



Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)



Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

Experiment 2 - 3

(Decision Tree)

Aim: Implement Decision Tree on the given Datasets to build a classifier and Regressor. Apply appropriate pruning method to overcome overfitting.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:

Dataset 1: PlayTennis.csv

Dataset 2: Iris.csv

Dataset 3: Breastcancer.csv

Dataset 4: car prediction.csv

1. Implement Decision tree classifier from scratch using Dataset 1 by defining Node class and Tree class.
2. Use python libraries to build a decision tree classifier on Dataset 2. Analyze the results using confusionmatrix and accuracy.
3. Write a code to show overfitting in the decision tree classifier built using Dataset 3. Use sklearn andmatplotlib.
4. Implement Decision tree regressor on Dataset 4.



Department of Computer Science and Engineering (Data Science)

Implement decision tree on IRIS dataset

CODE:

```
[15] #libraries
import random
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

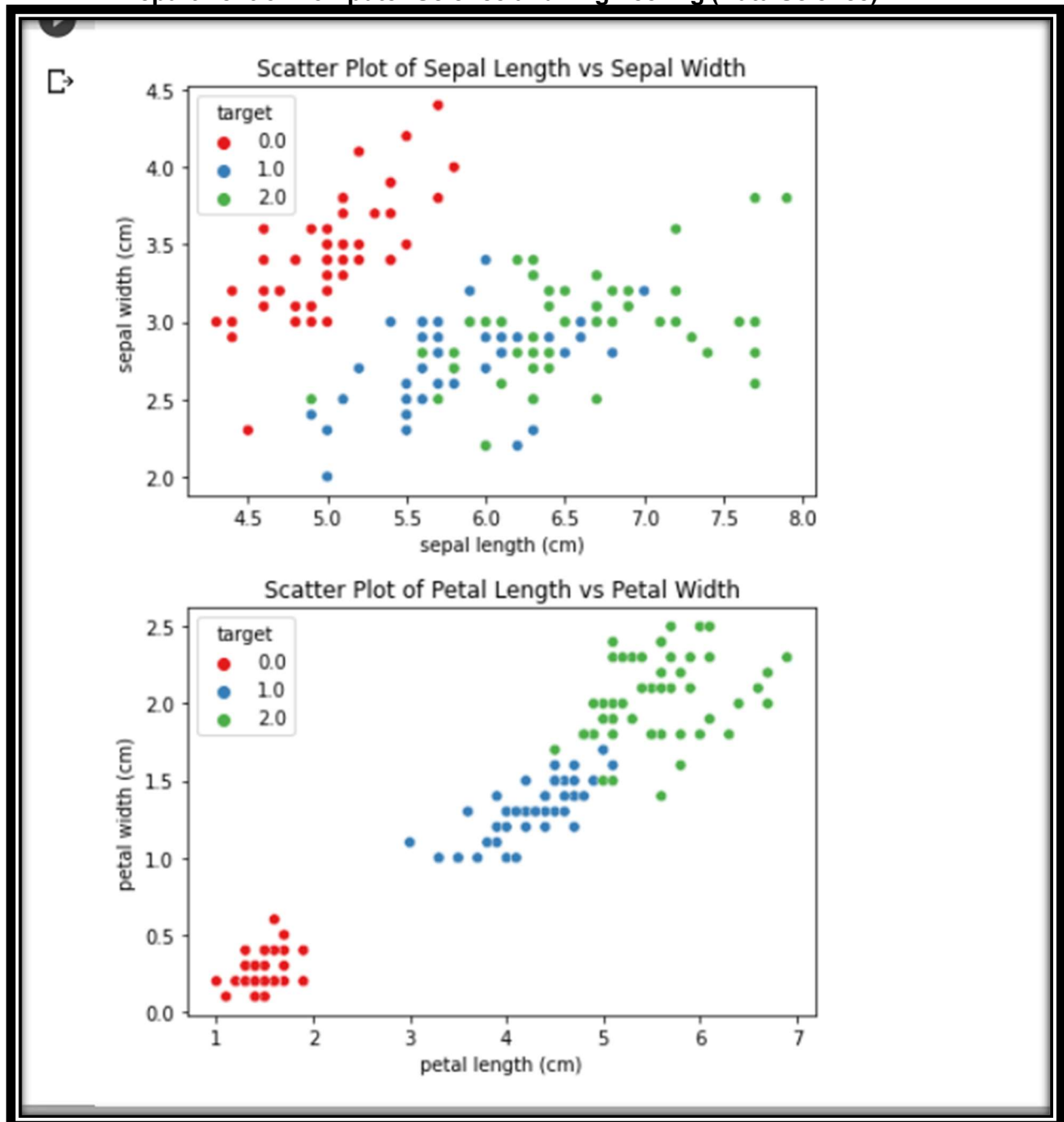
```
[45] #importing file
iris = pd.read_csv('/content/Iris.csv')
iris.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
data = pd.DataFrame(data= np.c_[iris['data'], iris['target']], columns= iris['feature_names'] + ['target'])

