Algorithm 2: Classification and Regression Tree (CART)

1.

The whole algorithms are divided into Decision Tree Regressor and Decision Tree Classifier part.

(1) Importing required libraries and loading the diabetes dataset from sklearn;

(2) Understanding the structure and content of the diabetes dataset;

(3) Using seaborn library to visualize the scatter charts and understanding the correlation between pair variables;

(4) Splitting the training and testing data and adapting decision tree regressor model;

2.

3.

4.

We built two different models by using two different dataset.
(1) Decision Tree Regressor
We only used mean_square_error to evaluate the result, and we did not apply other methods.

(2) Decision Tree Classifier
The code went smoothly because it was not extremely complicated.

(3) Graph of Decision Tree
We intended to use graphviz to draw the decision tree in Python, but did not succeed.

5.
To run the algorithm, we adapted the following Python library:
(1) sklearn:

- In python, sklearn is a machine learning package which include a lot of machine learning algorithms.

- We are using some of its modules like train_test_split, DecisionTreeClassifier and DecisionTreeRegressor.

(2) NumPy:
- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

(3) Pandas:
- Used to read and write different files.
- Data manipulation can be done easily with dataframes.

(4) Matplotlib:
- Used to draw the comparison lines between the prediction figures and actual figures.

(5) Seaborn:
- Used to draw scatter chart between the variables

We also adapted the following functions/methods
(1) Data Import:

- To import and manipulate the data we are using the pandas package provided in python

(2) Data Slicing:
- Before training the model we have to split the dataset into the training and testing dataset
- To split the dataset for training and testing we are using the sklearn module train_test_split
- Random-state variable is a pseudo-random number generator state used for random sampling

(3) Data Evaluating
- By exploring accuracy score, weighted average score of precision, weighted average score of recall, we evaluate the result.
- We used MSE to evaluate the result.

6.
a.
Library:
sklearn

Function:
```
print('Accuracy: ', metrics.accuracy_score(y_test, y_pred))
print(metrics.classification_report(y_test, y_pred))
```

Library:
Numpy

Function:
```
np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

y_test.std()

b.
In our model, we use the following indicators to evaluate the result.

(1) Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. For our model, we have got 0.97 which means our model is approximately 97% accurate.

Accuracy = TP+TN/TP+FP+FN+TN

(2) Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. We have got 0.97 precision which is pretty good.

Precision = TP/TP+FP

(3) Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class. We have got recall of 0.97 which is satisfying.

Recall = TP/TP+FN

(4) F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is 0.97.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

(5) The mean squared error (MSE) is largely used as a metric to determine the performance of an algorithm.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

c.
In the Decision Tree Regressor model, we use mean square error to find out the differences between actual dependent variable and predict dependent variable. The MSE is 70.62, which is not high based on the dataset. As a consequence, we consider the model to be successful.

In the Decision Tree Classifier model, the accuracy score is 0.97, which means the model is able to predict approximately 97% accurate results.

The weighted average score of precision and recall are both 0.97. They almost reached to a balance point. Based on the result, we consider that the model is efficient.

7.

Section	Links	Notes
1	https://scikit-learn.org/stable/modules/tree.html#tree-algorithms-id3-c4-5-c5-0-and-cart	

2	https://wiki.q-researchsoftware.com/wiki/Machine_Learning_-_Classification_And_Regression_Trees_(CART)#Example	
3	https://medium.com/@mathanrajsharma/fundamentals-of-classification-and-regression-trees-cart-e9af0b152503	
4	http://pages.stat.wisc.edu/~loh/treeprogs/guide/wires11.pdf	
5	https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart	
6	https://sefiks.com/2018/08/27/a-step-by-step-cart-decision-tree-example/	
7	https://www.analyticsvidhya.com/blog/2016/04/tree-based-algorithms-complete-tutorial-scratch-in-python/	
8	https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html#sphx-glr-auto-examples-tree-plot-iris-dtc-py	
9	https://www.youtube.com/watch?v=5eoFajw8TWk	
10	https://scikit-learn.org/stable/modules/tree.html	

8. Algorithm

In [44]:

```
# ADA_II
# HW3
# Team 2
# Huiwen Xu
# Qi Liu
# Jiahua Chen
```

Dataset links:

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html?highlight=iris#sklearn.datasets.load_iris

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html

We use iris dataset to display decision tree classification and use diabetes dataset to display decision tree regression.

Decision Tree Regressor¶

In [49]:

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [51]:

```
from sklearn import datasets, metrics
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
```

In [52]:

```
diabetes = datasets.load_diabetes()
diabetes.keys()
```

Out[52]:

```
dict_keys(['data', 'target', 'DESCR', 'feature_names',
'data_filename', 'target_filename'])
```

In [53]:

```
diabetes.DESCR
```

Out[53]:

```
'.. _diabetes_dataset:\n\nDiabetes dataset\n----------------\n\nTen
baseline variables, age, sex, body mass index, average
blood\npressure, and six blood serum measurements were obtained for
each of n =\n442 diabetes patients, as well as the response of
interest, a\nquantitative measure of disease progression one year
after baseline.\n\n**Data Set Characteristics:**\n\n :Number of
Instances: 442\n\n :Number of Attributes: First 10 columns are numeric
predictive values\n\n :Target: Column 11 is a quantitative measure of
disease progression one year after baseline\n\n :Attribute
Information:\n - Age\n - Sex\n - Body mass index\n - Average blood
pressure\n - S1\n - S2\n - S3\n - S4\n - S5\n - S6\n\nNote: Each of
these 10 feature variables have been mean centered and scaled by the
standard deviation times `n_samples` (i.e. the sum of squares of each
column totals 1).\n\nSource
URL:\nhttps://www4.stat.ncsu.edu/~boos/var.select/diabetes.html\n\nFor
more information see:\nBradley Efron, Trevor Hastie, Iain Johnstone
and Robert Tibshirani (2004) "Least Angle Regression," Annals of
Statistics (with discussion), 407-
499.\n(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pd
f)'
```

In [54]:

```
diabetes.feature_names
```

Out[54]:

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

In [19]:

```
diabetes.target
```

Out[19]:

```
array([151., 75., 141., 206., 135., 97., 138., 63., 110., 310., 101.,
69., 179., 185., 118., 171., 166., 144., 97., 168., 68., 49., 68.,
245., 184., 202., 137., 85., 131., 283., 129., 59., 341., 87., 65.,
102., 265., 276., 252., 90., 100., 55., 61., 92., 259., 53., 190.,
142., 75., 142., 155., 225., 59., 104., 182., 128., 52., 37., 170.,
170., 61., 144., 52., 128., 71., 163., 150., 97., 160., 178., 48.,
270., 202., 111., 85., 42., 170., 200., 252., 113., 143., 51., 52.,
210., 65., 141., 55., 134., 42., 111., 98., 164., 48., 96., 90., 162.,
150., 279., 92., 83., 128., 102., 302., 198., 95., 53., 134., 144.,
232., 81., 104., 59., 246., 297., 258., 229., 275., 281., 179., 200.,
200., 173., 180., 84., 121., 161., 99., 109., 115., 268., 274., 158.,
107., 83., 103., 272., 85., 280., 336., 281., 118., 317., 235., 60.,
174., 259., 178., 128., 96., 126., 288., 88., 292., 71., 197., 186.,
25., 84., 96., 195., 53., 217., 172., 131., 214., 59., 70., 220.,
268., 152., 47., 74., 295., 101., 151., 127., 237., 225., 81., 151.,
107., 64., 138., 185., 265., 101., 137., 143., 141., 79., 292., 178.,
91., 116., 86., 122., 72., 129., 142., 90., 158., 39., 196., 222.,
277., 99., 196., 202., 155., 77., 191., 70., 73., 49., 65., 263.,
248., 296., 214., 185., 78., 93., 252., 150., 77., 208., 77., 108.,
160., 53., 220., 154., 259., 90., 246., 124., 67., 72., 257., 262.,
275., 177., 71., 47., 187., 125., 78., 51., 258., 215., 303., 243.,
91., 150., 310., 153., 346., 63., 89., 50., 39., 103., 308., 116.,
145., 74., 45., 115., 264., 87., 202., 127., 182., 241., 66., 94.,
283., 64., 102., 200., 265., 94., 230., 181., 156., 233., 60., 219.,
80., 68., 332., 248., 84., 200., 55., 85., 89., 31., 129., 83., 275.,
65., 198., 236., 253., 124., 44., 172., 114., 142., 109., 180., 144.,
163., 147., 97., 220., 190., 109., 191., 122., 230., 242., 248., 249.,
192., 131., 237., 78., 135., 244., 199., 270., 164., 72., 96., 306.,
91., 214., 95., 216., 263., 178., 113., 200., 139., 139., 88., 148.,
88., 243., 71., 77., 109., 272., 60., 54., 221., 90., 311., 281.,
182., 321., 58., 262., 206., 233., 242., 123., 167., 63., 197., 71.,
168., 140., 217., 121., 235., 245., 40., 52., 104., 132., 88., 69.,
219., 72., 201., 110., 51., 277., 63., 118., 69., 273., 258., 43.,
198., 242., 232., 175., 93., 168., 275., 293., 281., 72., 140., 189.,
181., 209., 136., 261., 113., 131., 174., 257., 55., 84., 42., 146.,
212., 233., 91., 111., 152., 120., 67., 310., 94., 183., 66., 173.,
72., 49., 64., 48., 178., 104., 132., 220., 57.])
```

In [55]:

```
x = diabetes.data
y = diabetes.target
x.shape, y.shape
```

Out[55]:

```
((442, 10), (442,))
```

In [56]:

```
df = pd.DataFrame(x, columns=diabetes.feature_names)
df['target'] = y
df.head()
```

Out[56]:

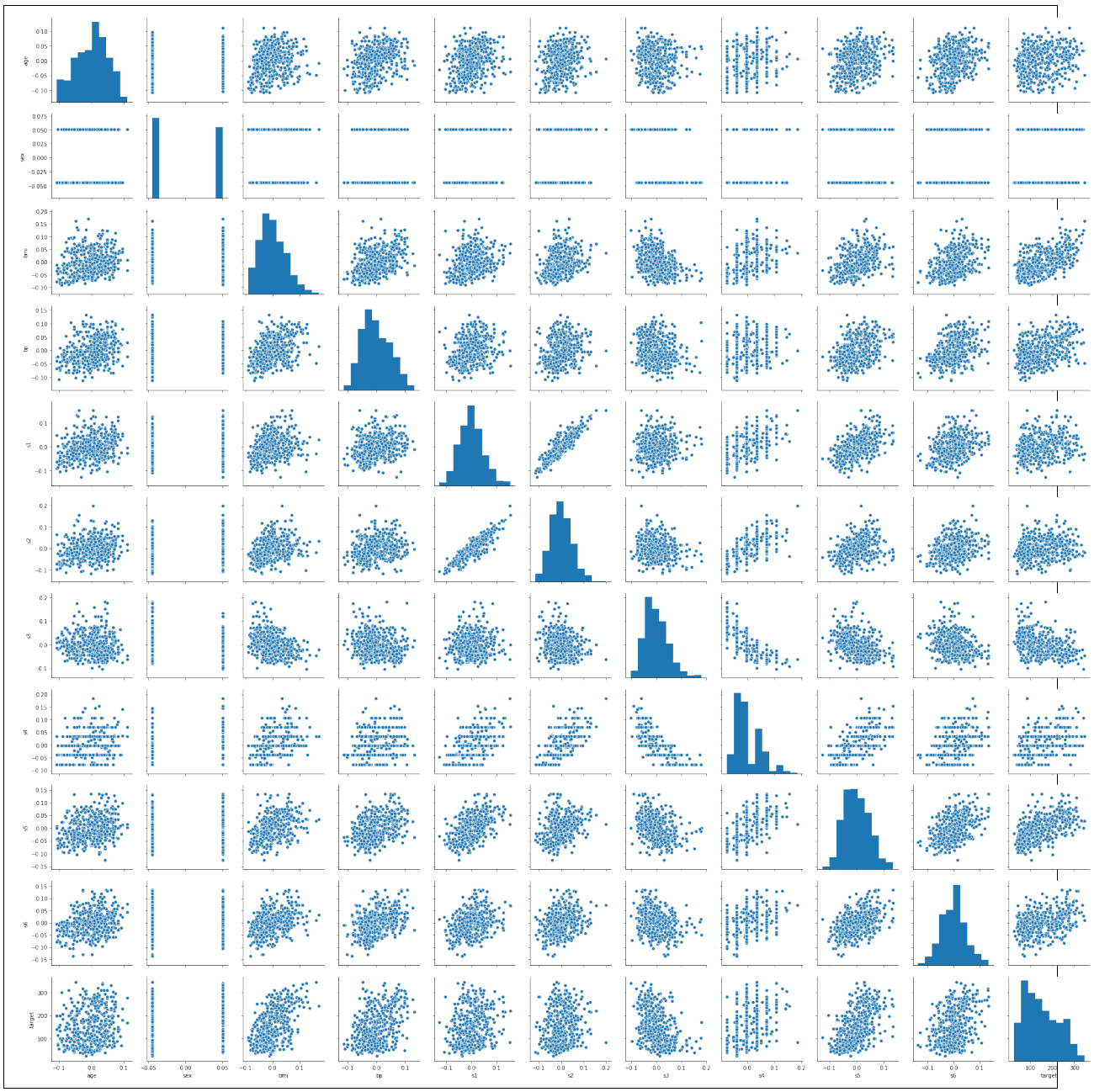
	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641	135.0

In [58]:

```
sns.pairplot(df)
```

Out[58]:

```
<seaborn.axisgrid.PairGrid at 0x26ae1e90>
```



In [59]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size =
0.2, random_state = 42)
```

In [61]:

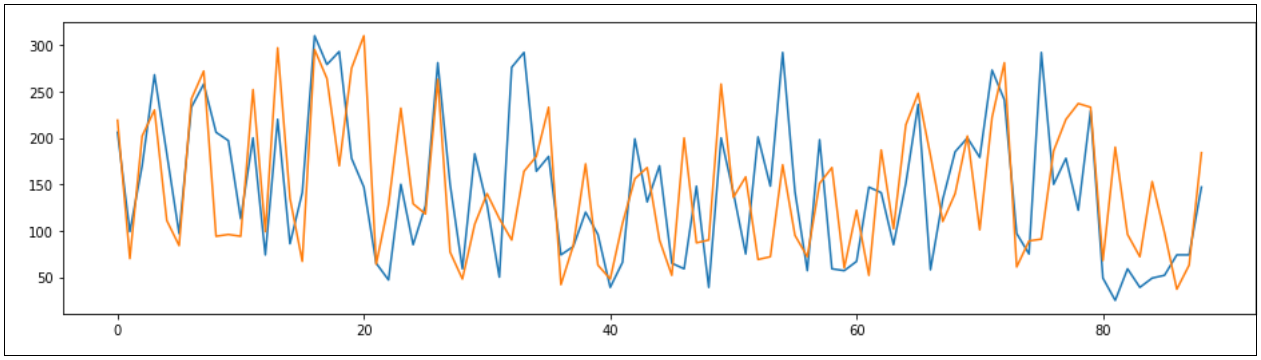
```
regressor = DecisionTreeRegressor(random_state=42)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

In [62]:

```
plt.figure(figsize=(16, 4))
plt.plot(y_pred)
plt.plot(y_test)
```

Out[62]:

```
[<matplotlib.lines.Line2D at 0x2a469e10>]
```



In [63]:

```
np.sqrt(metrics.mean_squared_error(y_test, y_pred))
```

Out[63]:

```
70.61829663921893
```

In [64]:

```
y_test.std()
```

Out[64]:

```
72.78840394263774
```

In [ ]:

Decision Tree Classifier¶

In [68]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [69]:

```
iris = datasets.load_iris()
```

In [70]:

```
iris.target_names
```

Out[70]:

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [71]:

```
iris.feature_names
```

Out[71]:

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal
width (cm)']
```

In [72]:

```
x = iris.data
y = iris.target
```

In [73]:

```
df = pd.DataFrame(x, columns=iris.feature_names)
df['target'] = y
df.head()
```

