

**Banknote Authentication Prediction Using Decision Tree**

Machine learning project

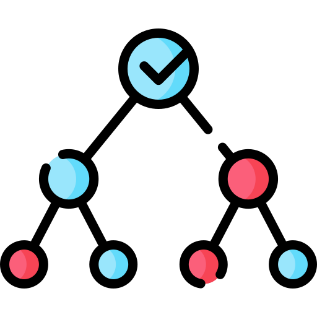
* Banknotes are currencies used by any nation to carry-out financial activities, so there is a need for an efficient authentication system which predicts accurately whether the given note is genuine or not.

Problem

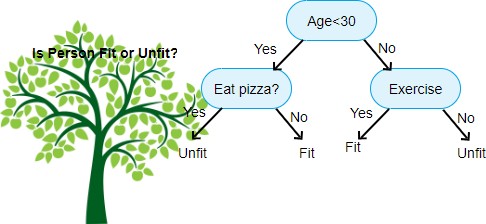
*  This model is predicting whether a given **banknote** is **authentic** given a number of measures taken from a photograph Using **decision tree classifier**.
* It is a binary (2-class) classification problem. (**0** for **authentic**, **1** for

# inauthentic).

* Decision Trees are a non-parametric supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](https://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features



Decision Tree

* In simple words, it can be understood as a machine learning model that asks some questions about the input features.
* These “questions” or rules determined by the tree and you can control on its depth and shape.
* Node splitting, or simply splitting, is the process of dividing a node into multiple sub-nodes to create relatively pure nodes. There are multiple ways of doing this, which can be broadly divided into two categories based on the type of target variable:
* **Continuous** Target Variable
  + Reduction in Variance
* **Categorical** Target Variable
  + Gini Impurity
  + Information Gain
  + Chi-Square
* Consider the following example ( OR Gate):



Decision Tree Example

0

F2

0

1

F1

1

1

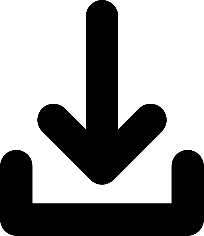
0

1

|  |  |  |
| --- | --- | --- |
| **F1** | **F2** | **class** |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

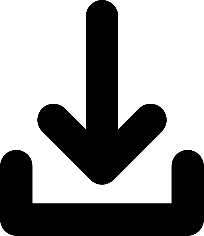
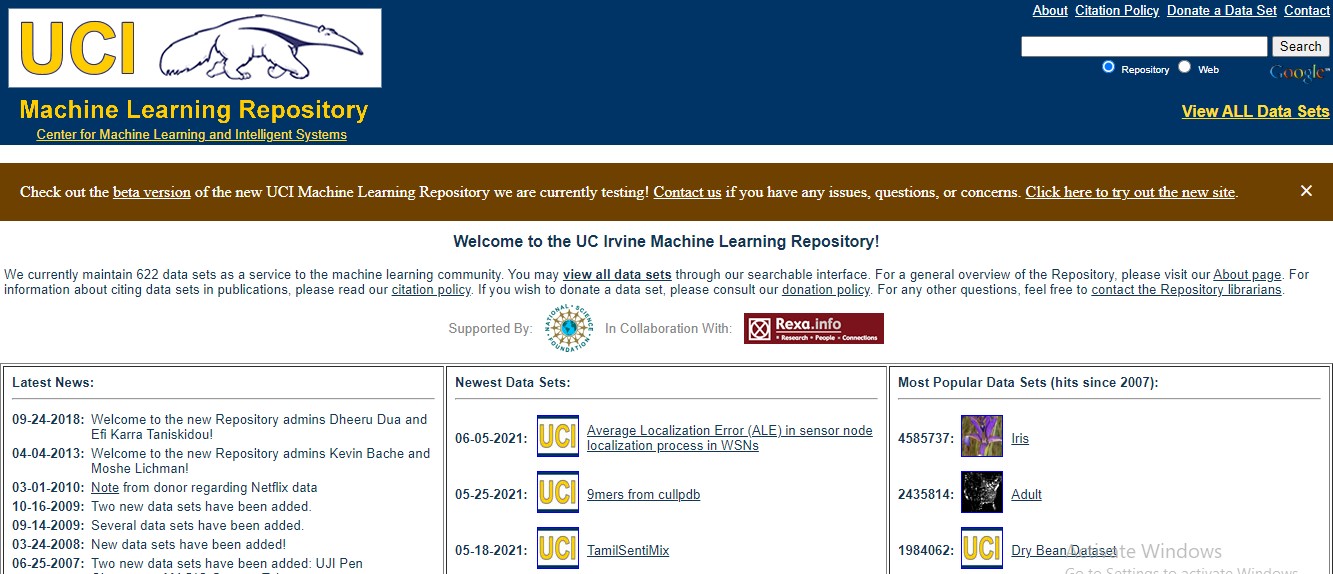
**Data Decision Tree**

#  From :UC Irvine Machine Learning Repository

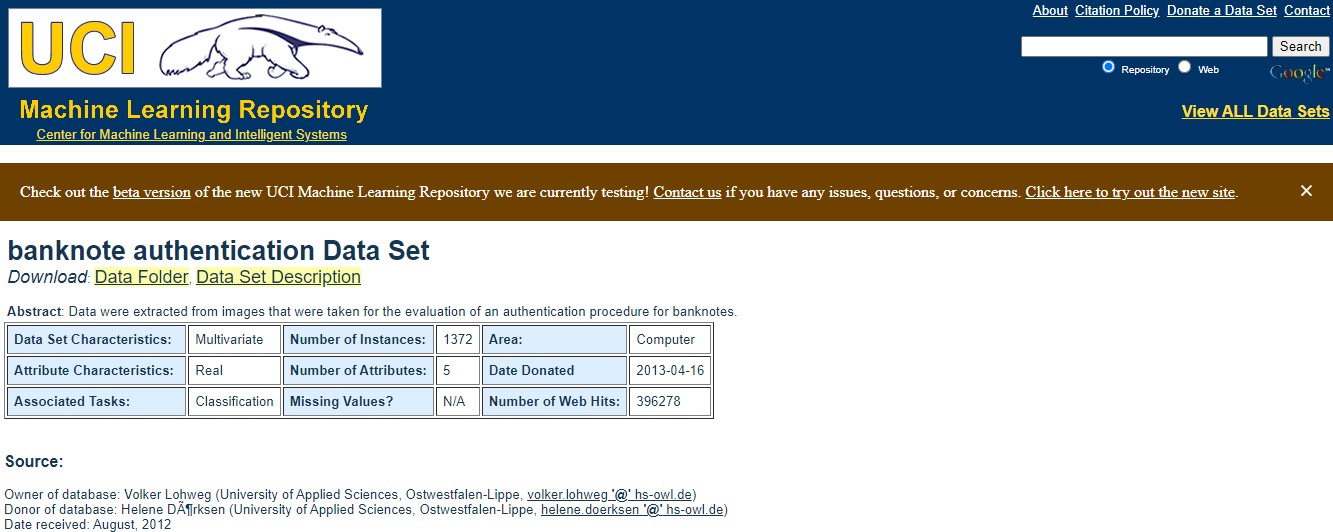
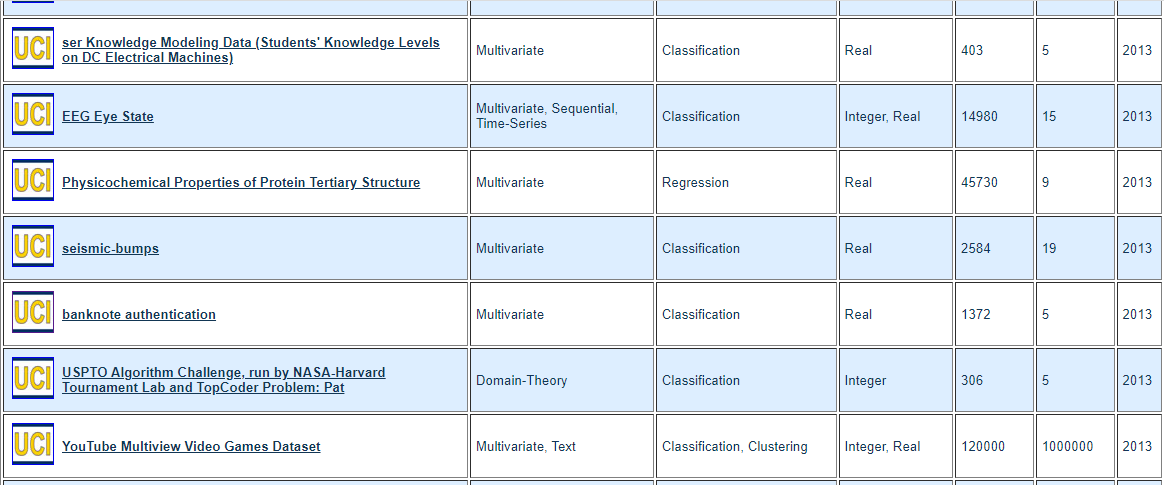


Data set Downloading

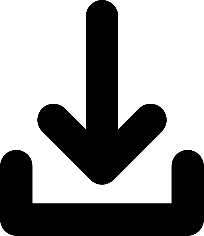
<https://archive.ics.uci.edu/ml/index.php>



Data set Downloading



* [data\_banknote\_authentication.txt](https://archive.ics.uci.edu/ml/machine-learning-databases/00267/data_banknote_authentication.txt) : Actual data file.

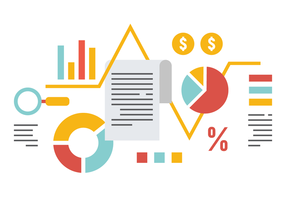


Data set Downloading

* Save the file as csv file on your PC.
* Now, data set is ready to use.

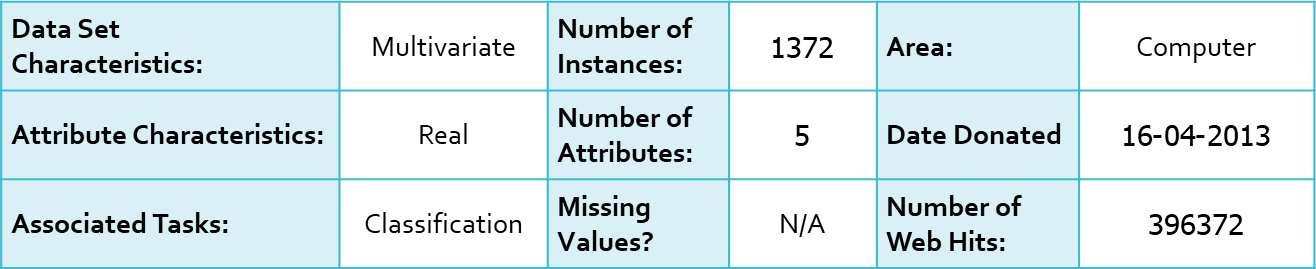


* + Data were extracted from images that were taken for the evaluation of an authentication procedure for banknotes.

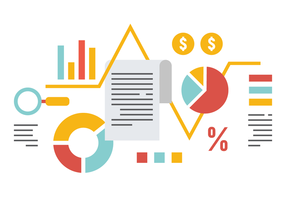


Data set Information

* + images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used.
  + The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object, gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.



* **Attribute Information:**



Data set Information

1. **Variance** of Wavelet Transformed image (continuous)
2. **Skewness** of Wavelet Transformed image (continuous)
3. **Kurtosis** of Wavelet Transformed image (continuous)
4. **Entropy** of image (continuous)
5. **Class** (integer)

* **Sample of the first three rows:**

3.6216,8.6661,-2.8073,-0.44699,0

4.5459,8.1674,-2.4586,-1.4621,0

3.866,-2.6383,1.9242,0.10645,0

* **Why we choose this Data Set ?**

The suggested data set is standard for the banknote authentication problem.in addition, with the proposed model, it gives a high accuracy score and produces a fitting model with good results.

* The first stage in the model is to read the dataset using the

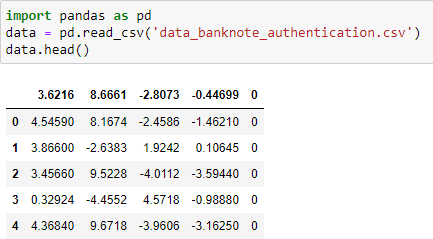


Data Preprocessing

Pandas library that deals with any dataset through:

import pandas as pd

* But we will notice a problem when reading the data, which is that this dataset does not contain the names of the columns of the data we have.



* so, we add them in the stage of reading the data through :



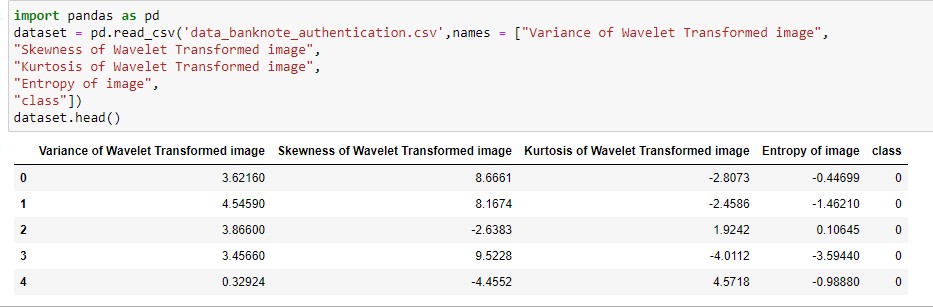
Data Preprocessing

**dataset = pd.read\_csv('data\_banknote\_authentication.csv',names =**

**[“Variance of Wavelet Transformed image", "Skewness of Wavelet Transformed image", "Kurtosis of Wavelet Transformed image", "Entropy of image",**

**"class"]).**

* To display only the first five rows of the dataset through:



* The second stage is the stage of dividing the data into input and output so that we know how to deal with it, through the drop function that separates the input from the output by giving it the columns that we want to delete from the data in order to make the input part alone through :



Data Preprocessing

dataset\_input = dataset.drop(columns = ['class'])

dataset\_output = dataset['class‘]

* And to display both the dataset\_input and the dataset\_output through:

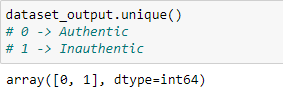
dataset\_input.head() dataset\_output.head()

* After that, to make sure if the output consists of only 2 classes or more, through a unique function that displays the values for me without any repetition in data by:

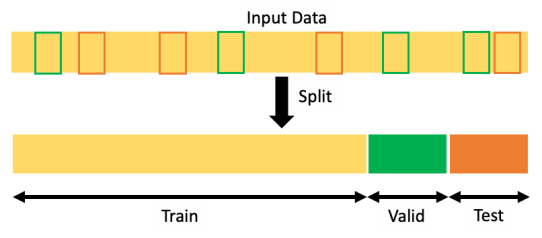


Data Preprocessing

dataset\_output.unique()



* After the preprocessing phase is over, we will divide the data into the training, validation and testing part in order to prepare the data for the model selection phase using the train\_test\_split function of the model selection class that you will call from the sklearn library.



Splitting the Data set

* The train\_test\_split function takes 4 parameter of the input, output and test\_size. This is specific to the testing ratio of data and random\_state. It is responsible for the stability of the accuracy of the training and testing data that are chosen randomly and the function returns for four variables as follows:

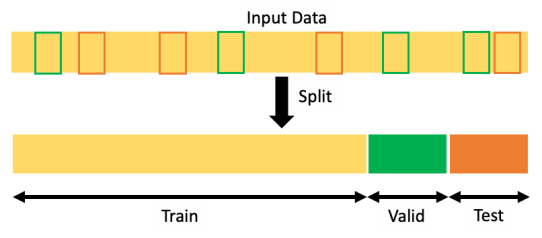
from sklearn.model\_selection import train\_test\_split

x,x\_test,y,y\_test = train\_test\_split(dataset\_input,dataset\_output,test\_size = 0.33,random\_state=2)

x\_train,x\_val,y\_train,y\_val =

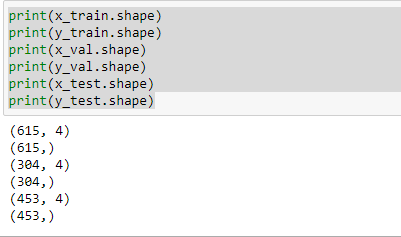
train\_test\_split(x,y,test\_size=0.33,random\_state=2)

* The next step is to display the shape of all the variables in order to know the number of rows and columns for all the variables through:

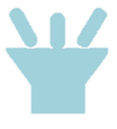


Splitting the Data set

print(x\_train.shape) print(y\_train.shape) print(x\_val.shape) print(y\_val.shape) print(x\_test.shape) print(y\_test.shape)



* Now we're going to move on to the stage of building the model,

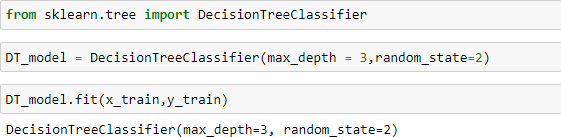


Model Selection (Defining and Training)

and that's going to happen by using model (max\_depth) from

(DecisionTreeClassifier)

* The best value of max\_depth is 3 , then we training model on data by using function called (model.fit).



* Accuracy :is one metric for evaluating classification models.



Model Selection (Validation)

Informally, **accuracy** is the fraction of predictions our model got right.

* For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:



* Where TP = True Positives, TN = True Negatives, FP = False

Positives, and FN = False Negatives.

* use accuracy in model code:



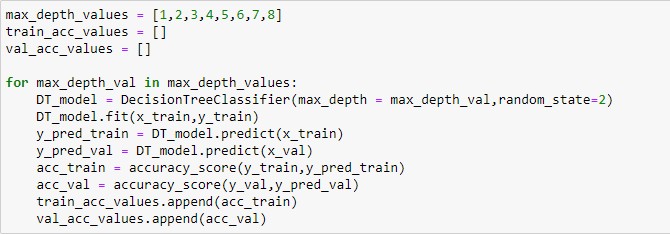
Model Selection (Validation)



1. import library:
2. Calculate accuracy (training and validation):



* In following code, the for loop calculates the accuracy when changing depth of the tree.in order to reach a high accuracy with training and testing data and to avoid over fitting and under fitting.

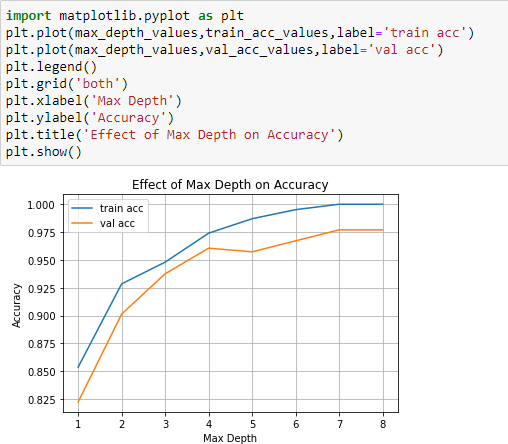


* matplotlib can be used to draw a chart to observe accuracy values with different depth of tree values.

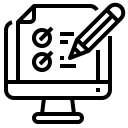


Model Selection (Validation)

* By observing the chart, we can specify that the best depth to avoid over fitting is = 3.

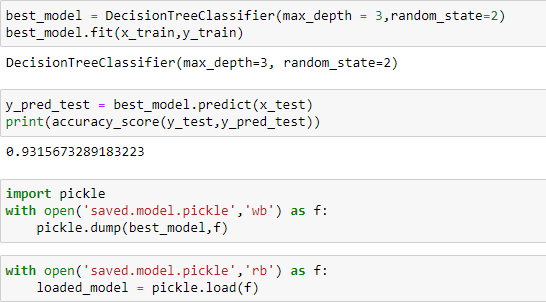


* At this step, we'll take object from the model that was already built. The model will be trained by giving it data its outputs are well known ,and that's what's called (**training**). and then we give the model data its outputs are unknown or unseen by the model before (**testing**).

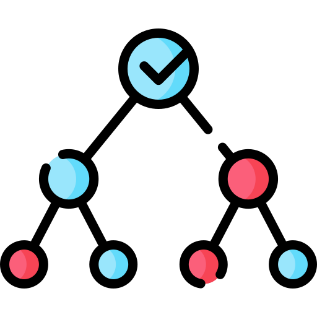


Model Selection (Testing)

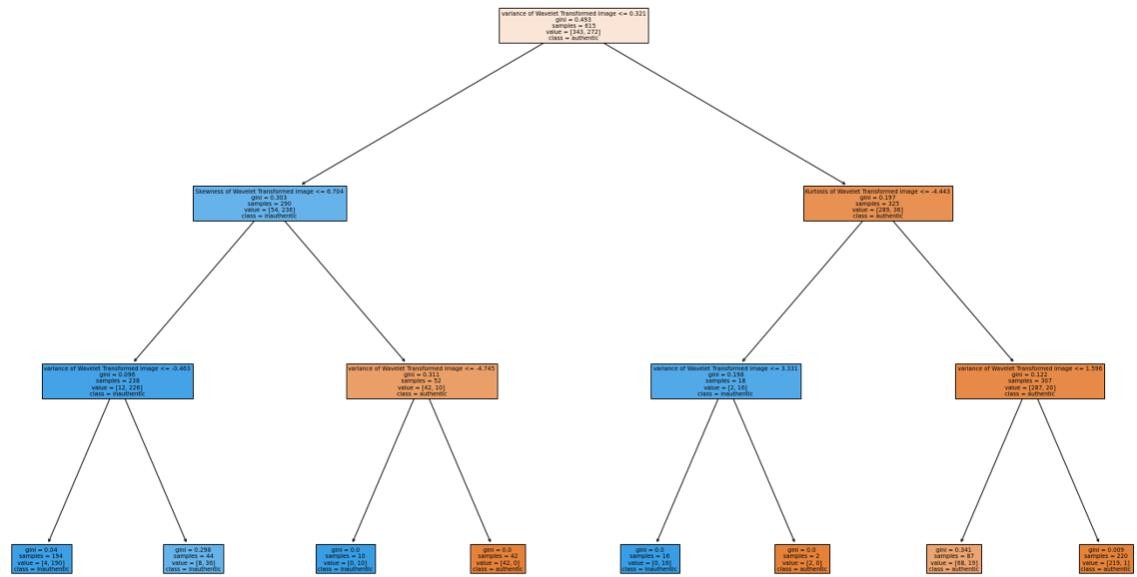
* accuracy is then calculated to see if we will change hyper parameters or not, and see if the performance is suitable for a test of the final model results.
* The final test accuracy is **0.9315673289183223**
* The model will be saved so that it can be used at any time it is needed without doing those previous steps and not repeating the training and validation again and that will happen through a package called (“pickle”)



* After that we can show an illustration of the tree through a function called )”tree.plot\_tree”).

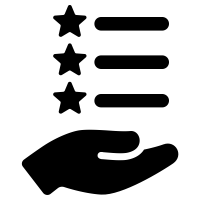


Decision Tree Visualization

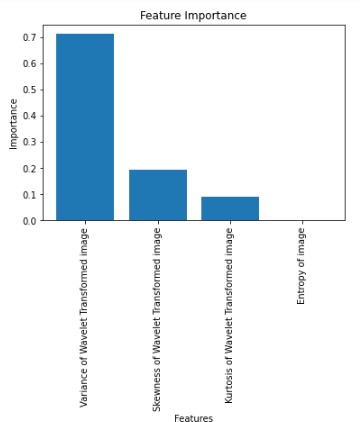
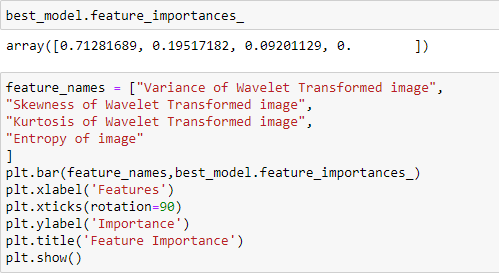




* In this step, an array is displayed through which the importance of each feature is known and whether all features are used or not, and if they are used, what is the percentage of their use?. that's done through (“feature\_importances”), then we view a graph of that array.



Features Importance

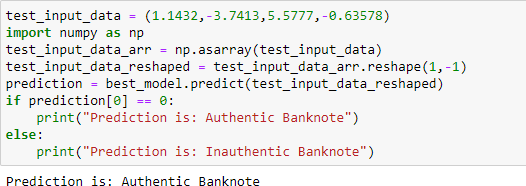


* Now, we test the final model prediction for a randomly selected



Building a Predictive Model

sample of data :



* The final model predict that the sample is for an **Authentic** Banknote ( belongs to class **0**). We can check the data set and see if the model predict the result correctly.

**Y correct prediction**