sns.scatterplot(data=data, x='sepal length (cm)', y='sepal width (cm)', hue='target', palette='Set1')
plt.title('Scatter Plot of Sepal Length vs Sepal Width')
plt.show()

sns.scatterplot(data=data, x='petal length (cm)', y='petal width (cm)', hue='target', palette='Set1')
plt.title('Scatter Plot of Petal Length vs Petal Width')
plt.show()
```

Department of Computer Science and Engineering (Data Science)

**Department of Computer Science and Engineering (Data Science)**

```
#splitting into train and test data
data = np.hstack([iris.data, iris.target.reshape(-1, 1)])
random.shuffle(data)
split_idx = int(0.8 * len(data))    #for 80:20 ratio
train_data = data[:split_idx]

X_train, y_train = train_data[:, :-1], train_data[:, -1]
X_test, y_test = test_data[:, :-1], test_data[:, -1]

print("Data size:", len(data))
print("Training set size:", len(train_data))
print("Testing set size:", len(test_data))
```

```
Data size: 150
Training set size: 120
Testing set size: 30
```

```
[34] def fit(X, y, max_depth=float('inf')):
      tree = _grow_tree(X, y, max_depth=max_depth)
      return tree
```

```
[35] def predict(tree, X):
      y_pred = np.zeros(len(X))
      for i, inputs in enumerate(X):
          y_pred[i] = _predict(tree, inputs)
      return y_pred
```

```
def _predict(node, inputs):
    if node['leaf']:
        return node['value']
    else:
        if inputs[node['feature']] < node['threshold']:
            return _predict(node['left'], inputs)
        else:
            return _predict(node['right'], inputs)
```

[+ Code](#)[+ Text](#)

**Department of Computer Science and Engineering (Data Science)**

```
#tree
def _grow_tree(X, y, depth=0, max_depth=float('inf')):
    num_samples, num_features = X.shape
    num_labels = len(set(y))

    if depth >= max_depth or num_labels == 1 or num_samples < 2:
        return {'leaf': True, 'value': _most_common_label(y)}

    best_feature, best_threshold = _best_split(X, y)

    left_indices = X[:, best_feature] < best_threshold
    right_indices = X[:, best_feature] >= best_threshold
    left = _grow_tree(X[left_indices], y[left_indices], depth + 1, max_depth=max_depth)
    right = _grow_tree(X[right_indices], y[right_indices], depth + 1, max_depth=max_depth)

    return {'leaf': False, 'feature': best_feature, 'threshold': best_threshold, 'left': left, 'right': right}
```

```
def _best_split(X, y):
    bestgini = float('inf')
    best_feature, best_threshold = None, None

    for feature in range(X.shape[1]):
        thresholds = sorted(set(X[:, feature]))

        for threshold in thresholds:
            left_indices = X[:, feature] < threshold
            left_labels = y[left_indices]
            right_labels = y[~left_indices]

            if len(left_labels) == 0 or len(right_labels) == 0:
                continue

            gini = (len(left_labels) / len(y)) * gini(left_labels) \
                + (len(right_labels) / len(y)) * gini(right_labels)

            if gini < bestgini:
                bestgini = gini
                best_feature = feature
                best_threshold = threshold

    return best_feature, best_threshold
```




Department of Computer Science and Engineering (Data Science)

```
[ ] def gini(y):
    counts = {label: 0 for label in set(y)}

    for label in y:
        counts[label] += 1

    impurity = 1
    for label in counts:
        prob = counts[label] / len(y)
        impurity -= prob ** 2

    return impurity

[ ] def _most_common_label(y):
    counts = {label: 0 for label in set(y)}

    for label in y:
        counts[label] += 1

    return max(counts, key=counts.get)
```

```
▶ #implementation
tree = fit(X_train, y_train, max_depth=3)
y_pred = predict(tree, X_test)
accuracy = sum(y_pred == y_test) / len(y_test)
print('The gini index is:', _gini(iris.target))
print(f"Accuracy: {accuracy:.3f}")
```

