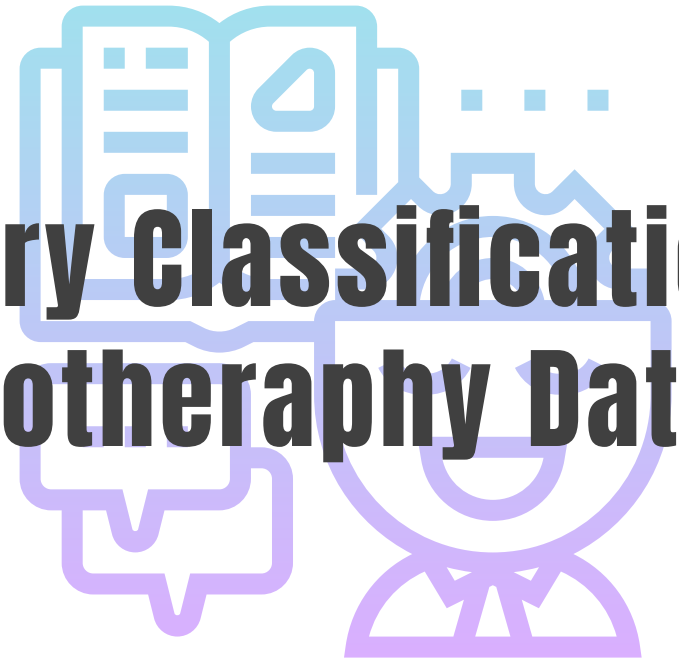


Binary Classification on Cryotheraphy DataSet

Table of Contents

Objective

Applying Binary Classification on cryotherapy dataset to classify whether the skin wart of a person can be cured with cryotherapy or not .

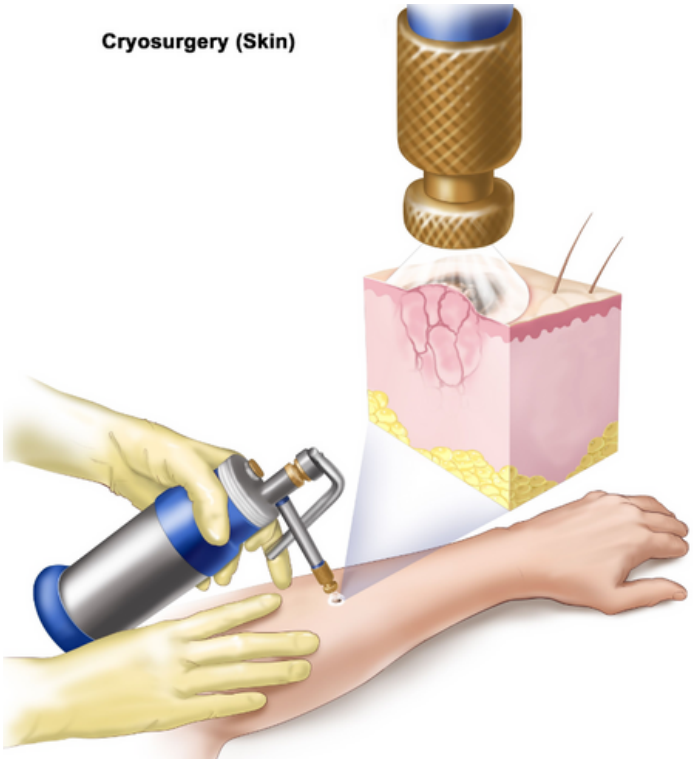
System Requirements

1.Python3 environment to be set up on the PC.
2.Libraries such as numpy, pandas, scikit-learn and graphviz should be installed.
3.The code can be run in the command terminal on systems with a python environment installed.

Introduction

(KRY-oh-THAYR-uh-pee) It is A procedure in which an extremely cold liquid or an instrument called a cryoprobe is used to freeze and destroy abnormal tissue. A cryoprobe is cooled with liquid nitrogen, nitrous oxide, or compressed argon gas. Cryotherapy may treat certain types of cancer and some conditions that may become cancer. Also called cryoablation and cryosurgery.

On the other hand, Warts are caused by various strains of human papillomaviruses. Different strains may cause warts in different parts of the body. Warts can be spread from one location on the body to another or from person to person by contact with the wart.
The main symptom is a fleshy, painless growth on the skin. Common areas affected include the hands, feet, and genitals.
Treatment may include topical medication and removal through medical procedures including Cryotheraphy.



Data Set Description

Database for cryotherapy

Information from the source: Fahime Khozeimeh, MD is the creator.

Univariate is a term used to describe a set of characteristics.

The whole set of patients with their skin infection area and days from which it is occurring is when treated by cryotherapy the result of this is encoded in this database. "Successful Treatment" is the target notion (i.e., true when cryotherapy can successfully treat warts on the skin).

6 attributes, each one tells about the infection of a particular patient

Attribute Information:

1.Sex {1=MALE , 2=FEMALE}
2.Age
3. Time(in days)
4.Number Of Warts
5.Type
6.Area

Class Information: {1= successful , 0 = not successfull }

Missing Attribute Values: None

Work Plan

To begin, we perform a preliminary examination of the data set using the following procedure:

1.The following libraries will be imported:

Numpy is used to do linear algebra and other mathematical calculations.

Pandas is used to read, select, slice, and change data sets.

Matplotlib and Graphviz are used to visualise and plot the model.Scikit-learn

(sklearn) is a Python library for dividing data, conducting classification, and producing model metrics.

2. Using the read function of the pandas library, we will read the data from cryotheraphy.csv.

3. Then we plot the data description and class distribution of the dataset.

4. After this we we perform analysis of our dataset by plotting graph for analyzing all the attributes.

5. We also try to infer all the information we can get from the plots.

We're now going to train 6 Decision Tree Model.

1.We visualize the decision tree, create a model, and use the model to generate predictions.

3. We then enumerate all the prediction to make a bigger model using bagging.

2.We will then calculate the trained model's assessment metrics.

We will justify the categorization method based on the outcomes. We offer our instincts and back them up with proof.

Preliminary Analysis Of Data

Importing libraries

```
[ ] import pandas as pd
    import numpy as np
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, ConfusionMatrixDisplay, f1_score
    from sklearn import tree
    import joblib
    from sklearn.tree import export_graphviz
    from six import StringIO
    from IPython.display import Image
    import pydotplus
    from matplotlib import pyplot as plt
```

Reading the dataset

```
[ ] dataset = pd.read_csv('Cryotherapy.csv')
```

```
[ ] dataset.head(100)
```

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
0	1	35	12.00	5	1	100	0
1	1	29	7.00	5	1	96	1
2	1	50	8.00	1	3	132	0
3	1	32	11.75	7	3	750	0
4	1	67	9.25	1	1	42	0
...	...	...	...	...	...	...	...
85	2	34	12.00	3	3	95	0
86	2	20	3.50	6	1	75	1
87	2	35	8.25	8	3	100	0
88	1	24	10.75	10	1	20	1
89	1	19	8.00	8	1	160	1

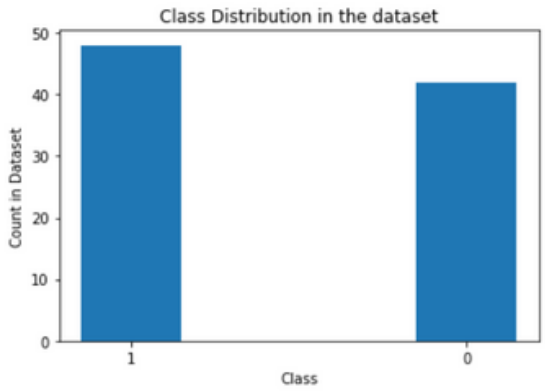
Data Description

```
description=dataset_copy.describe()
description.head()
```

	sex	age	Time	Number_of_Warts	Type	Area	Result_of_Treatment
count	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000	90.000000
mean	1.477778	28.600000	7.666667	5.511111	1.700000	85.833333	0.533333
std	0.502304	13.360852	3.406661	3.567155	0.905042	131.733153	0.501683
min	1.000000	15.000000	0.250000	1.000000	1.000000	4.000000	0.000000
25%	1.000000	18.000000	4.562500	2.000000	1.000000	20.000000	0.000000

Class Distribution of Dataset

```
df1 = dataset_copy['Result_of_Treatment'].value_counts()
X_sex = []
Y_sex = []
for key, value in df1.to_dict().items():
    X_sex.append(f"{key}")
    Y_sex.append(value)
plt.bar(X_sex, Y_sex, width=0.3)
plt.xlabel("Class")
plt.ylabel("Count in Dataset")
plt.title("Class Distribution in the dataset")
plt.show()
```

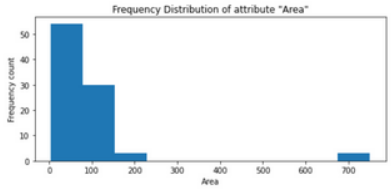


Influence of attributes on dataset

Now let's look at what the attributes say . The various graphs below show how each attribute help us to find information about the patients, the nature of disease and its treatment.

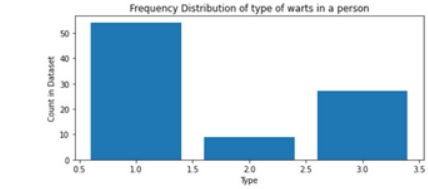
Each figure tells abut the distribution of various attributes in whole dataset.

```
[ ] plt.hist(dataset_copy['Area'])
    plt.ylabel('Frequency count')
    plt.xlabel('Area');
    plt.title('Frequency Distribution of attribute "Area"')
    plt.show()
```



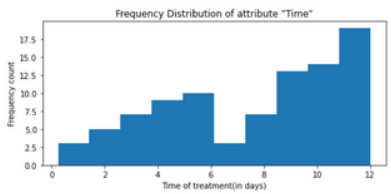
From the graph we can deduce that the infected region mostly lie between 0-200 unit sq.

```
[ ] df1 = dataset_copy['Type'].value_counts()
    X_Age = []
    Y_Age = []
    temp = []
    df1_dict = df1.to_dict()
    for i in sorted(df1_dict.keys()):
      temp.append(i)
    for key in temp:
        X_Age.append(key)
        Y_Age.append(df1_dict[key])
    plt.bar(X_Age, Y_Age)
    plt.xlabel("Type")
    plt.ylabel("Count in Dataset")
    plt.title("Frequency Distribution of type of warts in a person")
    plt.show()
```



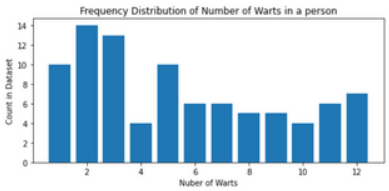
From the graph we can deduce that the most people are suffering with type 1 or 3 type of warts.

```
[ ] plt.hist(dataset_copy['Time'])
    plt.ylabel('Frequency count')
    plt.xlabel('Time of treatment(in days)');
    plt.title('Frequency Distribution of attribute "Time"')
    plt.show()
```



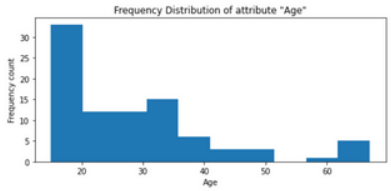
From the graph we are able to visualize amount of time required for treatment through cryotherapy of a person.

```
[ ] df1 = dataset_copy['Number_of_Warts'].value_counts()
    X_age = []
    Y_age = []
    for key, value in df1.to_dict().items():
        X_age.append(key)
        Y_age.append(value)
    plt.bar(X_age, Y_age)
    plt.xlabel("Nuber of Warts")
    plt.ylabel("Count in Dataset")
    plt.title("Frequency Distribution of Number of Warts in a person")
    plt.show()
```



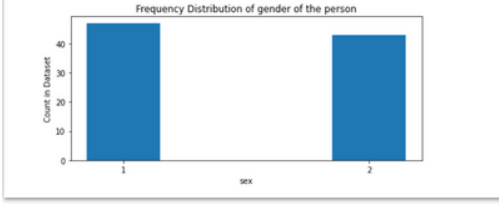
From the graph we come to know about how serious mostly the infection is(that is mostly the number of warts are less than 6 which tells it is a moderate level infection)

```
[ ] plt.hist(dataset_copy['age'])
    plt.ylabel('Frequency count')
    plt.xlabel('Age');
    plt.title('Frequency Distribution of attribute "Age"')
    plt.show()
```



From the graph we may predict that our patient is mostly below the age of 40.

```
df1 = dataset_copy['sex'].value_counts()
X_sex = []
Y_sex = []
for key, value in df1.to_dict().items():
    X_sex.append(f"{key}")
    Y_sex.append(value)
plt.bar(X_sex, Y_sex, width=0.3)
plt.xlabel("sex")
plt.ylabel("Count in Dataset")
plt.title("Frequency Distribution of gender of the person")
plt.show()
```



From the graph we can deduce that the the infection is not biased towards any gender.

Extracting attribute variable set and the class variable from the dataset

```
X_full = dataset.values[:, 0:6]
Y_full = dataset.values[:, 6:7]
```

Generating 6 Different Datasets using Bootstrap Method

```
def generateDataset(i):
  df1 = dataset.sample(n=len(dataset), axis='rows', replace=True)
  df1_X = df1.copy()
  df1_res = df1_X.pop("Result_of_Treatment")
  df1_sample = df1_X.sample(n=4, axis='columns')
  df1_sample["Result_of_Treatment"] = df1_res
  df1_sample.to_csv(f'Data_Sample_{i}.csv', header=True, index=False)
  print(f"Data_Sample_{i} saved")
```

```
for i in range(0, 6):
  generateDataset(i+1)
```

Dataset Generated Using Bootstrap Method

```
[ ]
data = pd.read_csv(f'Data_Sample_{1}.csv')
data.head(len(data))
```

	sex	Time	Area	Number_of_Warts	Result_of_Treatment
0	2	10.50	35	5	0
1	1	5.50	6	5	1
2	2	9.50	72	5	0
3	1	9.25	42	1	0
4	2	10.75	35	5	0
...	...	...	...	...	...
85	1	2.75	20	3	0
86	2	8.75	6	2	0
87	1	11.00	8	2	0
88	1	8.75	132	11	0
89	2	12.00	100	5	0

90 rows × 5 columns

```
[ ]  data = pd.read_csv(f'Data_Sample_{2}.csv')
     data.head(len(data))
```

	Time	Number_of_Warts	sex	Type	Result_of_Treatment
0	4.75	3	1	1	1
1	9.25	1	1	1	0
2	8.25	8	2	3	0
3	5.75	1	2	1	1
4	8.50	1	2	2	1
...	...	...	...	...	...
85	2.75	3	1	3	0
86	12.00	1	1	3	0
87	11.00	2	2	1	0
88	3.75	11	1	1	1
89	9.50	4	2	3	0

90 rows × 5 columns

```
[ ]  data = pd.read_csv(f'Data_Sample_{3}.csv')
     data.head(len(data))
```

	sex	Number_of_Warts	Area	age	Result_of_Treatment
0	1	11	20	67	0
1	2	6	160	19	1
2	2	6	8	36	0
3	1	5	6	27	0
4	1	2	10	17	0
...	...	...	...	...	...
85	1	3	20	63	0
86	1	7	72	23	0
87	2	2	20	41	1
88	2	10	70	15	1
89	1	3	6	20	1

90 rows × 5 columns

Dataset Generated Using Bootstrap Method

```
data = pd.read_csv(f'Data_Sample_{4}.csv')
data.head(len(data))
```

	age	Time	Area	sex	Result_of_Treatment
0	41	8.00	20	1	1
1	17	5.25	63	1	1
2	15	6.00	30	2	1
3	15	3.75	70	2	1
4	35	8.25	100	2	0
...	...	...	...	...	...
85	24	9.50	20	1	1
86	41	7.75	20	2	1
87	15	10.50	30	1	1
88	40	9.75	80	2	0
89	34	12.00	150	1	0

90 rows × 5 columns

```
[ ] data = pd.read_csv(f'Data_Sample_{5}.csv')
    data.head(len(data))
```

	Number_of_Warts	Time	sex	age	Result_of_Treatment
0	7	10.50	1	27	0
1	2	8.75	2	27	0
2	3	2.00	2	15	1
3	11	2.00	1	15	1
4	12	1.50	2	15	1
...	...	...	...	...	...
85	8	4.50	2	18	1
86	3	5.00	1	20	1
87	2	11.00	1	36	0
88	5	10.50	2	21	0
89	9	8.00	2	19	1

90 rows × 5 columns

```
[ ] data = pd.read_csv(f'Data_Sample_{6}.csv')
    data.head(len(data))
```

	age	Number_of_Warts	Area	Type	Result_of_Treatment
0	50	11	132	3	0
1	67	7	42	1	0
2	19	9	160	1	1
3	19	9	160	1	1
4	40	9	80	2	0
...	...	...	...	...	...
85	35	5	100	1	0
86	22	2	70	1	1
87	20	3	6	1	1
88	67	7	42	1	0
89	35	5	100	1	0

90 rows × 5 columns

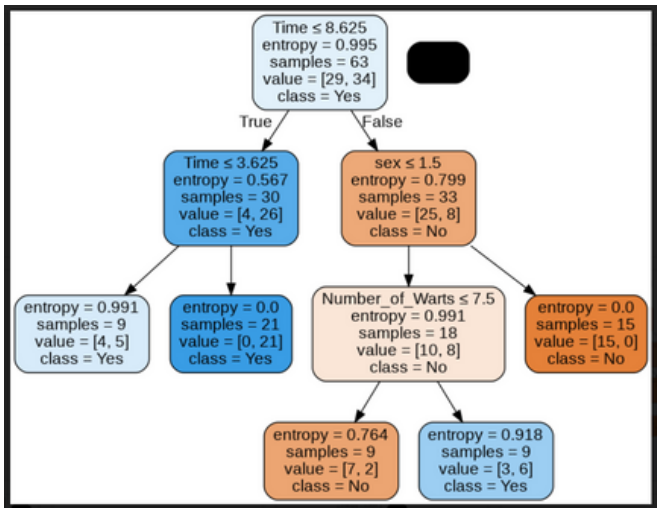
Training 6 Decision Tree Classifier With The Generated Datasets and Printing the Decision Tree Generated by Each Classifier

```
[ ]  def trainDT(i, X_train, Y_train, X_test, Y_test, atr_list):
       clf = DecisionTreeClassifier(criterion='entropy', max_depth=10, min_samples_leaf=5)
       clf = clf.fit(X_train,Y_train)
       Y_pred = clf.predict(X_test)
       print(f"Accuracy score for Dataset numbered {i} is {accuracy_score(Y_test, Y_pred)*100}%")
       joblib.dump(clf, f"DecisionTree{i}.pkl")
       print(f"DecisionTree{i} model saved")
       dot_data = StringIO()
       export_graphviz(clf, out_file=dot_data,
                       filled=True, rounded=True,
                       special_characters=True, feature_names=atr_list, class_names=['No','Yes'])
       graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
       graph.write_png(f'DecisionTree{i}.png')
       Image(graph.create_png())
       print(f"DecisionTree{i} visual saved\n")
```

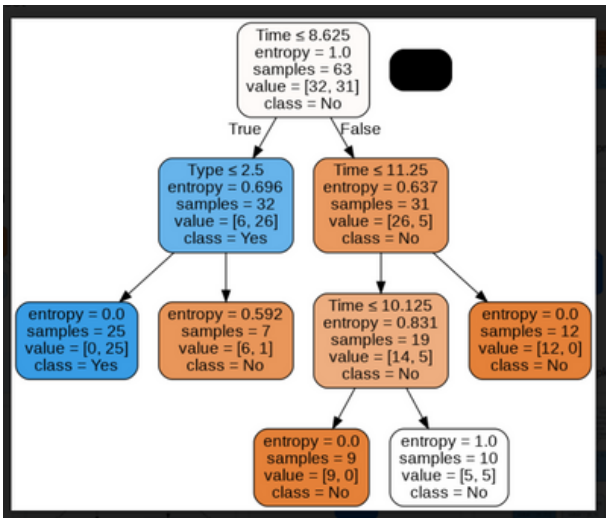
```
for i in range(0, 6):
  dataset_temp = pd.read_csv(f'Data_Sample_{i+1}.csv')
  atr_list = []
  for col in dataset_temp.columns:
    atr_list.append(col)
  atr_list.pop()
  X = dataset_temp.values[:, 0:4]
  Y = dataset_temp.values[:, 4:5]
  X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
  trainDT(i+1, X_train, Y_train, X_test, Y_test, atr_list)
```

Splitting the data set randomly into training data and test data with a 70-30 ratio

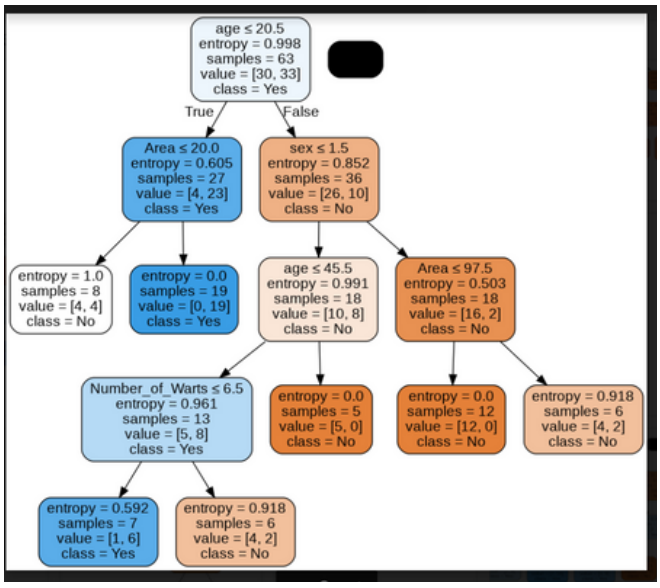
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
trainDT(i+1, X_train, Y_train, X_test, Y_test, atr_list)
```



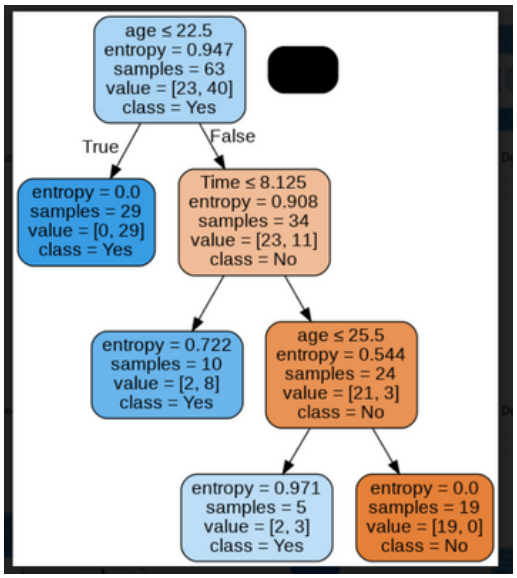
Decision Tree 1



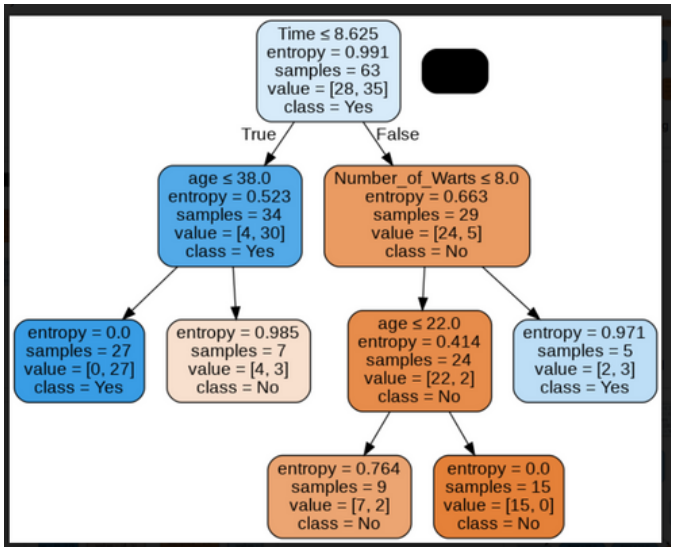
Decision Tree 2



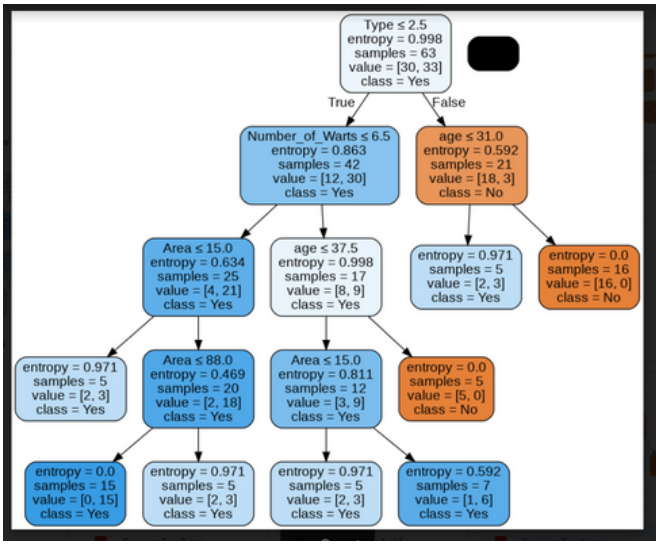
Decision Tree 3



Decision Tree 4



Decision Tree 5



Decision Tree 6

Accuracy of all the generated classifiers

