------------------+-------------------+
| NObeyesdad(Obesity_Type_II)       |            0.1852 |
+-----------------------------------+-------------------+
| NObeyesdad(Obesity_Type_III)      |            0.1670 |
+-----------------------------------+-------------------+
| NObeyesdad(Overweight_Level_I)    |            0.1277 |
+-----------------------------------+-------------------+
| NObeyesdad(Overweight_Level_II)   |            0.1656 |
+-----------------------------------+-------------------+
```

In [ ]:

```
q2_2 = inference.query(variables=['NObeyesdad'], evidence={
    'family_history_with_overweight': get_encoded_label(label_encoders,bins_dict,'family_histo
    'FAVC': get_encoded_label(label_encoders,bins_dict,'FAVC', 'no') # Consumption of high cal
    })
print(q2_2)
```

```
+---------------------------------+--------------------+
| NObeyesdad                      |   phi(NObeyesdad) |
+=================================+====================+
| NObeyesdad(Insufficient_Weight) |             0.0932 |
+---------------------------------+--------------------+
| NObeyesdad(Normal_Weight)       |             0.2343 |
+---------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)      |             0.0541 |
+---------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)     |             0.0342 |
+---------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)    |             0.0071 |
+---------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)  |             0.0982 |
+---------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II) |             0.4790 |
+---------------------------------+--------------------+
```

In [ ]:

```
q2_3 = inference.query(variables=['NObeyesdad'], evidence={
    'family_history_with_overweight': get_encoded_label(label_encoders,bins_dict,'family_histo
    'FAVC': get_encoded_label(label_encoders,bins_dict,'FAVC', 'no'), # Consumption of high co
    'FCVC': get_encoded_label(label_encoders,bins_dict,'FCVC', 2.9), # Consumption of vegetabl
    'CH2O': get_encoded_label(label_encoders,bins_dict,'CH2O', 2) # Consumption of water daily
    })
print(q2_3)
```

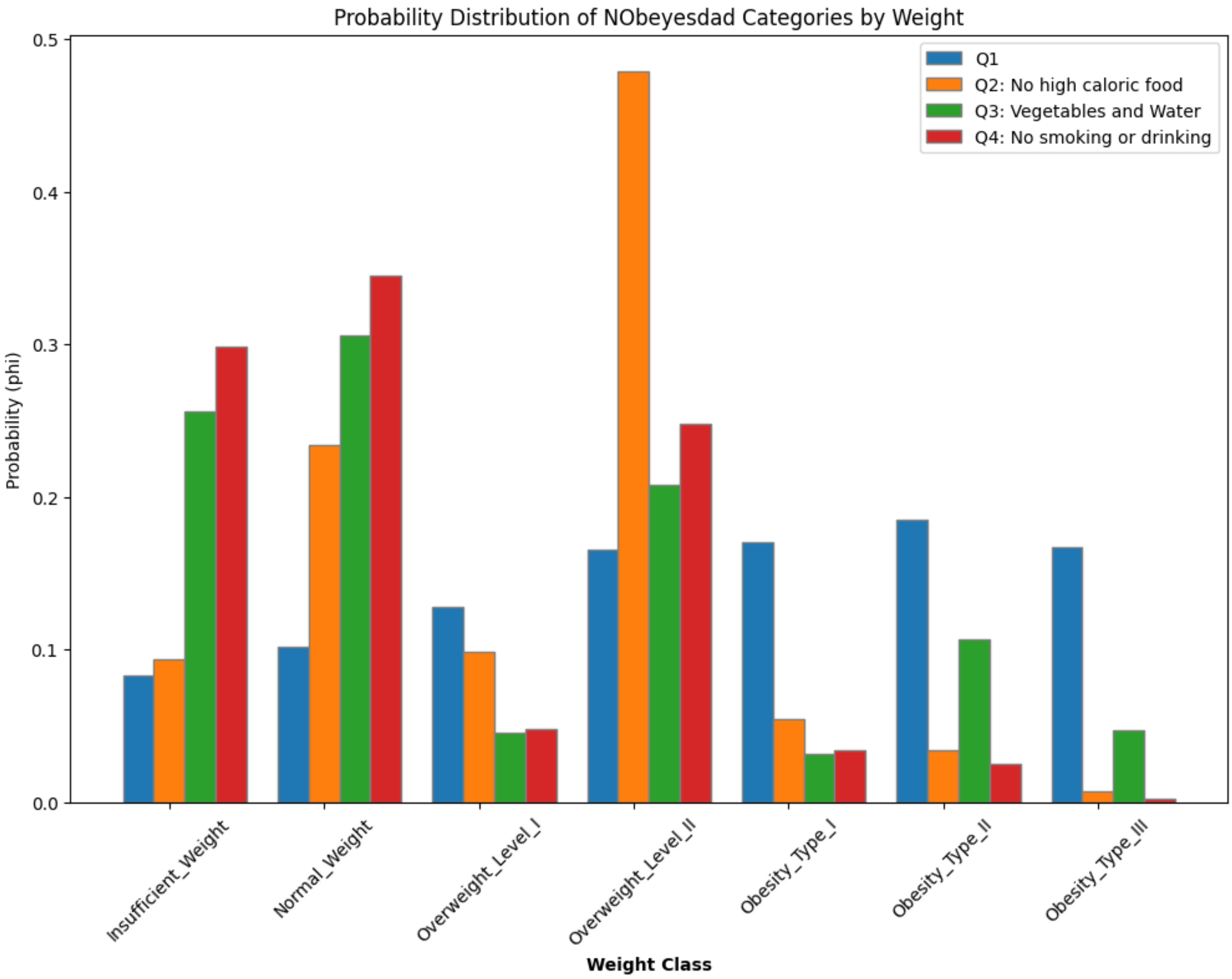
```
+---------------------------------+--------------------+
| NObeyesdad                      |   phi(NObeyesdad) |
+=================================+====================+
| NObeyesdad(Insufficient_Weight) |             0.2557 |
+---------------------------------+--------------------+
| NObeyesdad(Normal_Weight)       |             0.3058 |
+---------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)      |             0.0319 |
+---------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)     |             0.1063 |
+---------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)    |             0.0468 |
+---------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)  |             0.0457 |
+---------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II) |             0.2078 |
+---------------------------------+--------------------+
```

In [ ]:

```
q2_4 = inference.query(variables=['NObeyesdad'], evidence={
    'family_history_with_overweight': get_encoded_label(label_encoders,bins_dict,'family_histo
    'FAVC': get_encoded_label(label_encoders,bins_dict,'FAVC', 'no'), # Consumption of high co
    'FCVC': get_encoded_label(label_encoders,bins_dict,'FCVC', 2.9), # Consumption of vegetabl
    'CH2O': get_encoded_label(label_encoders,bins_dict,'CH2O', 2), # Consumption of water dail
    'CALC': get_encoded_label(label_encoders,bins_dict,'CALC', 'no'), # Consumption of alcohol
    'SMOKE': get_encoded_label(label_encoders,bins_dict,'SMOKE', 'no'), # Smoker or not
    })
print(q2_4)
```

```
+-----------------------------------+-------------------+
| NObeyesdad                        |   phi(NObeyesdad) |
+===================================+===================+
| NObeyesdad(Insufficient_Weight)   |            0.2983 |
+-----------------------------------+-------------------+
| NObeyesdad(Normal_Weight)         |            0.3453 |
+-----------------------------------+-------------------+
| NObeyesdad(Obesity_Type_I)        |            0.0337 |
+-----------------------------------+-------------------+
| NObeyesdad(Obesity_Type_II)       |            0.0251 |
+-----------------------------------+-------------------+
| NObeyesdad(Obesity_Type_III)      |            0.0018 |
+-----------------------------------+-------------------+
| NObeyesdad(Overweight_Level_I)    |            0.0480 |
+-----------------------------------+-------------------+
| NObeyesdad(Overweight_Level_II)   |            0.2478 |
+-----------------------------------+-------------------+
```

In [ ]:
```python
results = {
    'Q1': q2_1,
    'Q2: No high caloric food': q2_2,
    'Q3: Vegetables and Water': q2_3,
    'Q4: No smoking or drinking': q2_4
}

render_questions(results)
```



From the above it is shown that having a Family member who suffered or suffers from overweight does indicate risk of being overweight or obese, with a person having only a probability of 10.1% of being Normal Weight, and 8.3% having Insufficient Weight. Resulting in a cumulative 81.6% probability of being on the range of Overweight Level I to Obesity Type III, whit the highest being Obesity Type II at 18.5%.

The follow up questions then indicate that you can significantly reduce your probability of Obesity Type I - III through cutting high caloric food, consuming healthy amounts of vegetables and water, and not smoking or drinking. Cutting high caloric food alone already increases the probability of being of Normal weight to 23.4%, a significant increase in Overweight level II can be seen, but this can be justified by the cumulative decrease of the probability for Obesity Level I - III from 52.2% to 9.5%.

Implementing the healthy diet fully can decrease the probability of Obesity Type I - III further to 6.1% and increase the probability to be of normal weight or lower to 64.4%

How does smoking and alcohol consumption, alongside mode of transportation correlate to obesity risk?

For this question, the effect of walking to work has, compared to driving, for a person who smokes and drinks alcohol frequently, on their weight classification.

In [ ]:
```
print(data['SMOKE'].value_counts())
print(data['CALC'].value_counts())
print(data['MTRANS'].value_counts())
```

```
SMOKE
no     2067
yes      44
Name: count, dtype: int64
CALC
Sometimes     1401
no             639
Frequently      71
Name: count, dtype: int64
MTRANS
Public_Transportation    1580
Automobile                457
Walking                    63
Motorbike                  11
Name: count, dtype: int64
```

In [ ]:
```
q3_1 = inference.query(variables=['NObeyesdad'], evidence={
    'SMOKE': get_encoded_label(label_encoders,bins_dict,'SMOKE', 'yes'),
    'CALC': get_encoded_label(label_encoders,bins_dict,'CALC', 'Frequently'),
    'MTRANS': get_encoded_label(label_encoders,bins_dict,'MTRANS', 'Automobile'),
    })
print(q3_1)
```

```
+----------------------------------+--------------------+
| NObeyesdad                       |   phi(NObeyesdad)  |
+==================================+====================+
| NObeyesdad(Insufficient_Weight)  |             0.0100 |
+----------------------------------+--------------------+
| NObeyesdad(Normal_Weight)        |             0.0668 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)       |             0.2556 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)      |             0.1830 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)     |             0.0001 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)   |             0.1550 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II)  |             0.3295 |
+----------------------------------+--------------------+
```

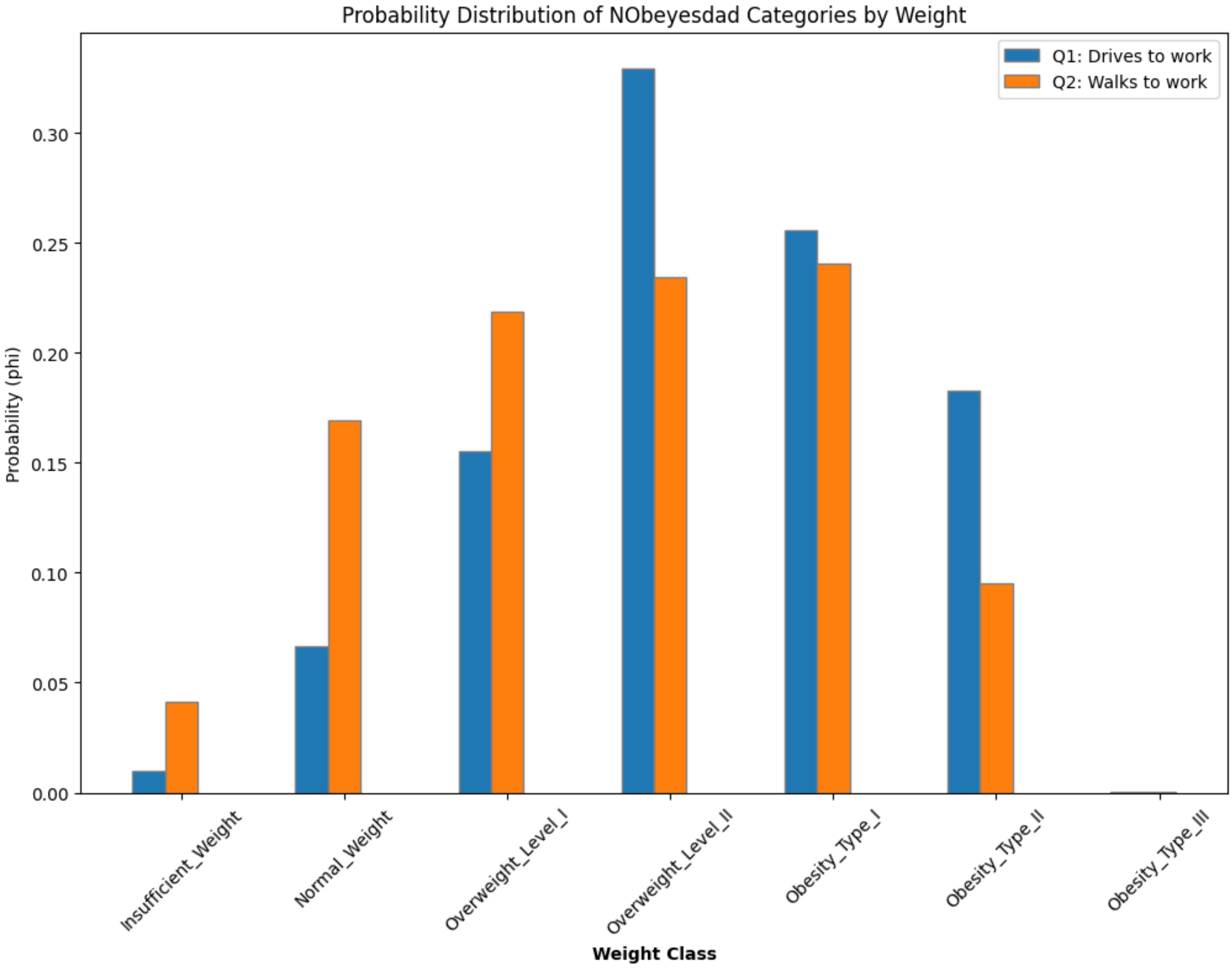
In [ ]:

```
q3_2 = inference.query(variables=['NObeyesdad'], evidence={
    'SMOKE': get_encoded_label(label_encoders,bins_dict,'SMOKE', 'yes'),
    'CALC': get_encoded_label(label_encoders,bins_dict,'CALC', 'Frequently'),
    'MTRANS': get_encoded_label(label_encoders,bins_dict,'MTRANS', 'Walking'),
    })
print(q3_2)
```

```
+----------------------------------+--------------------+
| NObeyesdad                       |   phi(NObeyesdad)  |
+==================================+====================+
| NObeyesdad(Insufficient_Weight)  |             0.0415 |
+----------------------------------+--------------------+
| NObeyesdad(Normal_Weight)        |             0.1691 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_I)       |             0.2406 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_II)      |             0.0954 |
+----------------------------------+--------------------+
| NObeyesdad(Obesity_Type_III)     |             0.0002 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_I)   |             0.2187 |
+----------------------------------+--------------------+
| NObeyesdad(Overweight_Level_II)  |             0.2345 |
+----------------------------------+--------------------+
```

In [ ]:

```
results = {
    'Q1: Drives to work': q3_1,
    'Q2: Walks to work': q3_2
}

render_questions(results)
```



From the above we can evaluate that smoking and drinking, but driving to work leans towards being overweight or obese, with a cumulative probability of 92.3.9%, the highest of which is Overweight Level II at 23.5%.

Switching mode of transport to walking increases the probability to be of normal weight form 6.7%, to 16.9%. And decreases the cumulative probability of being overweight or obese to 78.9%.