```
↳ The gini index is: 0.6666666666666665
   Accuracy: 0.700
```



Department of Computer Science and Engineering (Data Science)

Taken from net

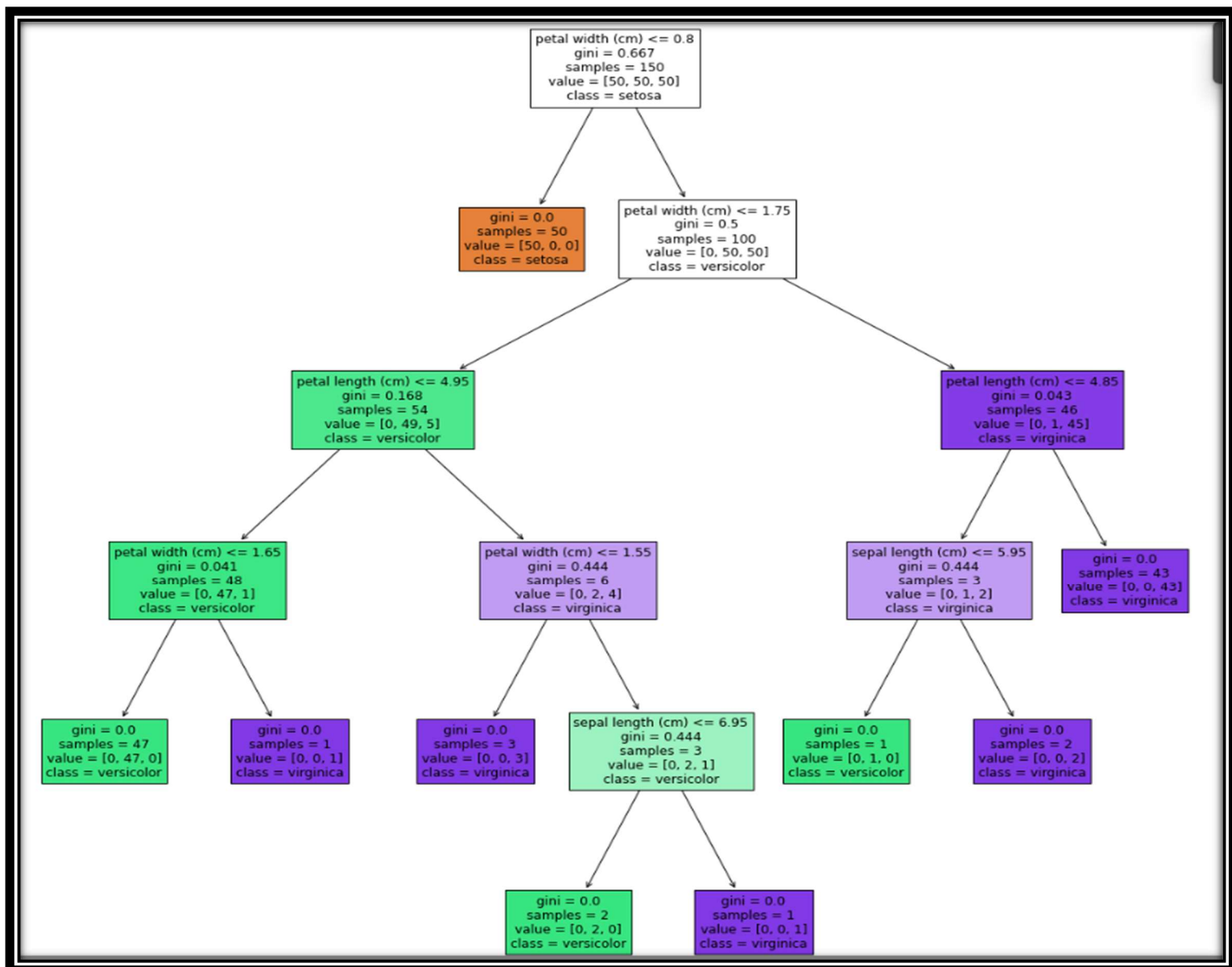
```
#printing decision tree
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

iris = load_iris()

clf = DecisionTreeClassifier()

clf.fit(iris.data, iris.target)

plt.figure(figsize=(20,20))
plot_tree(clf, filled=True, feature_names=iris.feature_names, class_names=iris.target_names)
plt.show()
```





#Q2

```
from sklearn.metrics import confusion_matrix

tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X_train, y_train)

y_pred = tree.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```



```
Accuracy: 0.77
Confusion Matrix:
[[ 9  0  0]
 [ 0 12  0]
 [ 0  7  2]]
```




BREAST CANCER DATASET

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
[140] breast=pd.read_csv("/content/Breast_cancer_data.csv")
breast.head()
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0

Splitting the dataset

```
X = breast.iloc[:, :-1]
y = breast.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```



Department of Computer Science and Engineering (Data Science)



```
tree = DecisionTreeClassifier(max_depth=3, random_state=42)
tree.fit(X_train, y_train)

# Predict on the testing set
y_pred = tree.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```



Accuracy: 0.77



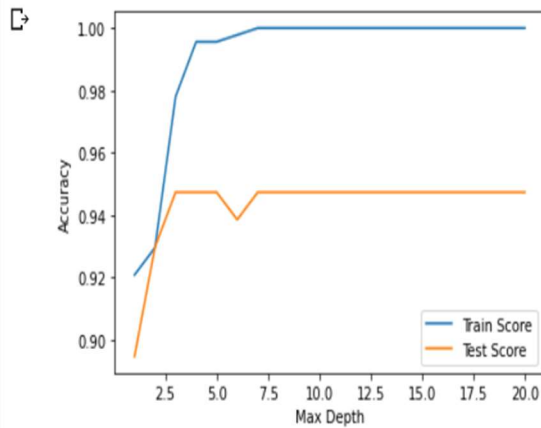
```
#Q3
max_depths = range(1, 21)
train_scores = []
test_scores = []

for max_depth in max_depths:
    tree = DecisionTreeClassifier(max_depth=max_depth, random_state=42)
    tree.fit(X_train, y_train)
    train_score = tree.score(X_train, y_train)
    test_score = tree.score(X_test, y_test)
    train_scores.append(train_score)
    test_scores.append(test_score)

# Plot the training and testing scores
plt.plot(max_depths, train_scores, label="Train Score")
plt.plot(max_depths, test_scores, label="Test Score")
plt.xlabel("Max Depth")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```



Department of Computer Science and Engineering (Data Science)



here the graph tells that If the training score keeps increasing while the testing score starts to plateau and then decrease, it indicates that the model is overfitting on the training data and performing poorly on the testing data

CAR PREDICTOR DATASET

car

```
car=pd.read_csv("/content/carprediction.csv")  
car.head()
```

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Price
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High- Performance	Compact	Coupe	26	19	19000
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	18000
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	19000
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	17000
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	17000



Department of Computer Science and Engineering (Data Science)

Applying datapreprocessing

✓ [276] car.info()
0s

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 11914 entries, 0 to 11913  
Columns: 1080 entries, Year to Vehicle Style_Wagon  
dtypes: float64(3), int64(5), uint8(1072)  
memory usage: 12.9 MB
```

✓ ▶ car.dtypes
0s

Make	object
Model	object
Year	int64
Engine Fuel Type	object
Engine HP	float64
Engine Cylinders	float64
Transmission Type	object
Driven_Wheels	object
Number of Doors	float64
Market Category	object
Vehicle Size	object
Vehicle Style	object
highway MPG	int64
city mpg	int64
Popularity	int64
MSRP	int64
dtype:	object



Department of Computer Science and Engineering (Data Science)

✓ [269] `car.isnull().sum()`

0s

```
Make                0
Model               0
Year               0
Engine Fuel Type    3
Engine HP           69
Engine Cylinders    30
Transmission Type   0
Driven_Wheels       0
Number of Doors     6
Market Category    3742
Vehicle Size        0
Vehicle Style       0
highway MPG         0
city mpg            0
Popularity          0
MSRP                0
dtype: int64
```



```
#replacing null values by mean wherever possible
mean_height = car['Engine HP'].mean()
car['Engine HP'].fillna(mean_height, inplace=True)
```

✓ [271] `car.isnull().sum()`

0s


```
Make                0
Model               0
Year               0
Engine Fuel Type    3
Engine HP           0
Engine Cylinders    30
Transmission Type   0
Driven_Wheels       0
Number of Doors     6
Market Category    3742
Vehicle Size        0
Vehicle Style       0
highway MPG         0
city mpg            0
Popularity          0
MSRP                0
dtype: int64
```




Department of Computer Science and Engineering (Data Science)

✓ [273] `car=car.dropna()`

0s

✓  `car.isnull().sum()`

0s

```
↳ Make      0
   Model     0
   Year      0
   Engine Fuel Type  0
   Engine HP  0
   Engine Cylinders  0
   Transmission Type  0
   Driven_Wheels  0
   Number of Doors  0
   Market Category  0
   Vehicle Size  0
   Vehicle Style  0
   highway MPG  0
   city mpg    0
   Popularity  0
   MSRP        0
   dtype: int64
```



Department of Computer Science and Engineering (Data Science)

```
from sklearn.metrics import mean_squared_error, r2_score

# Load the breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a decision tree regressor
regressor = DecisionTreeRegressor(random_state=42)

# Train the model on the training data
regressor.fit(X_train, y_train)

# Predict the target values on the test set
y_pred = regressor.predict(X_test)

# Evaluate the performance of the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean squared error: {:.2f}".format(mse))
print("R2 score: {:.2f}".format(r2))
```

```
Mean squared error: 0.05
R2 score: 0.78
```