COLLEGE OF BASIC AND APPLIED SCIENCES
SCHOOL OF PHYSICAL AND MATHEMATICAL SCIENCES
BSC COMPUTER SCIENCE

Report, Coursework1: "PREDICTIVE MODELING APPROACH USING DECISION TREES"

BY

IBRAHIM ANYARS SAFIANU

10826754

Table of Contents

DEFINITION OF TERMS

Decision Tree is a popular and intuitive machine learning algorithm that is widely used in classification and regression tasks.

Kappa Statistics is a measure of inter-rater agreement or reliability for categorical or nominal data. It quantifies the agreement between two or more raters who classify items into mutually exclusive categories.

Mean Absolute Error (MAE) is a common metric used to measure the average magnitude of errors in a regression model. It quantifies the average absolute difference between the predicted values and the actual values.

Root Mean Squared Error (RMSE) is a commonly used metric for evaluating the performance of a regression model. RMSE measures the average magnitude of the residuals or errors between the predicted values and the actual values, considering the square of the differences.

Relative Absolute Error (RAE) is a metric used to evaluate the accuracy of a regression model. It measures the average absolute difference between the predicted values and the actual values, relative to the average absolute difference between the actual values and their mean.

Relative Root Squared Error (RRSE) is a measure of the accuracy of a regression model. It is calculated by taking the square root of the ratio of the total squared error to the total squared error of a simple model, such as the mean of the actual values.

Validation dataset is an independent subset of data used to fine tune model during training to evaluate model performance.

Training dataset is an unseen subset of dataset used to evaluate the final performance of the trai ned model.

Testing dataset is the largest subset of data used in training a machine learning model.

PART ONE

CLASSIFICATION: USING VERTERAL COLUMN DATASETS FROM UCI REPOSITORY AND THE WAIKATO ENVIRONMENT FOR KNOWLEDGE ANALYSIS (WEKA) TOOL

Introduction

The objective of this project is to implement and evaluate a predictive modeling approach based on decision trees using the wine and vertebral column datasets available at the UCI repository. The http://archive.ics.uci.edu/datasets is a well-known repository of datasets in the field of machine learning.

The vertebral column dataset is composed of 2 Attribute-Relation File Format (arff) files, column_2C_weka.arff and column_2C_weka.arff, with measurements of seven 7 attributes each of whether a person suffers from hernia (a medical condition that occurs when an organ or tissue protrudes through an abnormal opening or weak spot in the surrounding muscles or connective tissue), Spondylolisthesis (a spinal condition in which one vertebra slips forward over the vertebra below it), or normal. There are 7 attributes that includes pelvic_incidence, pelvic_tilt, lumbar_lordosis_angle, sacral_slope, pelvic_radius, degree_spondylolisthesis, and class. It also has 310 records.

Data Partitioning:

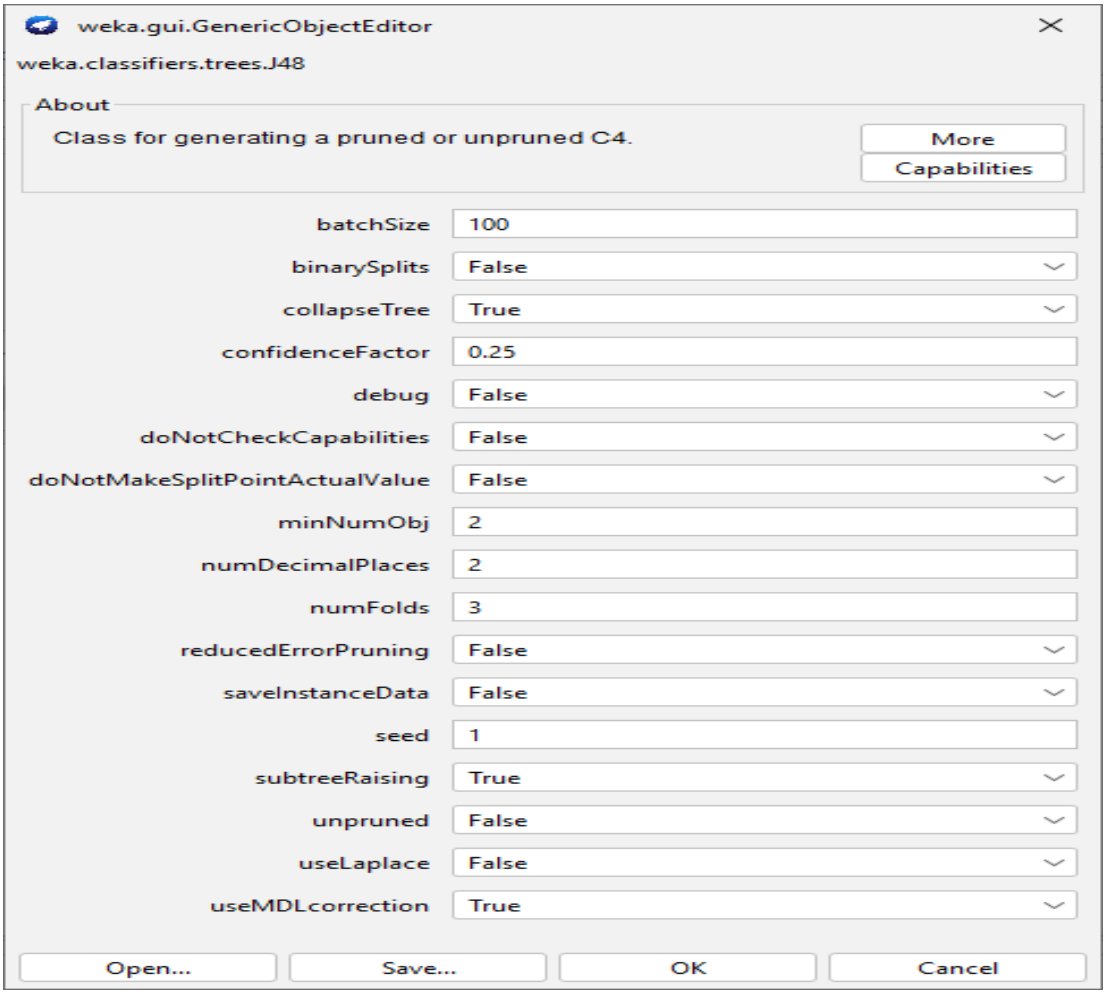
To begin the project, for the vertebral column dataset the column_3C_weka was used. The dataset was partitioned into two subsets: a training set and a validation set. The training set was used to construct the decision trees, while the validation was used to evaluate their performance.

Decision Tree Construction:

Different decision trees were constructed based on various partitions of the dataset. Gini index was the preferred splitting criterion used for tree construction with default parameters. The parameters such as maximum depth or minimum number of samples required to split a node should be specified for each decision tree.

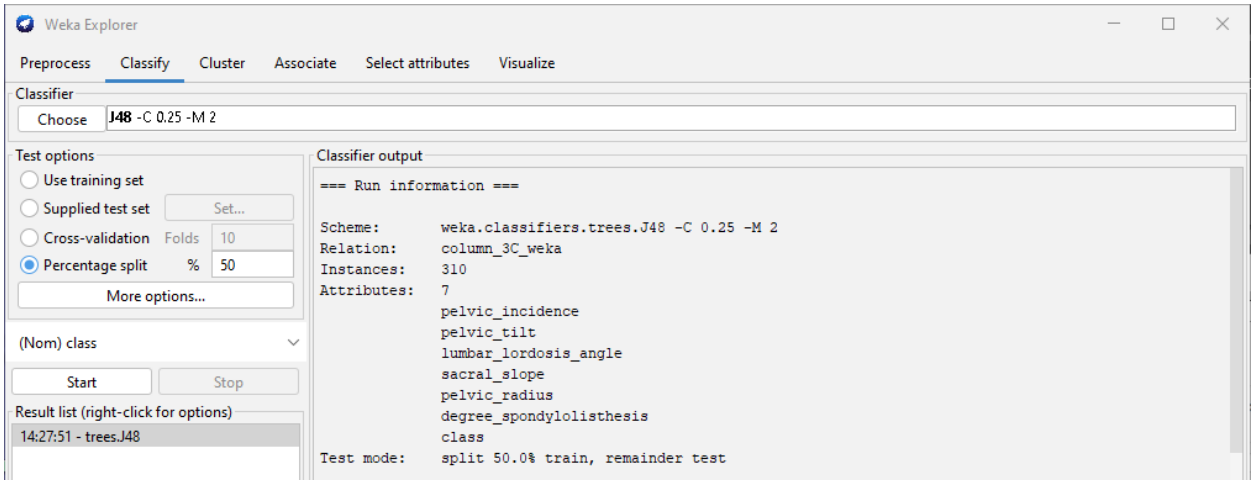
Constructing Decision Trees:

The decision tree constructed were of the following percentage split or partition for training and the rest for testing: 50%, 60%, and 70%. The J48 model in WEKA was used, the spitting criteria and parameters were kept constant using Gini index with default parameters.

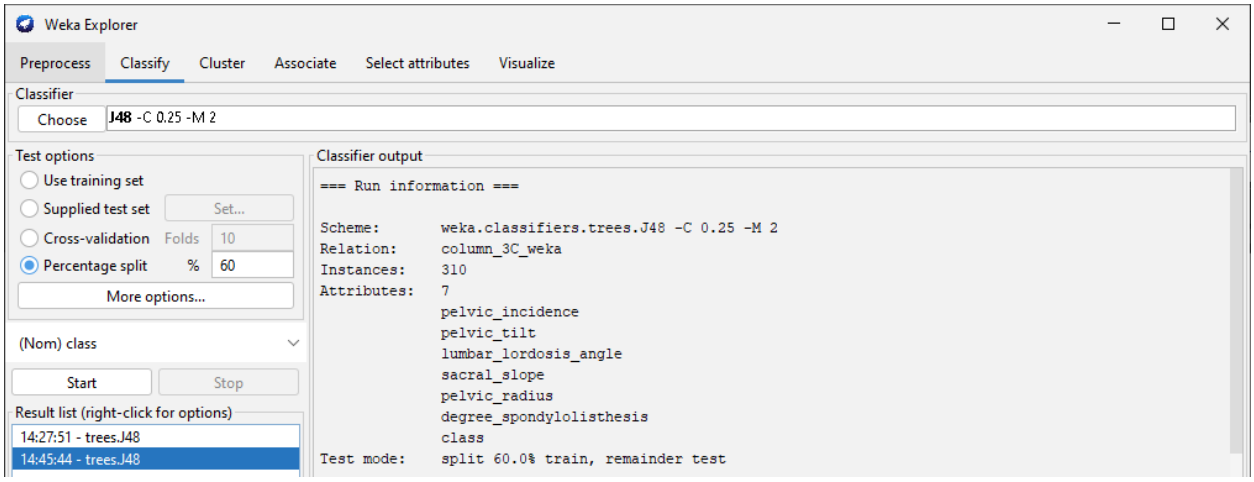


Default parameters: confidence factor 0.25, min number of instance parameters 2.

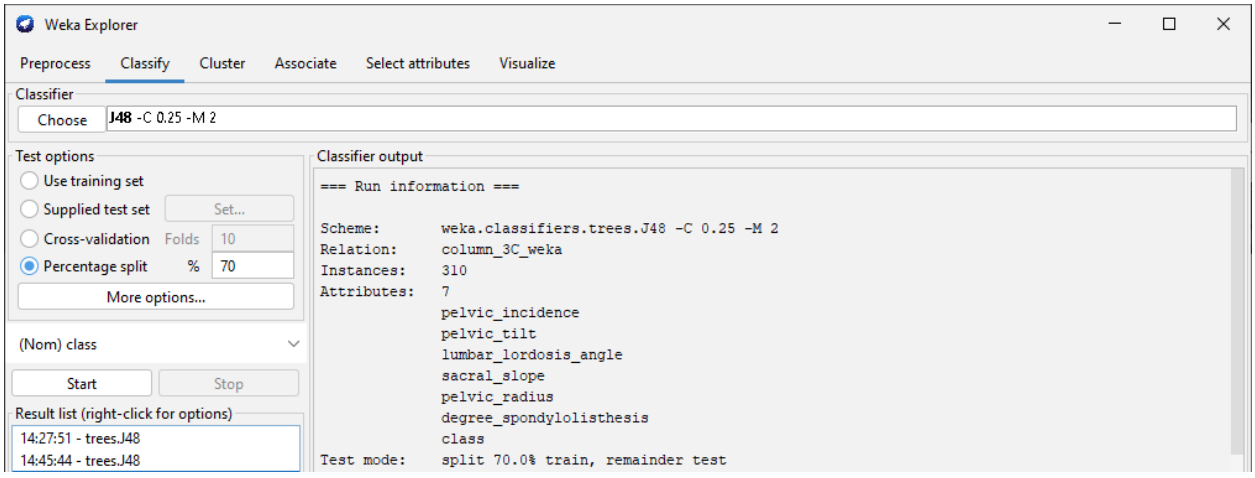
i) Decision tree 1: 50%:50% split train vs validation



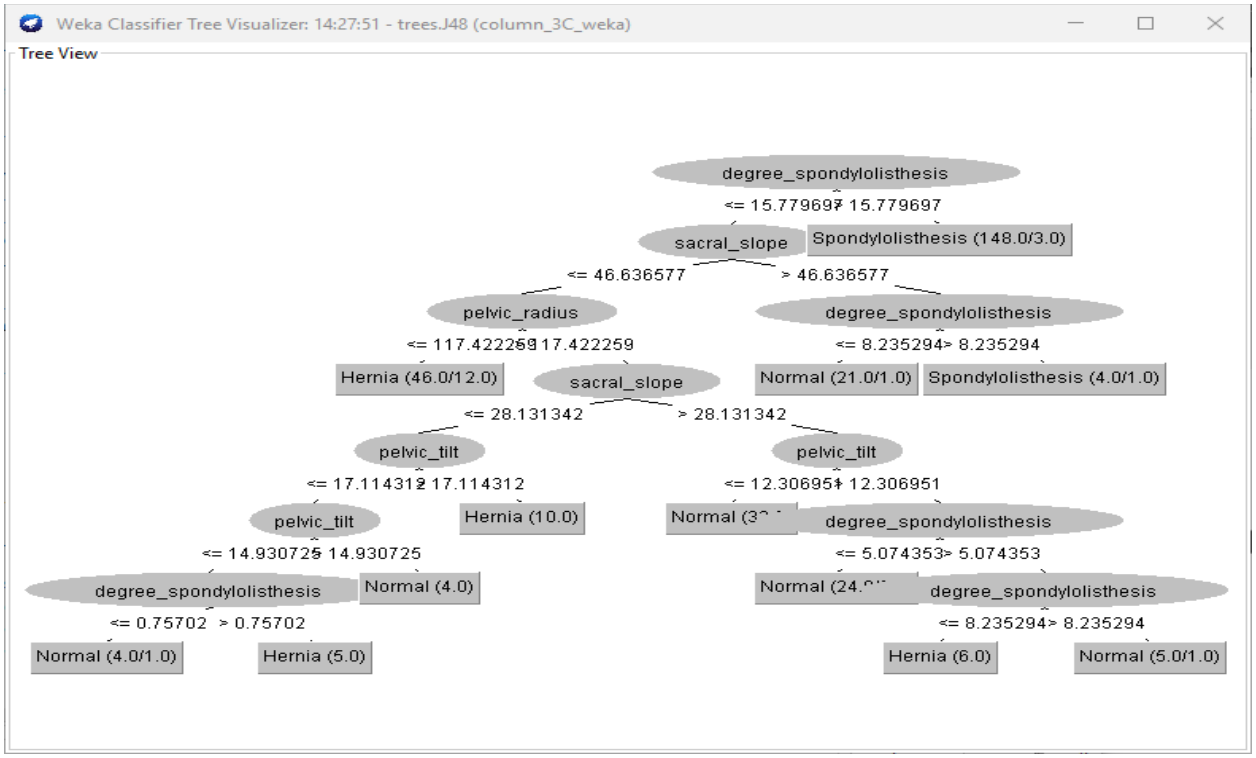
ii) Decision tree 2: 60%: 40% split



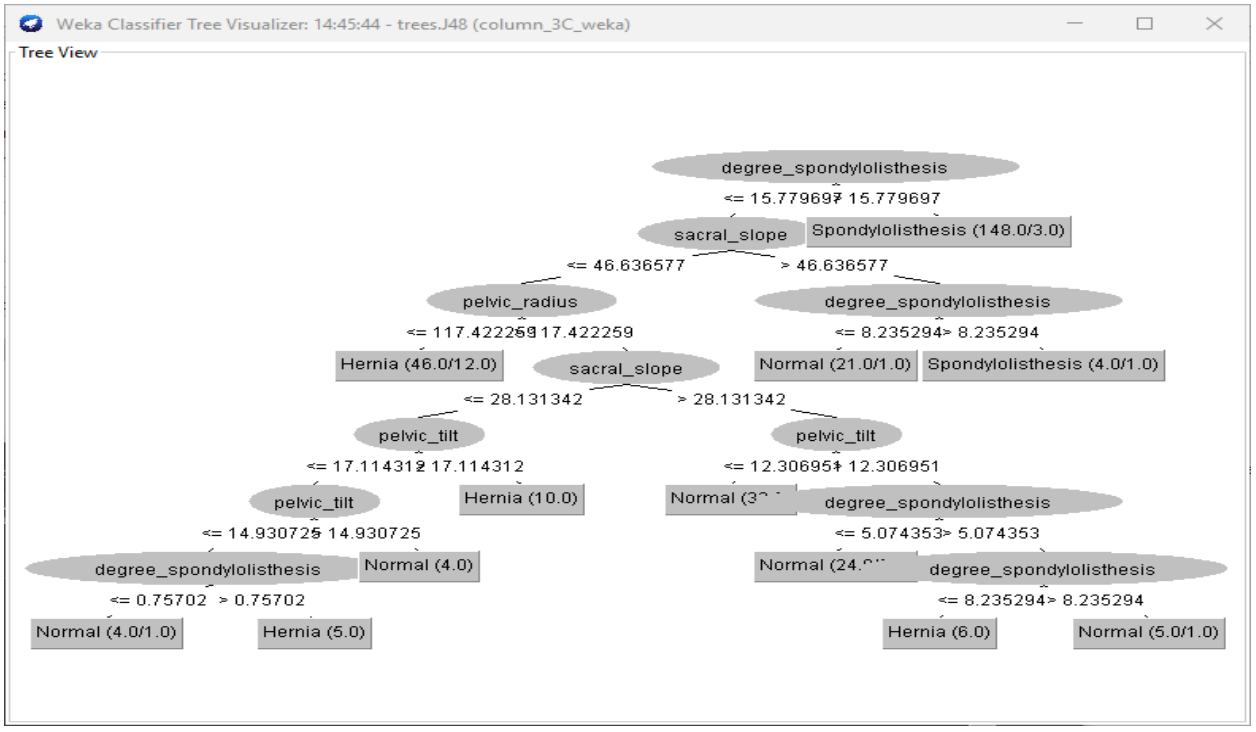
iii) decision tree 3: 70%: 30%



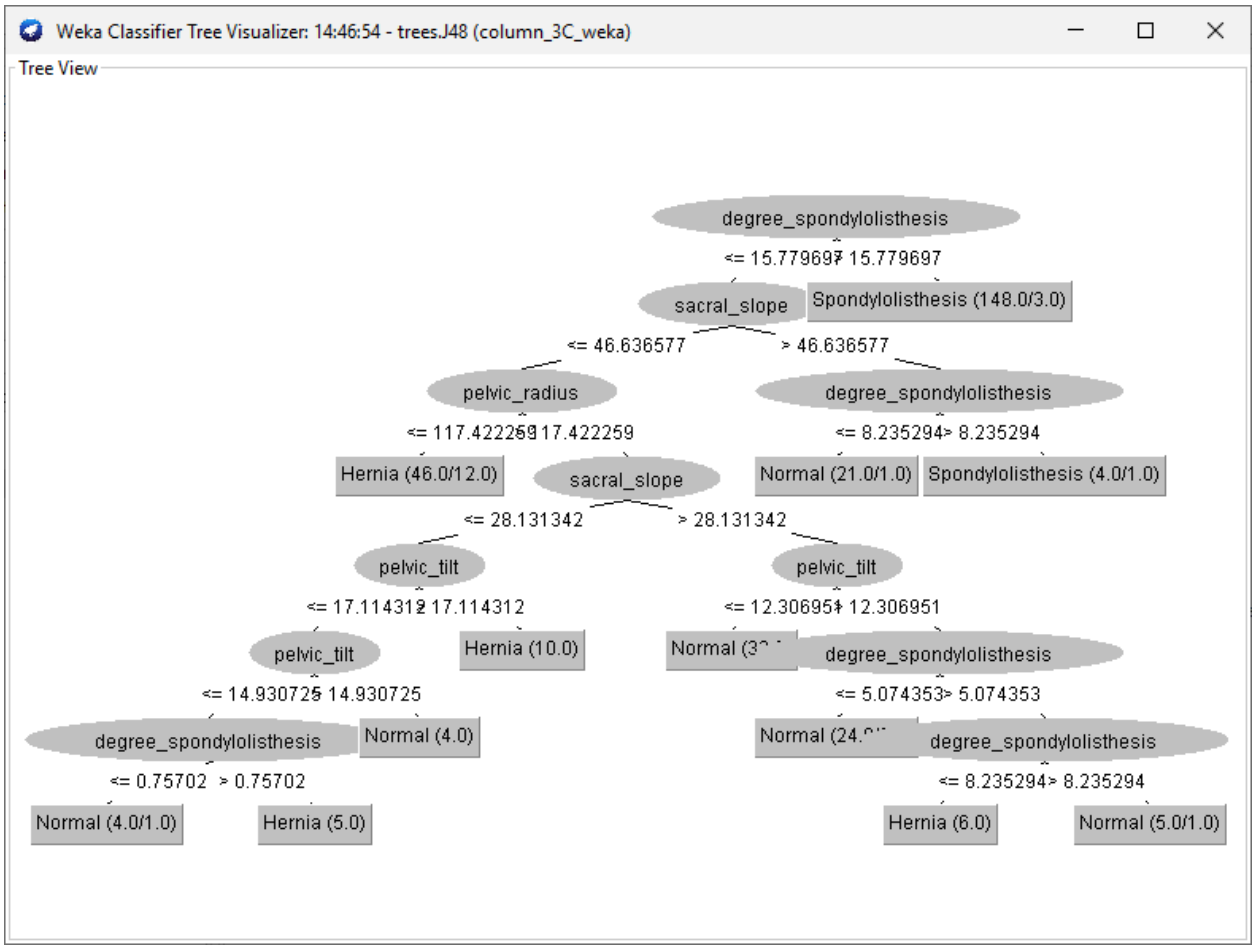
Comparing Tree Structures and Performance:



Tree view of 50% split



Tree view of the 60% split



Tree view of 70% split

Performance table

Decision tree	Keppa statistics	MAE	RMSE	RAE	RRSE	Accuracy
50:50 split	0.6905	0.1488	0.3484	35.5485	75.5638	80.6452
60:40 split	0.6763	0.1625	0.3487	38.6135	75.305	79.8387
70:30 split	0.6905	0.153	0.3218	36.2567	69.1727	80.6452

It can be seen that both the decision tree with 50:50 split has almost similar score to the 70:30 split with accuracy of 80.6452 while the 60:40 split record 79.8287 accuracy.

Paired of classes that can be confused based on accuracy of the classes:

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.600    0.106    0.643      0.600    0.621      0.506    0.738     0.462     Hernia
               0.968    0.016    0.984      0.968    0.976      0.952    0.987     0.979     Spondylolisthesis
               0.645    0.151    0.588      0.645    0.615      0.480    0.751     0.489     Normal
Weighted Avg.  0.798    0.072    0.802      0.798    0.800      0.726    0.868     0.731
```

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.625    0.101    0.682      0.625    0.652      0.539    0.880     0.663     Hernia
               1.000    0.063    0.938      1.000    0.968      0.938    0.948     0.938     Spondylolisthesis
               0.625    0.116    0.652      0.625    0.638      0.516    0.841     0.591     Normal
Weighted Avg.  0.806    0.086    0.798      0.806    0.801      0.726    0.903     0.777
```

```
=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.625    0.101    0.682      0.625    0.652      0.539    0.880     0.663     Hernia
               1.000    0.063    0.938      1.000    0.968      0.938    0.948     0.938     Spondylolisthesis
               0.625    0.116    0.652      0.625    0.638      0.516    0.841     0.591     Normal
Weighted Avg.  0.806    0.086    0.798      0.806    0.801      0.726    0.903     0.777
```

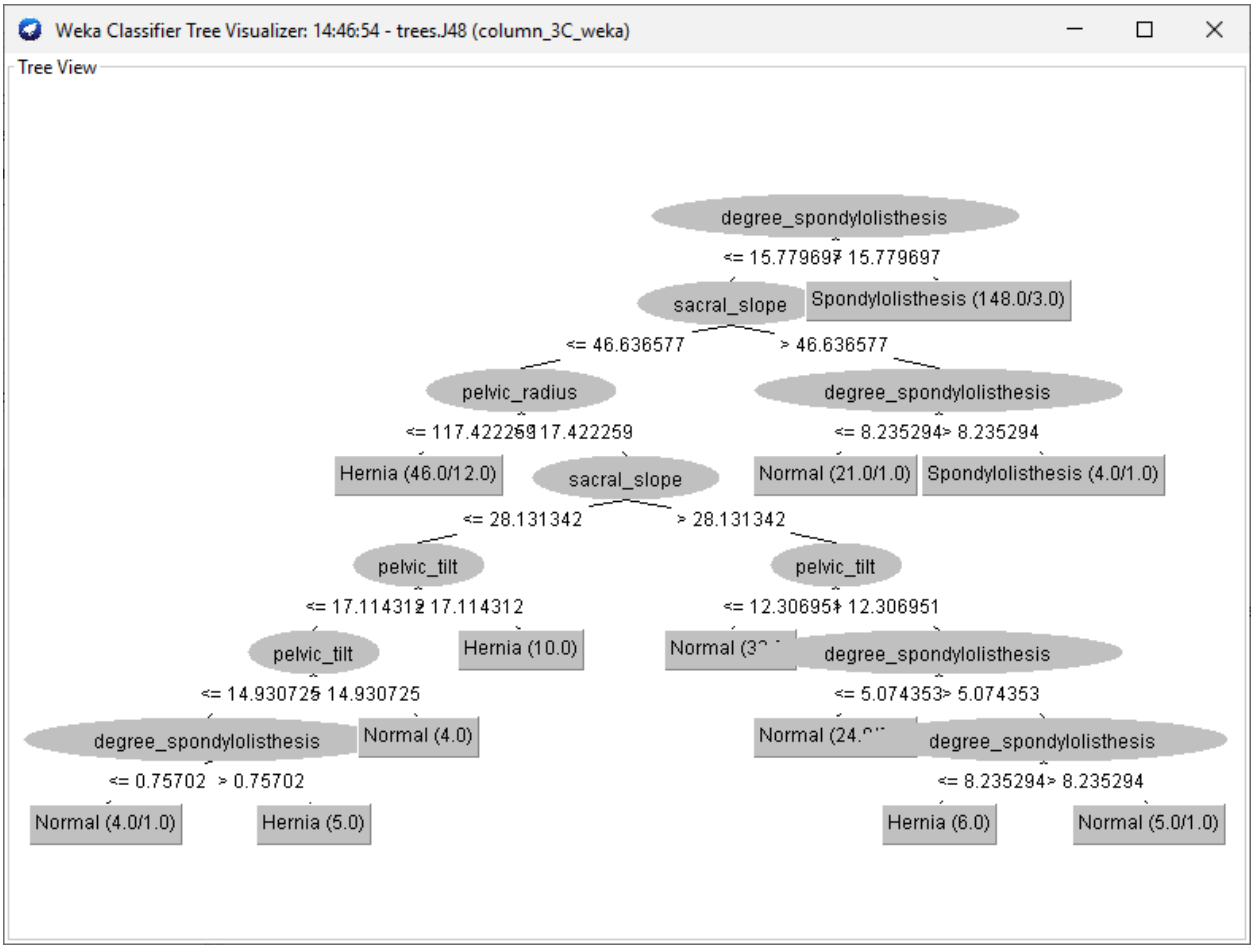
Looking at the detailed accuracy by class above for the 50%, 60% and 70% split respectively, Hernia and Normal can be confused with each other across the true positive, false positive, precision, recall f1-score, ROC Area, etc., score—especially in the case of the 70% split.

This can be further confirmed by the confusion matrix for 70% split

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  1  8 |  a = Hernia
  0 45  0 |  b = Spondylolisthesis
  7  2 15 |  c = Normal
```

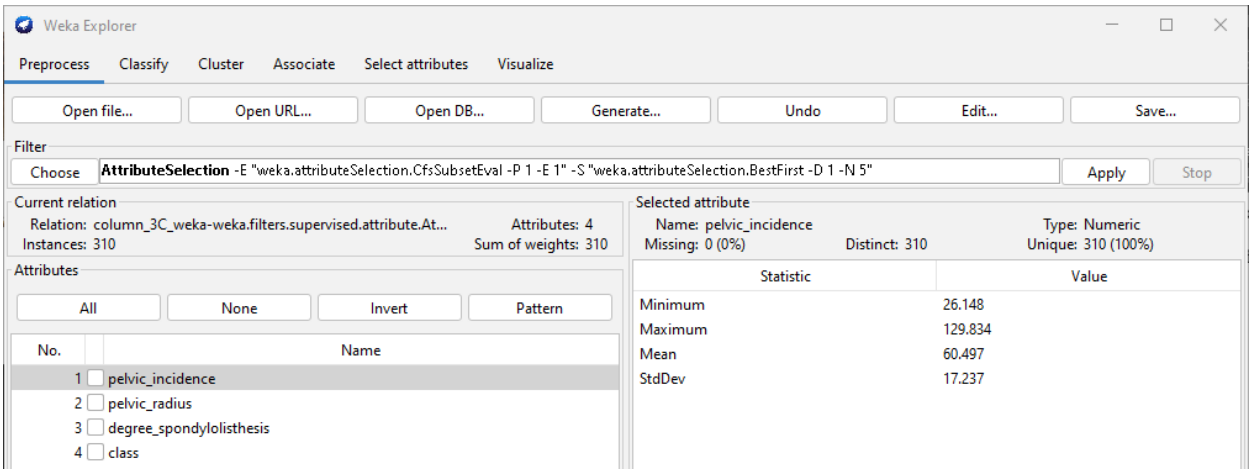
What led to misclassification.

Using the confused class for the 70% train 30% test, it can be seen that indeed Normal and Hernia can be confused which each other.

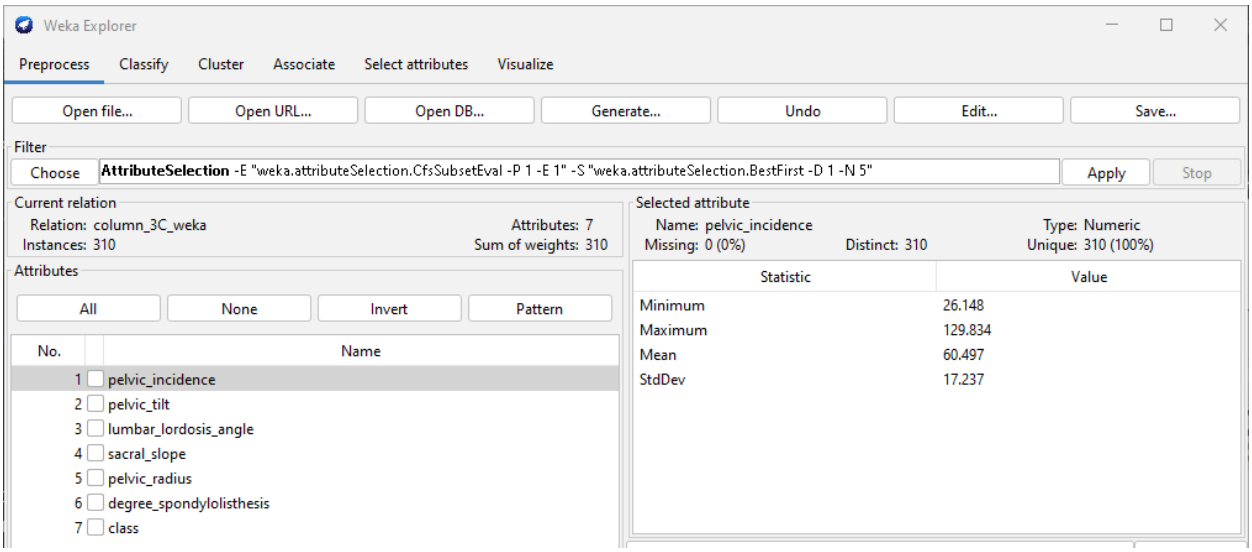


Attribute Subset Comparison:

Selecting attribute subset for the 70% train, 30% validation dataset decision tree, in WEKA, under Preprocessing tab, choose filters > supervised> attribute > attribute selection and then apply. The 7 attributes get reduced to 4.



Attributes after applying attribute selection.



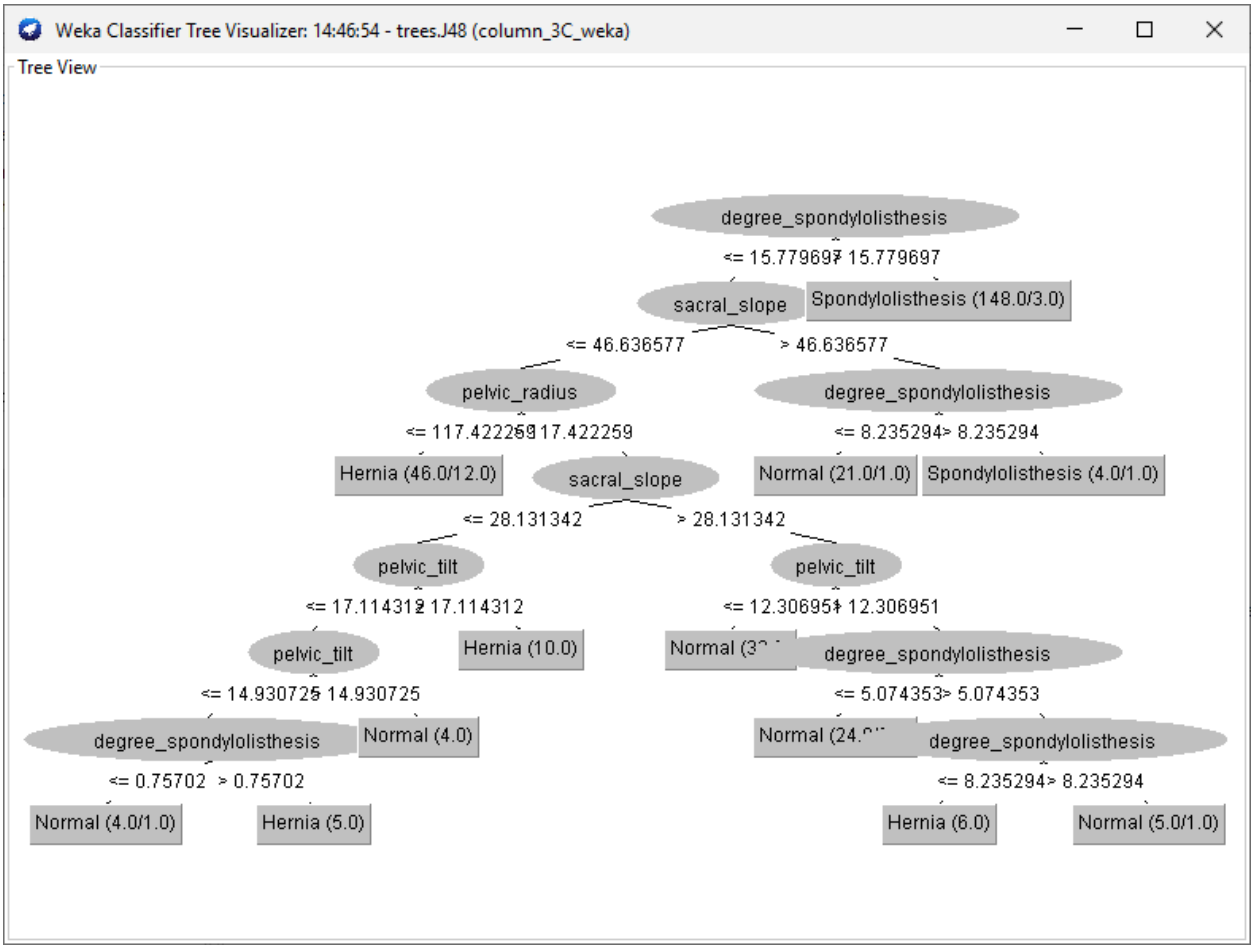
Attributes before.

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     column_3C_weka-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSelection.CfsSubsetEval -P 1 -E 1-Sweka.attributeSelection.BestFirst -D 1 -N 5
Instances:    310
Attributes:   4
              pelvic_incidence
              pelvic_radius
              degree_spondylolisthesis
              class
Test mode:    split 70.0% train, remainder test
```

