Classification using a Decision Tree

This notebook demonstrates generating a practice dataset, visualising it, and using a decision tree to classify it. It is important that you generate a unique dataset so that you have different results to your peers so that you can undertake a unique analysis. To achieve this, where a random number seed is specified you need to replace the *0* with the *last 3 digits of your student number*.

You are required to document your work in *markdown cells*. Empty cells have been included, but you can add more if you want for either code experimentation or futher explanation. Concise documentation for *markdown* can be found at https://sourceforge.net/p/jupiter/wiki/markdown_syntax/#md_ex_pre and https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet

Original notebook by Dr Kevan Buckley, University of Wolverhampton, 2019. This submission by ***your name and student number***

In markdown cells like this one explain the code or results below

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pydotplus
from sklearn import tree
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In the above code block different libraries are imported.
Numpy- Numpy stands for Numerical Python and is a python library which is used for working with arrays. It aims to provide 50x faster array object in comparision to the traditional lists in python.Functions for working in linear algebra, fourier transform and matrices are also there in numpy. It supports big multidimensional array and matrix.

Pandas- Pandas is a software library for Python which is used for data manipulation and data analysis. It provides features like reshaping datasets,data alignment, dataset merging/joining,data filetration and many more.

matplotlib.pyplot- Matplotlin is a plotting library for Python. Object oriented API for embedding plots

is provided by it. Pyplot function is used to make changes to the figure like creating, plotting line or creating plotting areas. seaborn- It is data visualization library for python which is based upon matplotlib. Drawing attractive and infromative graphics is possible using it.
Pydotplus- It provides a python interface to Graphviz's Dot language. It is improved version of the old pydot project.

Scikit-learn - This is pyhton library used for machine learning. it contains efficient tools for machine learning and also for classification, regression, clustering and dimensionality reduction.

Make sure that you replace the zero with the last 3 digits of your student number. If this includes a leading zero use the last 4 digits.

```
features, target = make_classification(
    n_samples=294, n_features=4, n_classes=3, n_clusters_per_class=1, random_state=0)
```

The make_classification() function is used to generate a random classification dataset. It is an inbuilt function.Having multiple features and addition of noise to the data is allowed for the users

Double-click (or enter) to edit

```
features.shape
```

```
(294, 4)
```

Ndarray.shape is used in order to get array's dimension. The value is obtained as tuple. The output generated is (200,4) which indicates that for 200 samples there are 4 features or we can say that there are 200 rows and 4 columns.

Double-click (or enter) to edit

```
target.shape
```

```
(294,)
```

In the above code, target.shape is used to get the dimension of target value. The tuple obtained here is (200,0) which indicates that there are 200 rows and 0 columns.

Double-click (or enter) to edit

```
features[0]
```

```
array([ 1.51386415,  0.50563903, -1.95184332,  1.5660616 ])
```

The output here indicates the array which contains features of the first data sample of training dataset obtained after entering the above code features[0].

Double-click (or enter) to edit

```
target[0]
```

```
1
```

The output of the target value at 0 index which is the first training dataset is 2. This indicates that the first training dataset belongs to the third class.

Double-click (or enter) to edit

```
feature_names = ['feature_0', 'feature_1', 'feature_2', 'feature_3']
feature_names
```

```
['feature_0', 'feature_1', 'feature_2', 'feature_3']
```

In the above code, feature names are added to feature_names variable as a list. The name of the features are 'feature_0', 'feature_1', 'feature_2' and 'feature_3'.

Double-click (or enter) to edit

```
features_df = pd.DataFrame(features, columns=feature_names)
```

pd.DataFrame is used to create 2D data structure which helps to align data in tabular form as row and column. The parameters contained in it are data,indexn columns, dtype and copy. In the above code it is used to align the data in tabular form. The rows are features and the columns are the 4 feature_names.