Out[73]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2

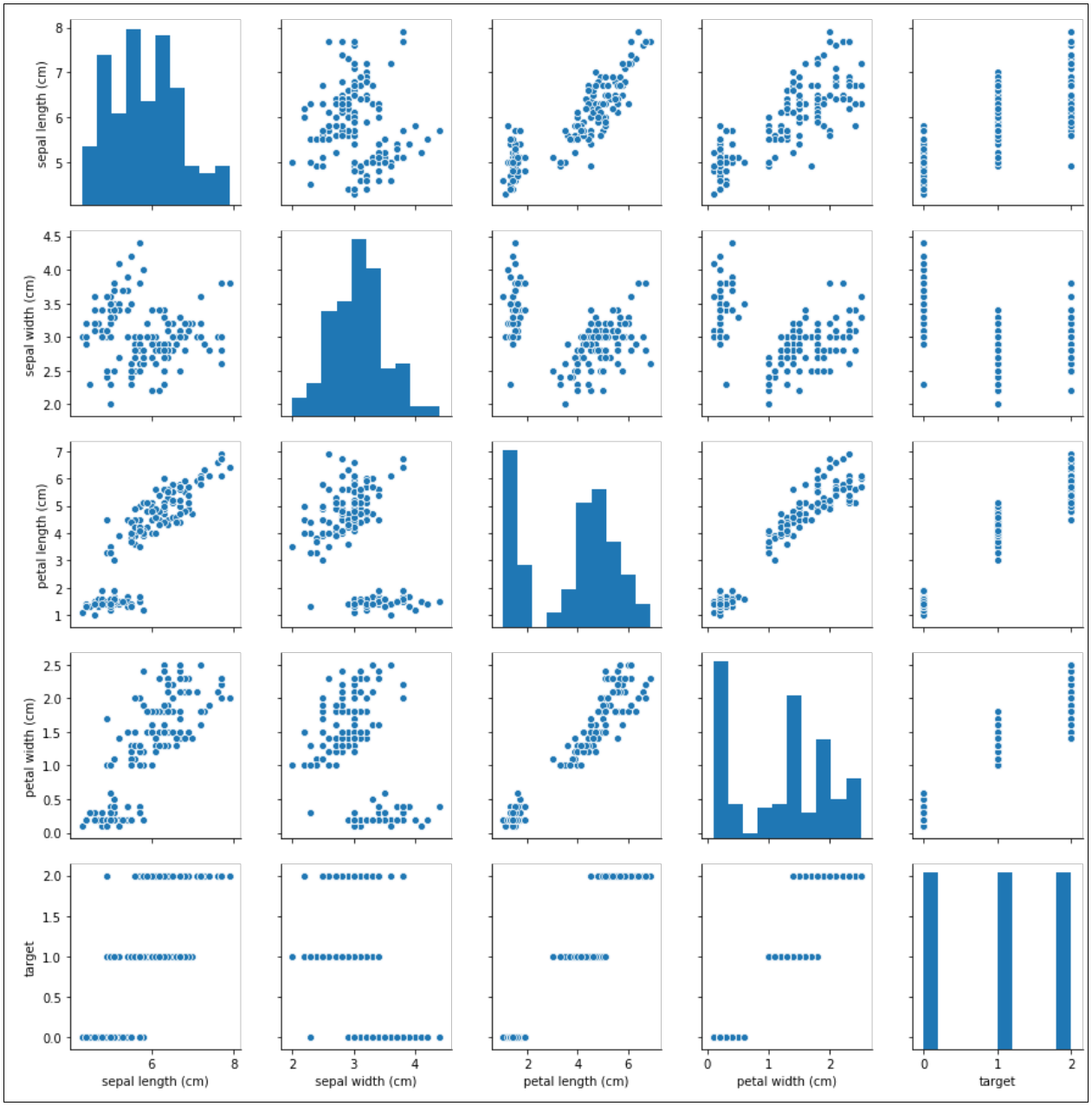
4	5.0	3.6	1.4	0.2

In [36]:

```
sns.pairplot(df)
```

Out[36]:

```
<seaborn.axisgrid.PairGrid at 0x1ef83430>
```



In [74]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y,
random_state = 1, test_size = 0.2, stratify = y)
```

In [75]:

```
clf = DecisionTreeClassifier(criterion='gini', random_state=1)
```

In [76]:

```
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
```

In [77]:

```
print('Accuracy: ', metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9666666666666667
```

In [78]:

```
print(metrics.classification_report(y_test, y_pred))
```

```
precision recall f1-score support 0 1.00 1.00 1.00 10 1 0.91 1.00 0.95
10 2 1.00 0.90 0.95 10 accuracy 0.97 30 macro avg 0.97 0.97 0.97 30
weighted avg 0.97 0.97 0.97 30
```

Conclusion¶

We use classical dataset (diabetes dataset) to perform decision tree regressor, and use train_test_split (0.8/0.2) to evaluate the model. We use mean_square_error to evaluate the result, with the score being 70.62, which is acceptable.

We use classical dataset (iris dataset) to perform decision tree classifier, and use train_test_split (0.8/0.2) to evaluate the decisiontreeclassifier model. The accuracy, weighted average of precision and weighted average of recall are all 0.97, which are satisfying.

Additional Notes¶

Because the dataset does not contain missing data, we need not to perform data cleaning.

We use 80% of the dataset as training dataset, and 20% of the dataset as testing dataset because it is widely used. If we change the figure, we may get better mean_square_error and better metrics result.

In [ ]: