CP8318 Machine Learning - Assignment 1

Student: Nebojsa Djosic, nebojsa.djosic@torontomu.ca
Lecturer: Elodie Lugez, Ph.D., cps8318@cs.torontomu.ca

Contents

Introduction **1**

Part 1 **2**

Part 2 **3**

Part 2.1 . . . 3

Part 2.2 . . . 4

Part 3 **5**

Question 1 . . . 5

Question 2 . . . 6

Question 3 . . . 7

Question 4 . . . 7

Part 4 - Graduate Report **8**

Introduction . . . 8

Methodology . . . 10

Results . . . 10

Conclusions and Future Research . . . 12

Introduction

This report has four main sections in addition to this introduction. Each section is named according to the Assignment 1 Part numbering. All sections have corresponding Python code delivered in a separate file: *script_Nebojsa_Djosic.py*.

Section titled Part 4, is the graduate report, corresponding to Part 4 of Assignment 1. The code for the graduate part, Part 4, is in the same Python source code file: *script_Nebojsa_Djosic.py*.

The Python script requires a minimum of Python 3.12.3 and will install dependencies and create a log file that will be overwritten on every run. This log file will allow more efficient grading and evaluation or troubleshooting if necessary. In addition, it will create data and results subdirectories. Datasets will be downloaded only once and stored under the data subdirectory. All graphs will be stored in the results subdirectory and will be overwritten on each script run.

Please note that the script will detect and use all available CPU cores when executing the graduate

portion of the script. This can be changed in the code by setting AUTO_CPU_CORE to False and setting manually the value for CPU_CORES = 1.

When the script is running it will show several plots that must be closed. They will be saved to the results subdirectory, and the log will show progress. There will be a delay when the exhaustive search is performed especially if it is single threaded run. The log file will start with installing dependency messages, will have more than 200 lines, and will end with the message: *End of the graduate part (Part 4)..*

The script has been developed and tested on MacOS 15 and may not run as-is on Windows OS. If you experience issues try first commenting out all file operations, since this is a known source of cross-platform issues. If you still experience issues, please contact me.

Part 1

For full source code and detailed answers please refer to the *script_Nebojsa_Djosic.py* uploaded with this file as a part of the submission for this assignment.

We downloaded the Breast Cancer Wisconsin (Diagnostic) dataset consisting of 569 instances with 30 features and no missing values. Most of the features are continuous values corresponding to the measurements taken from imaging of cells of the breast tissue [3].

The full dataset is split into two: 40% for testing and 60% training.

> Training set shape: (341, 30) (341,)
>
> Test set shape: (228, 30) (228,)

A decision tree classifier model using entropy is created using the DecisionTreeClassifier from the sklearn library. After training and testing the model we got accuracy score of 0.9254 using the sklearn.metrics library.

The tree visualisation is achieved using the matplotlib.pyplot library.

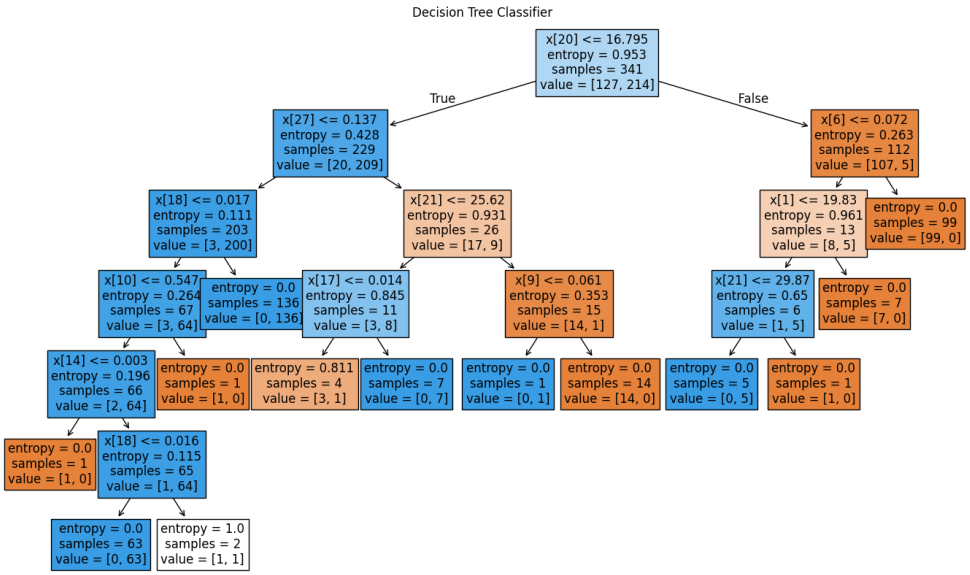


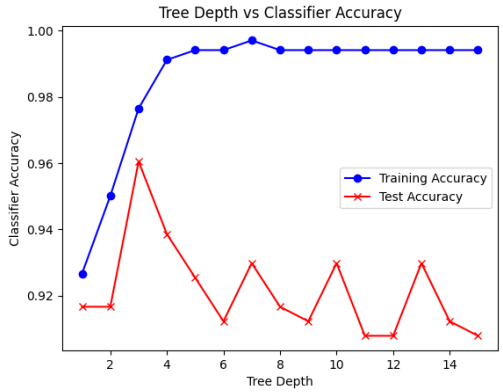
Figure 1: Decision Tree

Part 2

In this section, we will be plotting error and accuracy for a range of tree depths. This section has two subsections 2.1 and 2.2. For details see the Python code source file *script_Nebojsa_Djosic.py*.

Part 2.1

In this subsection, we are looping through a range between 1 and 16, and for each depth we are constructing, training and testing a decision-tree classifier. Each time we collected training and test accuracy in two separate lists. These lists are used to construct a line graph comparing the two accuracy lines representing the accuracy score with respect to the tree depth. We note here that the error is equal to 1 - accuracy and we show both graphs.



(a) Tree Depth vs Accuracy

(b) Tree Depth vs Error

Figure 2: Tree Depth

According to the test error, the best model to select is when the maximum depth is equal to 3, approximately. But, we should not use select the hyperparameters of our model using the test data, because it could lead to overfitting.

Part 2.2

In this section, we are using GridSearchCV from the sklearn library. This library is capable of empirically finding the best parameters for a decision tree: the best depth and the minimum number of samples for node split.

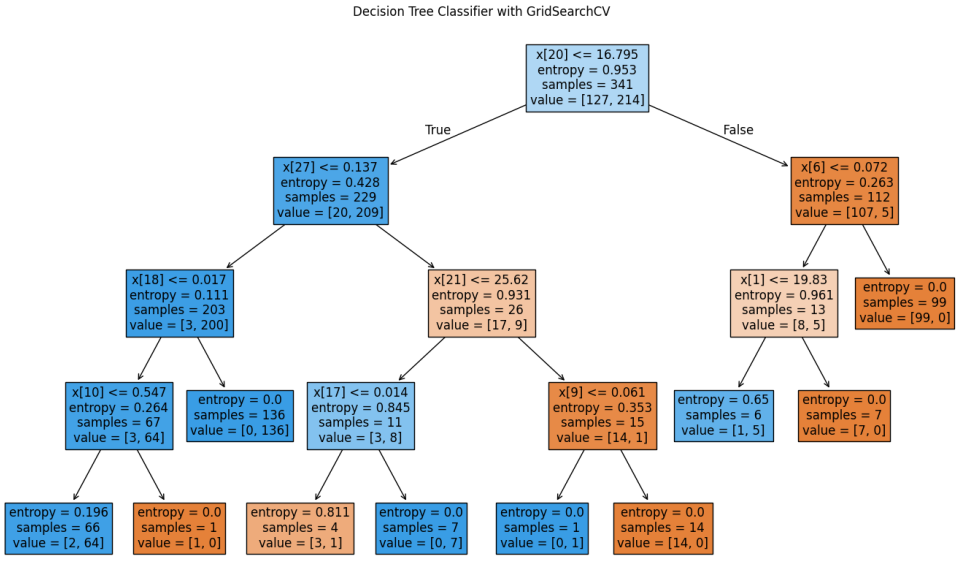


Figure 3: Decision Tree

We found that **the maximum depth of the tree is 4 and the minimum number of samples required to split a node is 6.**

Using an exhaustive search like this is a resource and time-consuming. However, in this specific case, it is acceptable, because:

- It is an exhaustive search of the hyperparameters, all possible combinations are tried
- It uses cross-validation to avoid overfitting
- It is reproducible due to the random_state parameter so we can compare
- It can handle the class imbalance handled by stratified cross-validation
- The Brest cancer dataset we're using is small so the computational cost is not too high

Tenfold stratified cross-validation is used to evaluate the performance of a model. The dataset is divided into 10 equal parts, and then the model is trained and tested 10 times. Each time, only one of the 10 is used for testing and all the rest for training. All the performance metrics like accuracy are averaged over all 10 iterations (folds). Stratified means that each dataset part has the same class distribution as the whole dataset. In the scikit-learn library, this is implemented in the StratifiedKFold(n_splits=10) and then cross_val_score(...) provides scores and we can print metrics.

Part 3

Question 1

What is the entropy of this collection of training examples with respect to the target class? (3 points)

Answer:

To calculate the entropy of the collection we use the following formula:

$$Entropy = -\sum_{i=1}^{c} p_i(t)\ log_2(p_i(t)) \tag{1}$$

where:

c is the number of classes, in our case 2 (True, False)

$(p_i(t))$ is the proportion of class (i) at node t

Calculation steps:

1. Count the number of instances for each class:

Number of True instances: 6

Number of False instances: 4

Total number of instances: $(6 + 4) = 10$

2. Calculate the proportions (probabilities):

Probability of True $p_(T) : 6/10 = 0.6$

Probability of False $p_(F) : 4/10 = 0.4$

3. Apply the entropy formula (2):

$Entropy = -(0.6\ log_2(0.6) + 0.4\ log_2(0.4))$

$Entropy = -(0.6 * -0.7370 + 0.4 * -1.3219)$

$Entropy = -(-0.4422 - 0.5288)$

$Entropy \approx 0.971$

The entropy of the collection of training examples with respect to the target class is approximately 0.971.

Question 2

What are the different options for the first split when constructing your decision tree?

Answer:

Features and Values:

Feature X1: Binary attribute with values 0, 1

Feature X2: Ordinal categorical attribute with values 0, 1, 2

Possible Splits:

Splits on X1 are clear since it is a binary attribute 0 or 1:

X1 = 0 or X1 = 1

Splits on X2 have more options since the values are 0, 1, and 2:

X2 = 0 or X2 != 0: either the value is 0 or not in which case it is 1 or 2

X2 = 1 vs. X2 != 1: either the value is 1 or not in which case it is 0 or 2

X2 = 2 vs. X2 != 2: either the value is 2 or not in which case it is 0 or 1

Question 3

For each potential first split option, compute the information gain. Only provide the results, there is no need to provide your calculations

Answer: Information gain is used to find the best split and relies of entropy before the split P and entropy after the split M: Gain = P - M.

To calculate the entropy of the collection we use the following formula:

$$Entropy(split) = \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \tag{2}$$

Where

n_i is the number of records at child i

n is the number of records at parent node p

Information gains are as follows for respective split on:

X1 = 0 or X1 = 1 gain is 0.125

X2 = 0 or X2 != 0 gain is 0.420

X2 = 1 or X2 != 1 gain is 0.091

X2 = 2 or X2 != 2 gain is 0.091

Question 4

Build the complete decision tree based on the given specifications and training set. The representation of the tree should adhere to the style used in the lecture notes of this course.

Answer: The best gain is at the split where X2 = 0 or X2 != 0 where the gain is 0.420. The only split on X1, when X1 = 0 or X1 = 1, is the next-best gain of 0.125. This is used to construct the decision tree below.

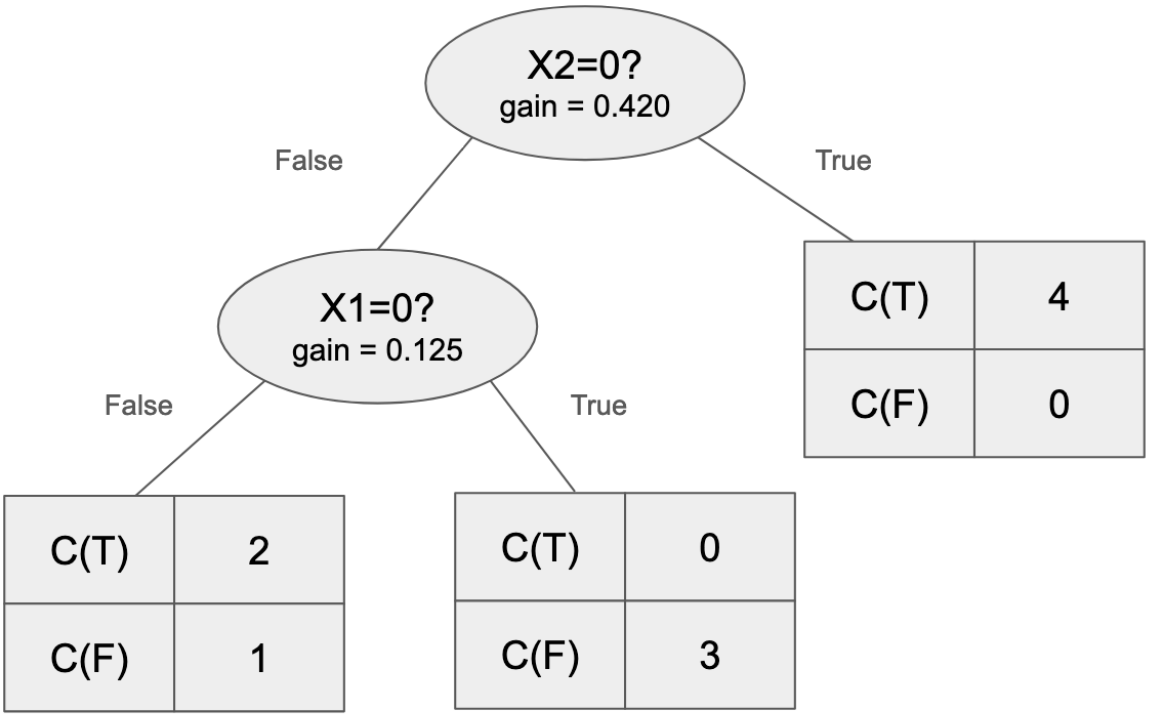


Figure 4: Decision Tree

Part 4 - Graduate Report

Introduction

For this part of the assignment, we will be using *Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico* [2] first introduced in 2019 [1]. The goal is to predict an individual's weight range which can fall into one of the seven categories listed in Table 1 below. Body Mass Index (BMI) is calculated using height and weight according to the formula:

$$BMI = weight/height^2$$

Note that Table 1 shows categories used by the dataset authors and they shouldn't be used as a ground truth. In addition, the dataset is balanced using synthetically generated data, to avoid having the majority falling under the normal weight category, which would throw the model off [2].

Target	Body Mass Index
Normal Weight	18.5 to 24.9
Overweight Level I	25.0 to 29.9
Overweight Level II	Not Provided
Obesity Type I	30.0 to 34.9
Insufficient Weight	< 18.5
Obesity Type II	35.0 to 39.9
Obesity Type III	> 40

Table 1: Obesity levels dataset targets

The authors stated that the goal of creating the dataset was to develop tools for predicting an individual's obesity level to create recommender systems for tracking and managing obesity [1]. To this effect, the dataset has sixteen features like habits, behaviours, physical characteristics, and some family history. Tables 2 and 3 below show a sample of instances from the dataset with descriptions for abbreviated features. Each row in Table 3 continues the same row from Table 2.

Gender	Age	Height	Weight	History	FAVC	FCVC	NCP
Female	21	1.62	64	yes	no	2	3
Female	21	1.52	56	yes	no	3	3
Male	23	1.80	77	yes	no	2	3
Male	27	1.80	87	no	no	3	3
Male	22	1.78	90	no	no	2	1

History: Has a family member suffered or suffers from being overweight?
FAVC: Do you eat high caloric food frequently?
FCVC: Do you usually eat vegetables in your meals?
NCP: Number of main meals per day (1 to 3)

Table 2: Obesity levels dataset sample features (Part 1)

CAEC	Smoke	CH2O	SCC	FAF	TUE	CALC	MTRANS
Sometimes	no	2	no	0	1	no	Public Tras
Sometimes	yes	3	yes	3	0	Sometimes	Public Trans
Sometimes	no	2	no	2	1	Frequently	Public Trans
Sometimes	no	2	no	2	0	Frequently	Walking
Sometimes	no	2	no	0	0	Sometimes	Public Trans

CAEC: Do you eat any food between meals?
CH2O: How much water do you drink daily?
SCC: Do you monitor the calories you eat daily?
FAF: How often do you have physical activity?
TUE: How much time do you use technological devices?
CALC: How often do you drink alcohol?
MTRANS: Which transportation do you usually use?

Table 3: Obesity levels dataset sample features (Part 2)

Methodology

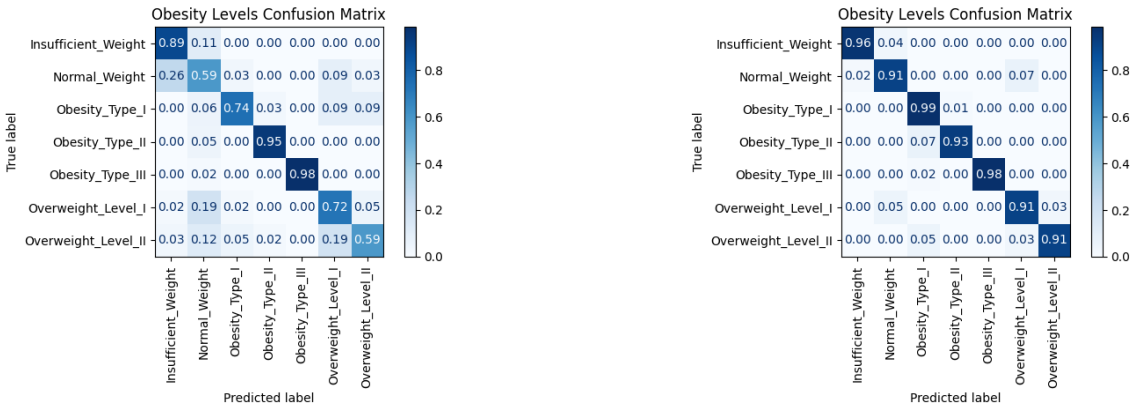
As required we will be using a decision tree classification model. A decision tree classifier expects numerical input. However, the dataset contains several categorical features such as Smoke and History: yes, no. Other categorical features are not binary such as Means of Transportation or Alcohol Consumption. We need to encode these categorical features which means they need to be converted into numerical values before we can train the model. Given that we have only 16, well-described, features, this can be accomplished manually. Another method is to use the function get_dummies() from the pandas library. Still, other options exist, like OneHotEncoder from the sklearn library. There are many ways we can pre-process and engineer features that fall outside this assignment's scope.

After pre-processing, the dataset is split into 80% training and 20% test (validation) instance sets using the train_test_split function from the sklearn model_selection library. To ensure a balanced model we set stratification to true. This produced two datasets with approximately 1,688 training and 423 test instances respectively having all target classes represented approximately corresponding to their overall distribution.

We constructed a decision tree classifier in two ways and compared the results. First, we constructed a decision tree "manually" with some best-guess parameters. Next, we used the GridSearchCV from the sklearn which performs an exhaustive search over a set of values we provided as possible parameters for the decision tree. The search for the best parameter combination can be further configured by specifying the type of score or scores to use when comparing performance, and the number of iterations, called folds, the number of threads, or jobs to run in parallel, among others.

Results

As expected the decision tree created using the exhaustive search performed better. The side-by-side confusion matrix comparison shown in Figure-5 clearly shows the advantage of using an exhaustive search.



(a) Without Exhaustive Search

(b) Best Exhaustive Search Classifier

Figure 5: Confusion Matrix Comparison

We observed that the performance of the classifier with optimized parameters has more balanced scores across all classes. The classification report comparison supports the same observation. Ta-

ble 4 shows the report for the non-optimized classifier. The ranges of values for all scores are wide and the overall scores are not great.

class	**precision**	**recall**	**f1-score**	**support**
Insufficient Weight	0.73	0.89	0.80	54
Normal Weight	0.52	0.59	0.55	58
Obesity Type I	0.90	0.74	0.81	70
Obesity Type II	0.95	0.95	0.95	60
Obesity Type III	1.00	0.98	0.99	65
Overweight Level I	0.66	0.72	0.69	58
Overweight Level II	0.76	0.59	0.66	58
accuracy	0.78	0.78	0.78	-
macro avg	0.79	0.78	0.78	423
weighted avg	0.79	0.78	0.78	423

Table 4: Classification Report For Non-Optimized Classifier

Table 5 shows the report for the optimized classifier. Here, the values across all scores are more consistent and higher, showing a much better overall performance.

class	**precision**	**recall**	**f1-score**	**support**
Insufficient Weight	0.98	0.96	0.97	54
Normal Weight	0.91	0.91	0.91	58
Obesity Type I	0.90	0.99	0.94	70
Obesity Type II	0.98	0.93	0.96	60
Obesity Type III	1.00	0.98	0.99	65
Overweight Level I	0.89	0.91	0.91	58
Overweight Level II	0.96	0.91	0.94	58
accuracy	0.95	0.95	0.95	-
macro avg	0.95	0.94	0.94	423
weighted avg	0.95	0.95	0.95	423

Table 5: Classification Report For Optimized Classifier

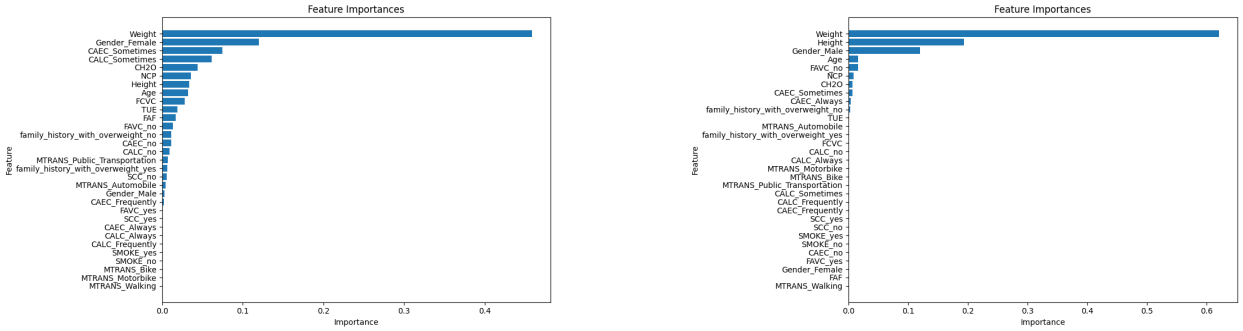
We also observe that the decision tree after the exhaustive search looks better when compared visually.

(a) Decision Tree Before Optimization

(b) Decision Tree After Optimization

Figure 6: Tree Depth

We also looked at the importance of each feature for the overall classification. It is not surprising that the weight feature is by far the most important predictor for both non-optimized and optimized classifiers. However, in the side-by-side comparison below, we still observe the difference across the other features between the two models. The optimized model found the path of least resistance which is also intuitive that age and gender along with weight are good predictors of an individual's weight classification.



(a) Without Exhaustive Search

(b) Best Exhaustive Search Classifier

Figure 7: Best Predictor Comparison

Conclusions and Future Research

When using decision tree classifiers performing an exhaustive search of a wide range of parameters leads to significantly more balanced and better results across all performance metrics.

We also observed that the best predictors by far are an individual's weight and height which is hardly surprising given that the targets are BMI ranges which are essentially weight relative to height. Statistically, males are also both taller and heavier than females on average, and taller and older individuals will be on average heavier. This almost makes the classification using this specific dataset impractical and unnecessary since it is rather obvious that having height and weight along with age statistical models will heavily rely on weight, height, and age, at the expense of the other features as our results show. This makes the rest of the features almost irrelevant which undermines the whole premise of this dataset. What's more interesting, is that the results show that taking into account behavioural features leads to worse predictions. In future research, we should look into more sophisticated datasets, models and methods that would be able to achieve similar results while more relying on behavioural, socio-economic, and demographic features. This could lead to

the development of more practical models for monitoring and helping with keeping the weight in a healthy range.

References

[1] F. M. Palechor and A. D. la Hoz Manotas. Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from colombia, peru and mexico. *Data in Brief*, 25, 2019.

[2] F. M. Palechor and A. D. la Hoz Manotas. Estimation of Obesity Levels Based On Eating Habits and Physical Condition . UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5H31Z.

[3] M. O. S. N. Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.