Double-click (or enter) to edit

```
features_df.head()
```

	feature_0	feature_1	feature_2	feature_3
0	1.513864	0.505639	-1.951843	1.566062
1	0.107449	0.157309	-0.589637	1.001099
2	-0.325184	0.160885	-0.581982	1.638892
3	2.351449	0.325585	-1.323450	-0.937662
4	0.865769	0.028451	-0.147614	-1.015327

The above code returned the first five rows of the data as the default value of n rows of the data to be displayed is 5 in the features_df.head() function.

Double-click (or enter) to edit

```
target_df = pd.DataFrame(target, columns=['target'])
```

This is done to align target values in tabular form. One column name is given i.e. target as there is only one column of data.

Double-click (or enter) to edit

```
target_df.head()
```

	target
0	1
1	0
2	0
3	2
4	2

As before, it gives the first five rows of the data as the default value of n rows of the data to be displayed is 5

Double-click (or enter) to edit

```
dataset = pd.concat([features_df, target_df], axis=1)
```

pd.concat is used in the above code to combine features_df and targer_df dataframes to dataset variable.

Double-click (or enter) to edit

```
dataset.head()
```

	feature_0	feature_1	feature_2	feature_3	target
0	1.513864	0.505639	-1.951843	1.566062	1
1	0.107449	0.157309	-0.589637	1.001099	0
2	-0.325184	0.160885	-0.581982	1.638892	0
3	2.351449	0.325585	-1.323450	-0.937662	2
4	0.865769	0.028451	-0.147614	-1.015327	2

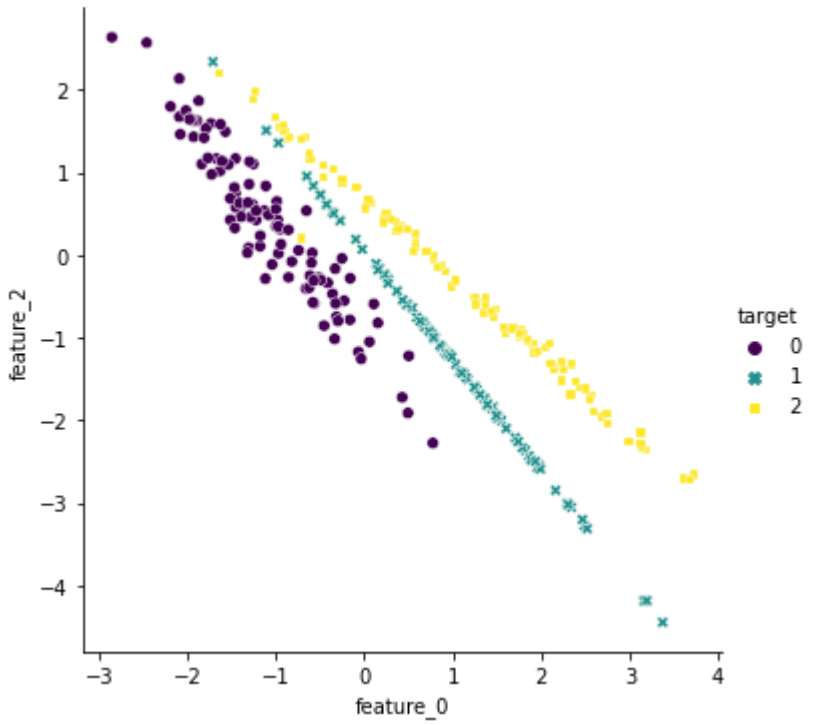
As before, this shows first five rows of the data as the default value of n rows of the data to be displayed is 5. In this table, features and target are shown together in combined form.

Double-click (or enter) to edit

```
sns.relplot(
    x='feature_0', y='feature_1', hue='target', style='target', data=dataset)
plt.show()
```

Try different palettes until you find find a suitable one then remove this comment. Available palettes on the lecture development machine were: Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r, CMRmap, CMRmap_r, Dark2, Dark2_r, GnBu, GnBu_r, Greens, Greens_r, Greys, Greys_r, OrRd, OrRd_r, Oranges, Oranges_r, PRGn, PRGn_r, Paired, Paired_r, Pastel1, Pastel1_r, Pastel2, Pastel2_r, PiYG, PiYG_r, PuBu, PuBuGn, PuBuGn_r, PuBu_r, PuOr, PuOr_r, PuRd, PuRd_r, Purples, Purples_r, RdBu, RdBu_r, RdGy, RdGy_r, RdPu, RdPu_r, RdYlBu, RdYlBu_r, RdYlGn, RdYlGn_r, Reds, Reds_r, Set1, Set1_r, Set2, Set2_r, Set3, Set3_r, Spectral, Spectral_r, Wistia, Wistia_r, YlGn, YlGnBu, YlGnBu_r, YlGn_r, YlOrBr, YlOrBr_r, YlOrRd, YlOrRd_r, afmhot, afmhot_r, autumn, autumn_r, binary, binary_r, bone, bone_r, brg, brg_r, bwr, bwr_r, cividis, cividis_r, cool, cool_r, coolwarm, coolwarm_r, copper, copper_r, cubehelix, cubehelix_r, flag, flag_r, gist_earth, gist_earth_r, gist_gray, gist_gray_r, gist_heat, gist_heat_r, gist_ncar, gist_ncar_r, gist_rainbow, gist_rainbow_r, gist_stern, gist_stern_r, gist_yarg, gist_yarg_r, gnuplot, gnuplot2, gnuplot2_r, gnuplot_r, gray, gray_r, hot, hot_r, hsv, hsv_r, icefire, icefire_r, inferno, inferno_r, jet, jet_r, magma, magma_r, mako, mako_r, nipy_spectral, nipy_spectral_r, ocean, ocean_r, pink, pink_r, plasma, plasma_r, prism, prism_r, rainbow, rainbow_r, rocket, rocket_r, seismic, seismic_r, spring, spring_r, summer, summer_r, tab10, tab10_r, tab20, tab20_r, tab20b, tab20b_r, tab20c, tab20c_r, terrain, terrain_r, twilight, twilight_r, twilight_shifted, twilight_shifted_r, viridis, viridis_r, vlag, vlag_r, winter, winter_r.

```
sns.relplot(
    x='feature_0', y='feature_2', hue='target', style='target', palette='viridis', data=datas
plt.show()
```



In the above plot, x axis represents feature_0 and y axis represents feature_2.

Hue parameter is color appearance parameters which produces elements with different colours.

Style parameter groups the target variable according to the value. For example: values 0 have the same styles of plotting whereas value 1 has different style and so on.

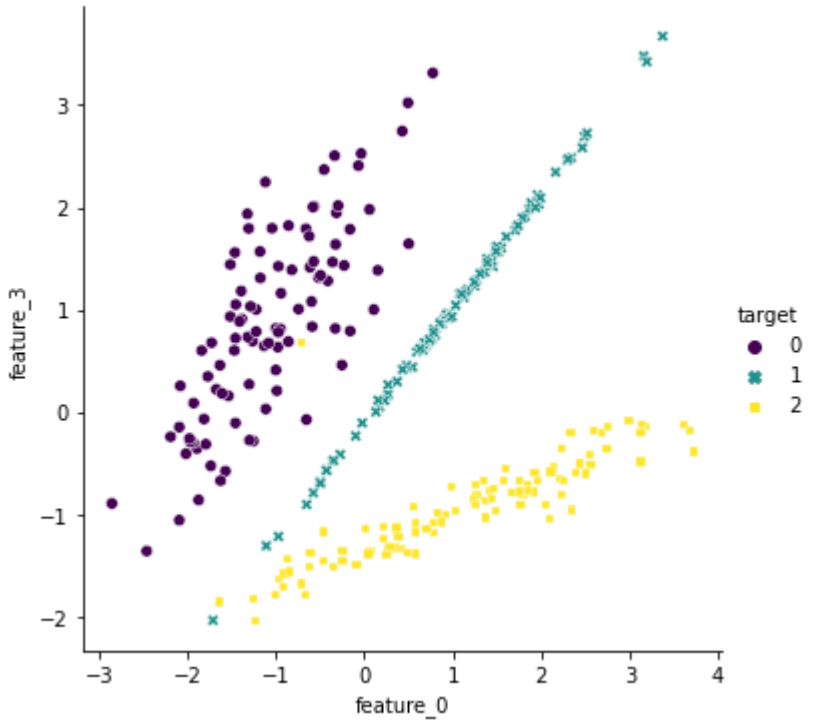
Data parameter can be assigned with named variables or dataset that will be reshaped. Dataset is used hera because it contains all the vvalues of x,y and target which was created before.

Palette parameter is used to choose various colour options when plotting the hue.

plt.show() is used for displaying a figure.

Double-click (or enter) to edit

```
sns.relplot(
    x='feature_0', y='feature_3', hue='target', style='target', palette='viridis', data=datas
plt.show()
```



In the above plot, x axis represents feature_0 and y axis represents feature_3. Hue parameter is color appearance parameters which produces elements with different colours.

Style parameter groups the target variable according to the value. For example: values 0 have the same styles of plotting whereas value 1 has different style and so on.

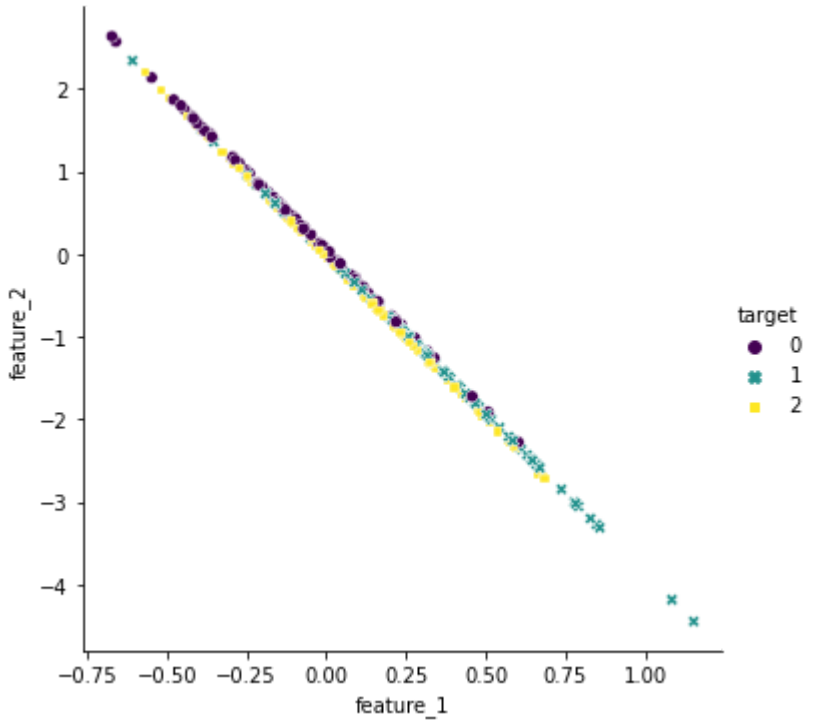
Data parameter can be assigned with named variables or dataset that will be reshaped. Dataset is used hera because it contains all the vvalues of x,y and target which was created before.

Palette parameter is used to choose various colour options when plotting the hue.

plt.show() is used for displaying a figure.

Double-click (or enter) to edit

```
sns.relplot(
    x='feature_1', y='feature_2', hue='target', style='target', palette='viridis', data=datas
plt.show()
```



In the above plot, x axis represents feature_1 and y axis represents feature_2.

Hue parameter is color appearance parameters which produces elements with different colours.

Style parameter groups the target variable according to the value. For example: values 0 have the same styles of plotting whereas value 1 has different style and so on.