```
Accuracy score for Dataset numbered 1 is 92.5925925925926%
DecisionTree1 model saved
DecisionTree1 visual saved

Accuracy score for Dataset numbered 2 is 96.29629629629629%
DecisionTree2 model saved
DecisionTree2 visual saved

Accuracy score for Dataset numbered 3 is 70.37037037037037%
DecisionTree3 model saved
DecisionTree3 visual saved

Accuracy score for Dataset numbered 4 is 81.48148148148148%
DecisionTree4 model saved
DecisionTree4 visual saved

Accuracy score for Dataset numbered 5 is 81.48148148148148%
DecisionTree5 model saved
DecisionTree5 visual saved

Accuracy score for Dataset numbered 6 is 85.18518518518519%
DecisionTree6 model saved
DecisionTree6 visual saved
```

Making Predictions using the generated classifiers through bagging method

```
[ ]  def predictSingle(X):
       result0 = 0
       result1 = 0
       X = [[X[0], X[2], X[5], X[3]],
               [X[2], X[3], X[0], X[4]],
               [X[0], X[3], X[5], X[1]],
               [X[1], X[2], X[5], X[0]],
               [X[3], X[2], X[0], X[1]],
               [X[1], X[3], X[5], X[4]]]
       for i in range(0, 6):
         clf = joblib.load(f'DecisionTree{i+1}.pkl')
         Y = clf.predict([X[i]])
         Y = Y[0]
         if Y == 0:
           result0 += 1
         else:
           result1 += 1
       if result0 > result1:
         return 0
       else:
         return 1
```