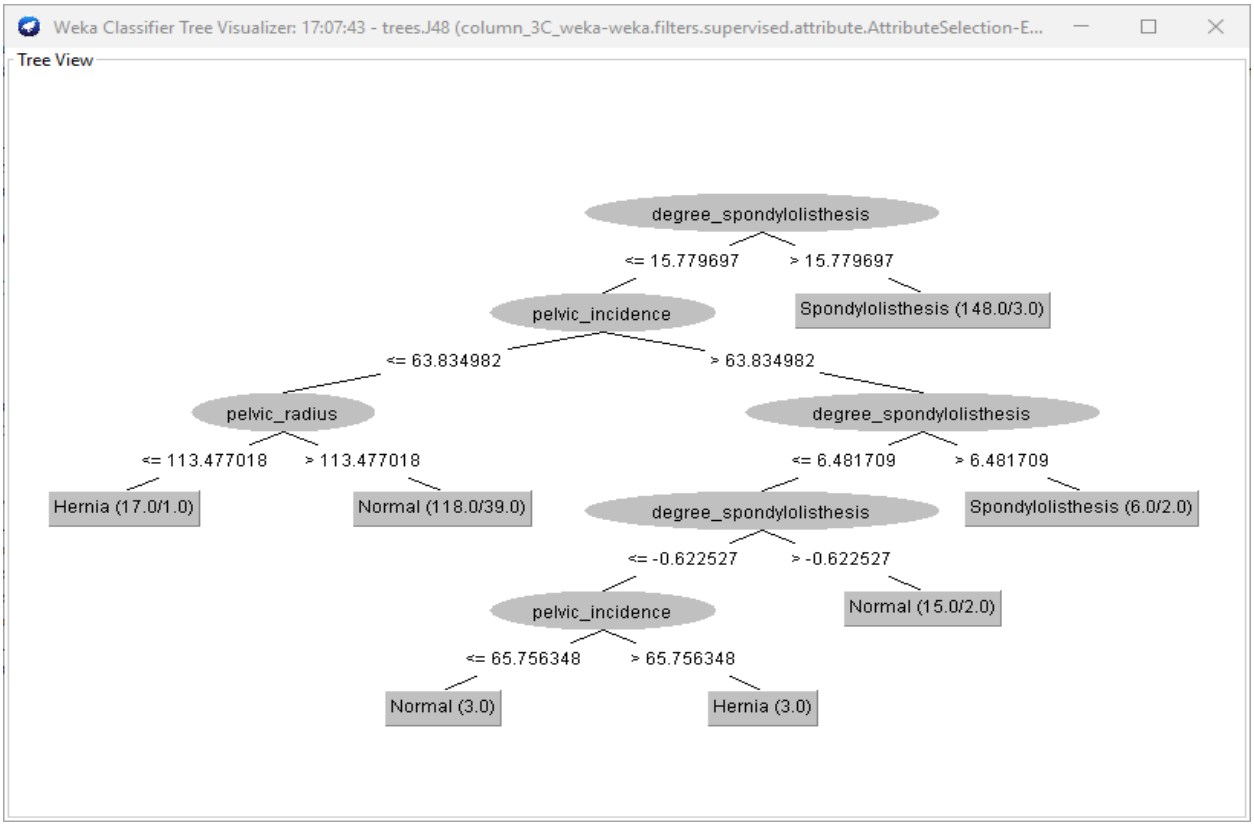
Confirmation of running decision tree split 70% with 4 attributes.

Structure and performance comparison

Tree structure comparison



With 7 attributes



Tree structure with 4 attributes.

Tree structure comparison:

Decision tree	Number of leaves	Size of tree	No of attributes	Number of instances
4 attributes	7	13	4	93
7 attributes	12	23	7	93

Performance compared.

Decision tree	Keppa statistics	MAE	RMSE	RAE	RRSE	Accuracy
4 attributes	0.6045	0.1764	0.3319	41.7969	71.3542	75.2688
7 attributes	0.6905	0.153	0.3218	36.2567	69.1727	80.6452

Confusion matrix compared:

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
  3  1 20 |  a = Hernia
  0 45  0 |  b = Spondylolisthesis
  0  2 22 |  c = Normal
```

Confusion matrix for decision tree with 4 attributes.

```
=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  1  8 |  a = Hernia
  0 45  0 |  b = Spondylolisthesis
  7  2 15 |  c = Normal
```

Confusion matrix for decision tree with 7 attributes.

Looking at the confusion matrix of both the parent (7 attributes) and subsets (4 attributes), the later i.e., the subset has eliminated the confusion that exist between Hernia and Normal when using the same split ratio but with 7 attributes.

PART TWO.

REGRESSION: USING WINE DATASET FROM THE SAME SOURCE WITH PHYTHON LIBRARY, SCIKIT-LEARN.

Introduction

This is a regression problem, for the Wine file contains numerical data as target. It has 178 instances or records and 14 attributes.

```
In [265]: print("Shape of the dataframe: ",df.shape)

          Shape of the dataframe:  (178, 14)
```

Data-preprocessing

```
In [276]: df.info()

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 178 entries, 0 to 177
          Data columns (total 14 columns):
           #   Column                        Non-Null Count  Dtype
          ---  ------                        --------------  -----
           0   alcohol                       178 non-null    float64
           1   malic_acid                    178 non-null    float64
           2   ash                           178 non-null    float64
           3   alcalinity_of_ash             178 non-null    float64
           4   magnesium                     178 non-null    float64
           5   total_phenols                 178 non-null    float64
           6   flavanoids                    178 non-null    float64
           7   nonflavanoid_phenols          178 non-null    float64
           8   proanthocyanins               178 non-null    float64
           9   color_intensity               178 non-null    float64
           10  hue                           178 non-null    float64
           11  od280/od315_of_diluted_wines  178 non-null    float64
           12  proline                       178 non-null    float64
           13  target                        178 non-null    int64
          dtypes: float64(13), int64(1)
          memory usage: 19.6 KB
```

The data also does not contain missing value:

```
In [271]: df.isna().sum() #no missing value

Out[271]: alcohol                         0
          malic_acid                      0
          ash                             0
          alcalinity_of_ash               0
          magnesium                       0
          total_phenols                   0
          flavanoids                      0
          nonflavanoid_phenols            0
          proanthocyanins                 0
          color_intensity                 0
          hue                             0
          od280/od315_of_diluted_wines    0
          proline                         0
          target                          0
          dtype: int64
```

Datatypes of the attributes:

```
In [276]: df.info()

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 178 entries, 0 to 177
          Data columns (total 14 columns):
           #   Column                        Non-Null Count  Dtype
          ---  ------                        --------------  -----
           0   alcohol                       178 non-null    float64
           1   malic_acid                    178 non-null    float64
           2   ash                           178 non-null    float64
           3   alcalinity_of_ash             178 non-null    float64
           4   magnesium                     178 non-null    float64
           5   total_phenols                 178 non-null    float64
           6   flavanoids                    178 non-null    float64
           7   nonflavanoid_phenols          178 non-null    float64
           8   proanthocyanins               178 non-null    float64
           9   color_intensity               178 non-null    float64
           10  hue                           178 non-null    float64
           11  od280/od315_of_diluted_wines  178 non-null    float64
           12  proline                       178 non-null    float64
           13  target                        178 non-null    int64
          dtypes: float64(13), int64(1)
          memory usage: 19.6 KB
```

Dataset also does not contain duplicate

```
In [215]: df.duplicated().any()
Out[215]: False
```

Feature selection:

Selecting which attributes should be the target and which should be the independent variable.

```
In [297]: X = df.drop(['target'], axis=1)
          y = df['target']
```

Normalization:

Then a preprocessing technique is applied to normalize the independent variables. This transforms the features of the dataset so that they have the same scale or range.

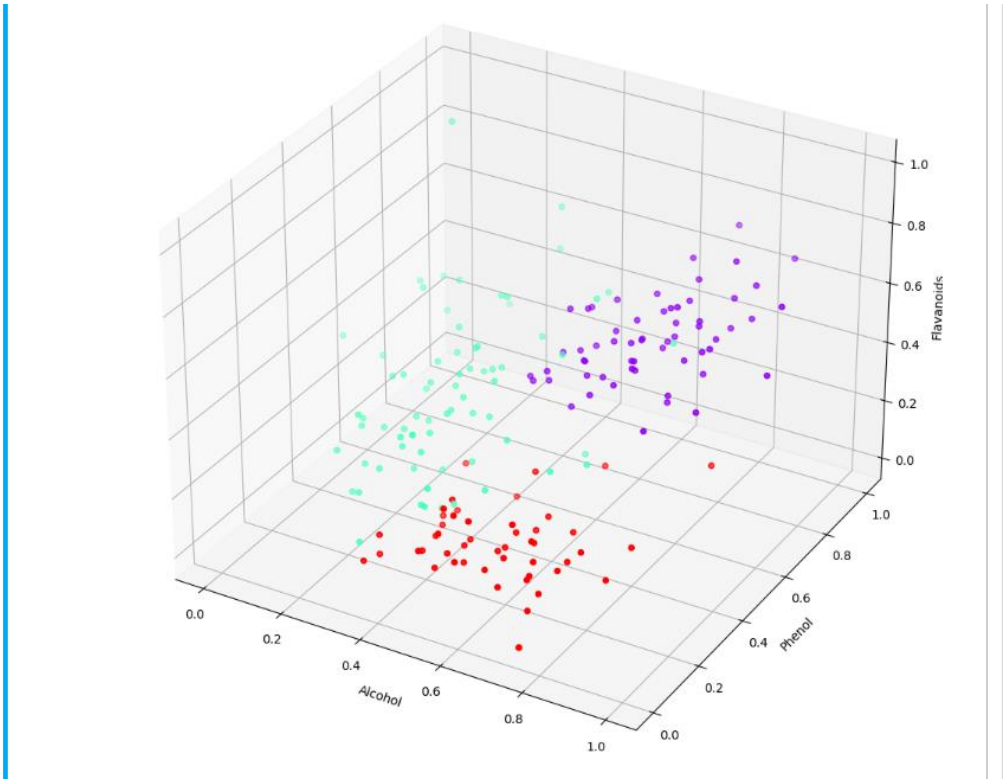
```
In [101]: X = minmax_scale(X)
```

Explorative analysis

Performing some explorative analysis to categories three of the attributes into the 3 targets:

```
In [102]: # Extract the desired attributes
alcohol = X[:, 0]
phenol = X[:, 5]
flavanoids = X[:, 6]

# Create a figure and subplot
fig = plt.figure(1, figsize=(12, 12))
ax = fig.add_subplot(111, projection='3d')

# Create a scatter plot
ax.scatter(alcohol, phenol, flavanoids, c=y, cmap='rainbow')

# Set labels and title
ax.set_xlabel('Alcohol')
ax.set_ylabel('Phenol')
ax.set_zlabel('Flavanoids')
plt.title('3D Scatter Plot of Wine Dataset')

# Display the plot
plt.show()
```

Result:



Constructing the decision trees:

GridSearchCv from scikit-learn was used in the construction of the decision tree:

The default parameters used in the construction of the different parameters are:

1. The criterion uses ‘gini’ index for splitting. We could have gone for ‘entropy’ but the GridSearchCV always had a better score for train model when using gini
2. The min_samples_split could be an array of values but to keep it uniform for the different split ratio of dataset, 2 was chosen.
3. The min_samples_leaf was kept as 1.
4. Random_state is 42.
5. Technique used to evaluate the performance of the model, cross_validation, is 10.

6. The model, estimator, used is DecisionTreeClassifier.

The split ratios used for testing dataset and training dataset are 20:80, 25:75 and 30:70.

In [110]:
```
# Define the test sizes
test_sizes = [0.2, 0.25, 0.3]

for test_size in test_sizes:

    # Split the data into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)

    # Define the parameter grid
    dtree_grid = {
        "criterion": ["gini"],
        "min_samples_split": [2],
        "min_samples_leaf": [1]
    }

    # Create the GridSearchCV object
    dtree = GridSearchCV(
        estimator=DecisionTreeClassifier(random_state=42),
        param_grid=dtree_grid,
        cv=10
    )

    # Fit the model to the training data
    dtree.fit(X_train, y_train)

    # Print the best hyperparameters and best score
    print("Decision Tuned Hyperparameters for test size {}: {}".format(test_size, dtree.best_params_))
    print("Decision Tuned Best Score for test size {}: {}".format(test_size, round(dtree.best_score_, 3)))

    # Use best classification model
    best_clf_dt = dtree.best_estimator_
    best_clf_dt.fit(X_train, y_train)
```

Output of 80% training set by 20% testing dataset

```
Decision Tuned Hyperparameters for test size 0.2: {'criterion': 'gini', 'm
in_samples_leaf': 1, 'min_samples_split': 2}
Decision Tuned Best Score for test size 0.2: 0.9
Decision Tree (DT) Classification Accuracy for test size 0.2: 94.44%
Kappa Statistics for test size 0.2: 0.914
Mean Absolute Error (MAE) for test size 0.2: 0.083
Root Mean Squared Error (RMSE) for test size 0.2: 0.373
Relative Absolute Error (RAE) for test size 0.2: 10.0%
Relative Root Squared Error (RRSE) for test size 0.2: 44.72%
```

Output of 75% training set by 25% testing dataset

```
Decision Tuned Hyperparameters for test size 0.25: {'criterion': 'gini',
'min_samples_leaf': 1, 'min_samples_split': 2}
Decision Tuned Best Score for test size 0.25: 0.924
Decision Tree (DT) Classification Accuracy for test size 0.25: 95.56%
Kappa Statistics for test size 0.25: 0.932
Mean Absolute Error (MAE) for test size 0.25: 0.067
Root Mean Squared Error (RMSE) for test size 0.25: 0.333
Relative Absolute Error (RAE) for test size 0.25: 7.14%
Relative Root Squared Error (RRSE) for test size 0.25: 35.71%
```

Output of 70% training set by 30% testing dataset

```
Decision Tuned Hyperparameters for test size 0.3: {'criterion': 'gini', 'm
in_samples_leaf': 1, 'min_samples_split': 2}
Decision Tuned Best Score for test size 0.3: 0.937
Decision Tree (DT) Classification Accuracy for test size 0.3: 96.3%
Kappa Statistics for test size 0.3: 0.943
Mean Absolute Error (MAE) for test size 0.3: 0.056
Root Mean Squared Error (RMSE) for test size 0.3: 0.304
Relative Absolute Error (RAE) for test size 0.3: 6.12%
Relative Root Squared Error (RRSE) for test size 0.3: 33.53%
```

Comparing the structures and classification performances of the different decision trees

Comparing structures:

Decision tree	Number of leaves	Size of tree	No of attributes	Number of instances
80:20	7	13	14	178
75: 25	7	13	14	178
70:30	7	13	14	178

Comparing performances:

Decision tree	Keppa statistics	MAE	RMSE	RAE	RRSE	Accuracy
80:20 split	0.914	0.083	0.373	10.0	44.72	94.44
75:25 split	0.932	0.067	0.333	7.14	35.71	95.56
70:30 split	0.943	0.056	0.304	6.12	33.53	96.3

Which of the classes is likely to be confused with each other.

```
                precision    recall  f1-score   support

           0       0.93      0.93      0.93        15
           1       0.95      1.00      0.97        18
           2       1.00      0.92      0.96        12

    accuracy                           0.96        45
   macro avg       0.96      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45

------------------------------------------------
```

Classification report of the 80:20 percent dataset training split

```
------------------------------------------------


                precision    recall  f1-score   support

           0       0.93      0.93      0.93        14
           1       0.93      1.00      0.97        14
           2       1.00      0.88      0.93         8

    accuracy                           0.94        36
   macro avg       0.95      0.93      0.94        36
weighted avg       0.95      0.94      0.94        36

------------------------------------------------
```

Classification report of the 75:25 percent dataset training split

	precision	recall	f1-score	support
0	0.95	0.95	0.95	19
1	0.95	1.00	0.98	21
2	1.00	0.93	0.96	14
accuracy			0.96	54
macro avg	0.97	0.96	0.96	54
weighted avg	0.96	0.96	0.96	54

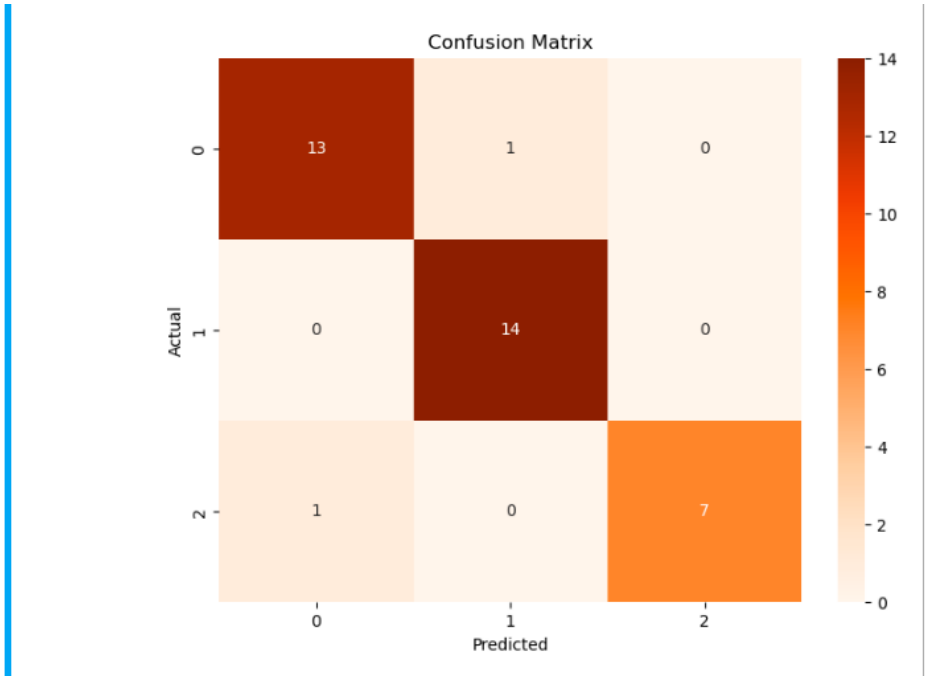
---

classification report of the 70:30 percent dataset training

Looking at the classification scores for the 80:20 percentage split, it can be seen that the target

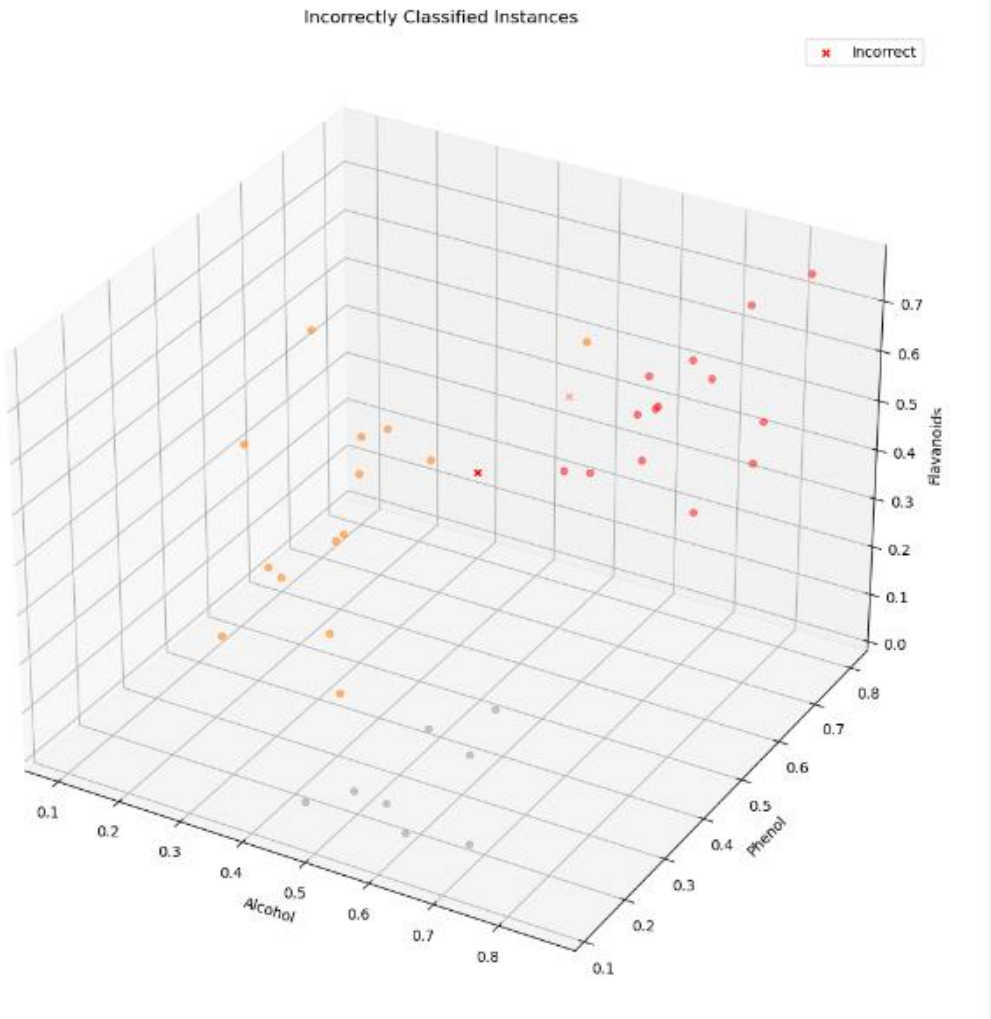
class 0 and 1 can be confused with each other. That can't be said of the other 2 decision trees.

This can also be confirmed from its confusion matrix below:

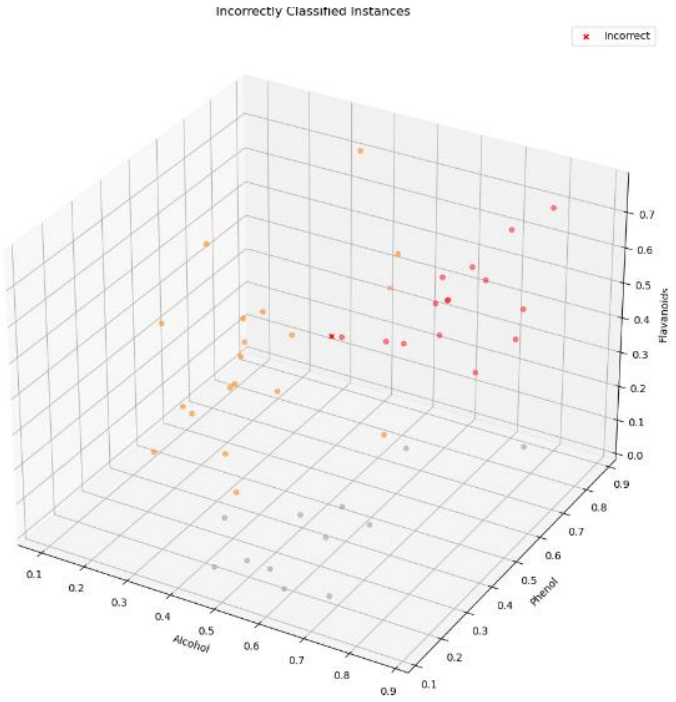


Misclassification compared found.

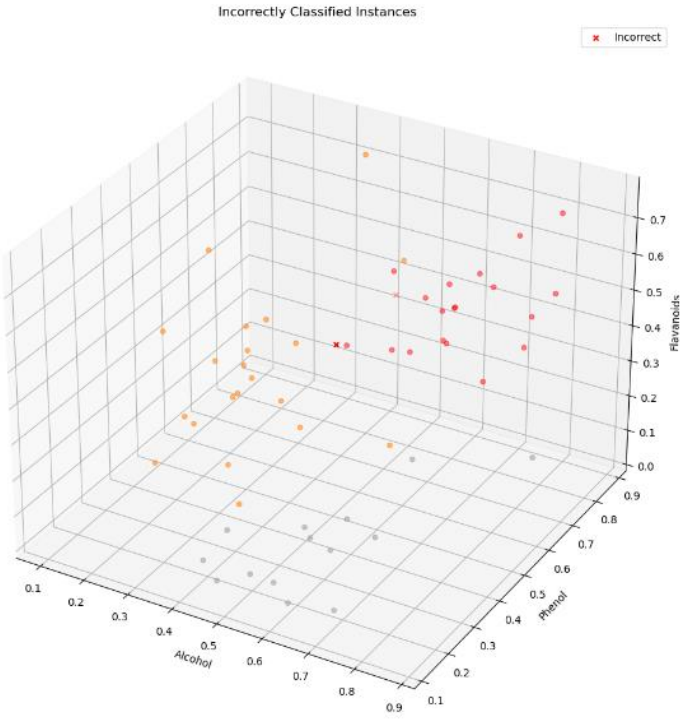
Using 3 attributes to compute the misclassification.



Misclassification for 80:20 dataset split.



Misclassification for 75:25 dataset split.



Misclassification for 70:30 dataset split.

Using sub attributes for the splitting 80:20

Feature selection

I will be using only 8 out of the 14 initial attributes.

```
In [133]: Xn=['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids']

In [134]: sub_features=df[Xn]

In [136]: sub_features.head()
```

Out[136]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69

Output:

```
Decision Tuned Hyperparameters for test size 0.2: {'criterion': 'gini', 'm
in_samples_leaf': 1, 'min_samples_split': 2}
Decision Tuned Best Score for test size 0.2: 0.909
Decision Tree (DT) Classification Accuracy for test size 0.2: 91.67%
Kappa Statistics for test size 0.2: 0.871
Mean Absolute Error (MAE) for test size 0.2: 0.083
Root Mean Squared Error (RMSE) for test size 0.2: 0.289
Relative Absolute Error (RAE) for test size 0.2: 10.0%
Relative Root Squared Error (RRSE) for test size 0.2: 34.64%
------------------------------------------------
```

Structures comparison between decision tree with same split 80:20 but different attributes.

Decision tree	Number of leaves	Size of tree	No of attributes	No of instances
14 attributes	7	13	14	178
7 attributes	10	19	7	178

Performance comparison between with the sub attributes

Decision tree	Keppa statistics	MAE	RMSE	RAE	RRSE	Accuracy
14 attributes	0.914	0.083	0.373	10.0	44.72	94.44
7 attributes	0.871	0.083	0.289	10.0	34.64	91.67

Comparison of the class reports

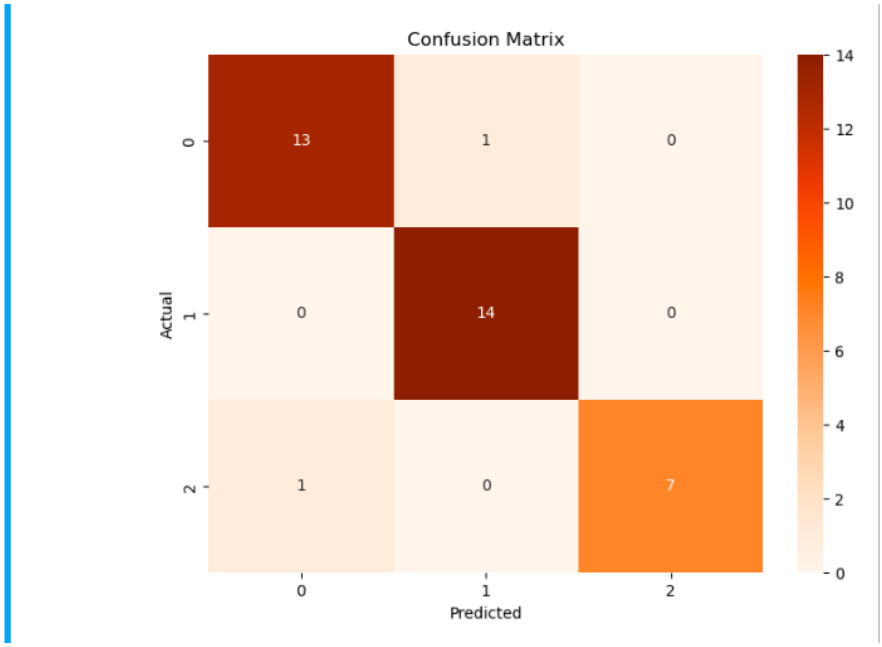
	precision	recall	f1-score	support
0	1.00	0.93	0.96	14
1	0.87	0.93	0.90	14
2	0.88	0.88	0.88	8
accuracy			0.92	36
macro avg	0.91	0.91	0.91	36
weighted avg	0.92	0.92	0.92	36

Classification report of the subset of attributes

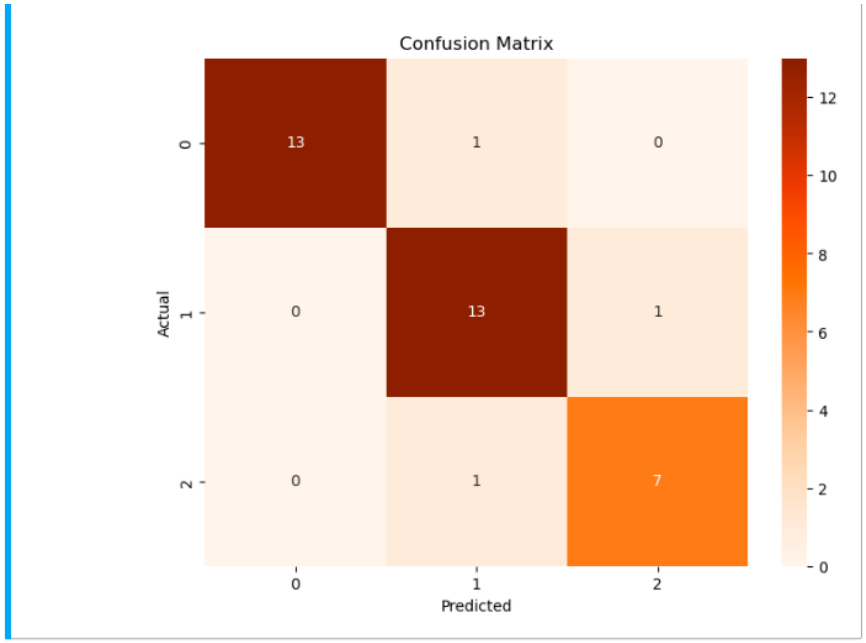
	precision	recall	f1-score	support
0	0.93	0.93	0.93	15
1	0.95	1.00	0.97	18
2	1.00	0.92	0.96	12
accuracy			0.96	45
macro avg	0.96	0.95	0.95	45
weighted avg	0.96	0.96	0.96	45

Classification report of the main set of attributes

Compare their confusion matrix.



confusion matrix of the subset of attributes



confusion matrix of the main set of attributes.

Looking at the classification report of the sub attributes and the confusion matrix, it can be seen that it targets 0 and 1 can be confused with each other, which was not the case in the case of the parent set of attributes.