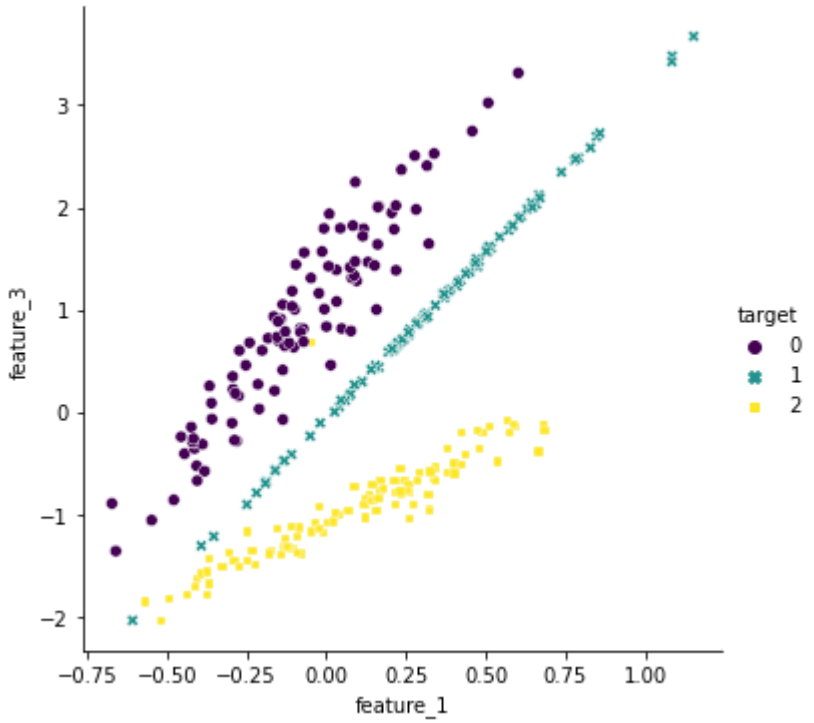
Data parameter can be assigned with named variables or dataset that will be reshaped. Dataset is used hera because it contains all the vvalues of x,y and target which was created before.

Palette parameter is used to choose various colour options when plotting the hue.

plt.show() is used for displaying a figure.

Double-click (or enter) to edit

```
sns.relplot(
    x='feature_1', y='feature_3', hue='target', style='target', palette='viridis', data=datas
plt.show()
```



In the above plot, x axis represents feature_1 and y axis represents feature_3. Hue parameter is color appearance parameters which produces elements with different colours.

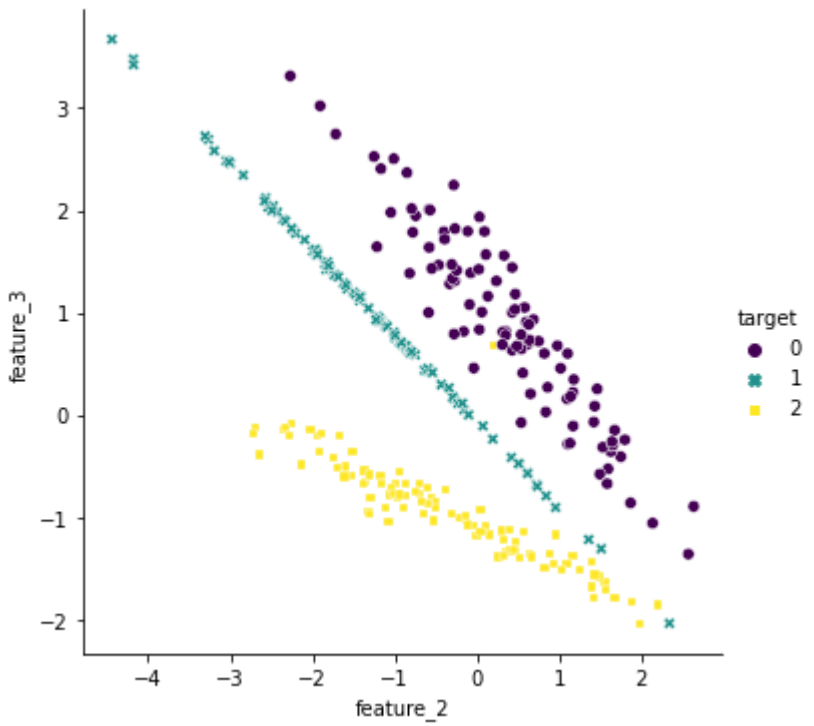
Style parameter groups the target variable according to the value. For example: values 0 have the same styles of plotting whereas value 1 has different style and so on.

Data parameter can be assigned with named variables or dataset that will be reshaped. Dataset is used hera because it contains all the vvalues of x,y and target which was created before.

Palette parameter is used to choose various colour options when plotting the hue.

plt.show() is used for displaying a figure

```
sns.relplot(
    x='feature_2', y='feature_3', hue='target', style='target', palette='viridis', data=datas
plt.show()
```



In the above plot, x axis represents feature_2 and y axis represents feature_3. Hue parameter is color appearance parameters which produces elements with different colours.

Style parameter groups the target variable according to the value. For example: values 0 have the same styles of plotting whereas value 1 has different style and so on.

Data parameter can be assigned with named variables or dataset that will be reshaped. Dataset is used hera because it contains all the vvalues of x,y and target which was created before.

Palette parameter is used to choose various colour options when plotting the hue.

plt.show() is used for displaying a figure

```
training_features, test_features, training_target, test_target = train_test_split(
    features, target, random_state=0)
```

train_test_split() method is used for splitting arrays or matrices into subsets which minimizes the potential for biasness while evaluating and validating process. Parameters used are features, target, and random state.

```
print(training_features.shape, test_features.shape)
```

```
(220, 4) (74, 4)
```

Dimension of training_features is shown by training_features.shape and dimension of test_features is shown by test_features.shape. The output shows that training data contains 150 rows and 4 columns whereas testing data contains 50 rows and 4 columns.

```
dtc = DecisionTreeClassifier(criterion='entropy')
dtc
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

It is used to classify the type of decision tree. The criterion is function to measure the quality of split using gini or entropy. Splitter is used to chosse the split at each node.

```
model = dtc.fit(training_features, training_target)
```

fit function is an estimator for classification which uses training_features and training_target as parameters

```
predictions = model.predict(test_features)
```

predict is used for predicting the test_features after the fit function has gained knowledge from the model.

```
matrix = confusion_matrix(test_target, predictions)
```

confusion_matrix generates confusion matrix in order to allow visualization of the performance of an algorithm i.e. in this case the classification accuracy. This confusion matrix deals with the actual result which is test_target and the predicted result which is the predictions.

```
print(matrix)
```

```
[[24  2  0]
 [ 1 23  0]
 [ 1  0 23]]
```

The confusion matrix helps in evaluating the performance of a classifier. For simpler explanatoin we can take a 2*2 confusion matrix: We have four cases here which are:

True Positive: If the actual value is true and predicted value is also true then this case is called True Positive.

False Positive:If the actual value is false and the predicted value is true then this case is called False Positive.(Type 1 error)

True Negative:If the actual value is false and the predicted value is also false then this case is called True Negative.

False Negative:If the actual value is true and the predicted result is false then this case is called False Negative. (Type 2 Error)

```
print(classification_report(test_target, predictions))
```

```
                  precision    recall  f1-score   support

               0       0.92      0.92      0.92        26
               1       0.92      0.96      0.94        24
               2       1.00      0.96      0.98        24

        accuracy                           0.95        74
       macro avg       0.95      0.95      0.95        74
    weighted avg       0.95      0.95      0.95        74
```

The precision, recall, F1, and support scores for the model is displayed by classification_report.

Precision measures the number of positive class predictions that actually belong to the positive class.

Recall measures the number of positive class predictions made out of all positive examples in the dataset

F1 score is the harmonic mean of precision and recall which conveys the balance between both

```
dot_data = tree.export_graphviz(
    model, out_file=None, feature_names=feature_names)
graph = pydotplus.graph_from_dot_data(dot_data)
graph.write_png('decision_tree.png')
```

```
True
```

✓ 0s completed at 12:10 PM