```
[ ]  print(predictSingle([1, 67, 10, 7, 1, 42]))
```

```
0
```

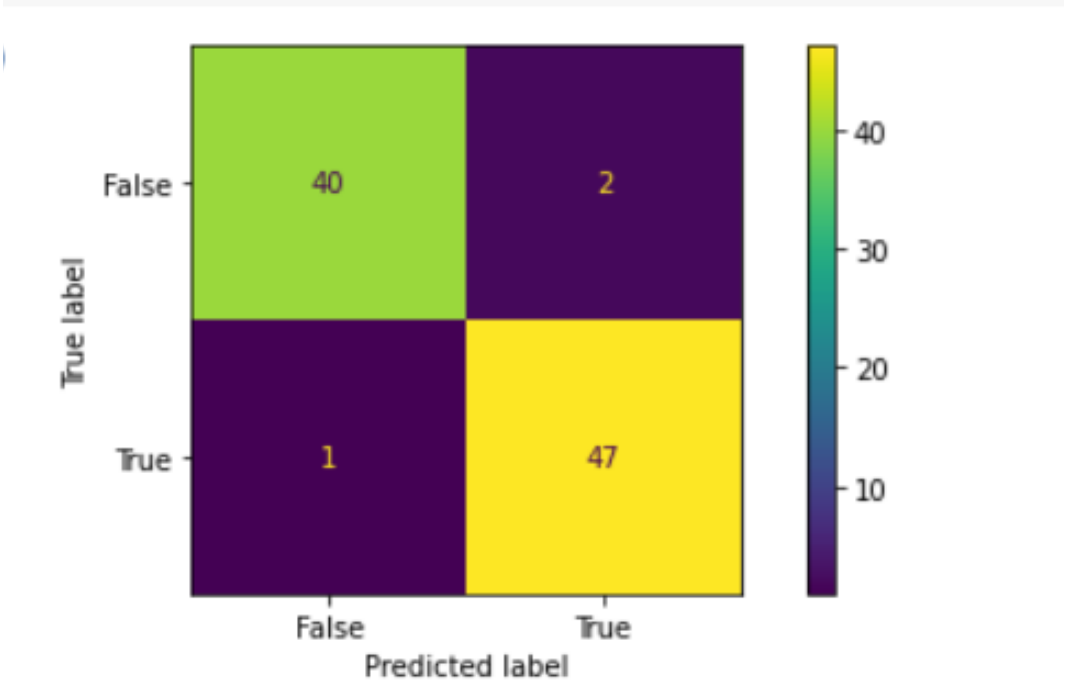
Computing the evaluation metrics of the trained model

```
[ ]  print(f'Accuracy of the model is {accuracy_score(Y_full, np.array(Y_pred))}')
     print(f'Precision of the model is {precision_score(Y_full, np.array(Y_pred))}')
     print(f'Recall of the model is {recall_score(Y_full, np.array(Y_pred))}')
     print(f'F1-Score of the model is {f1_score(Y_full, np.array(Y_pred))}')
```

```
Accuracy of the model is 0.9666666666666667
Precision of the model is 0.9591836734693877
Recall of the model is 0.9791666666666666
F1-Score of the model is 0.9690721649484536
```

Confusion Matrix

```
[ ]  cm_display = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [False, True])
     cm_display.plot()
     plt.show()
```



Justification for the classification algorithm

Initial Intuition

The challenge we're dealing with necessitates projecting good outcomes for the first participant. Because they can assess the outcomes of all potential options, decision trees are the greatest prediction model. You may, for example, need to repeat the procedure. The dataset includes categorical characteristics, which allow the decision tree model to analyse categorical variables without having to use hot coding. Decision trees need less work to prepare the data during preprocessing than other techniques. The decision tree provides a framework for assessing the worth and potential of each conceivable decision result, allowing decision-makers to make well-informed choices. The decision tree, unlike other decision models, makes all viable choices clear and records each choice to completion in a single display, making it simple to evaluate alternative options. When compared to other modelling methodologies, it needs less data purification. In light of the decision tree algorithm's benefits, we decided to move on.

Evidence supporting our intuition

We trained the model on training datasets using various classification algorithms. These are the results found when the model made predictions about the test dataset.

Comparison Table

Model	**Accuracy (%)**	**Precision(%)**	**Recall(%)**
Decision Tree	96.67	95.91	96.8
Naïve Bayes	72.40	73.38	90.4
SVM	94.79	98.32	93.6

This data is consistent with the first intuition as it shows that the decision tree algorithm provides the most accurate predictions and the best recognition values compared to the naive Bayes classifier and the SVM classification algorithm.

References

- https://towardsdatascience.com
- https://graphviz.readthedocs.io/en/stable/manual.html
- https://matplotlib.org/3.3.3/contents.html
- https://stackoverflow.com/
- scikit-learn: machine learning in Python — scikit-learn 0.23.2 documentation
- https://pandas.pydata.org/docs/
- https://numpy.org/doc/

- Dataset link that we used in this project :
  https://archive.ics.uci.edu/ml/datasets/Cryotherapy+Dataset+