Supervised Machine Learning using Classification

For Supervised Machine learning using classification we are going to use Scikit-learn library.
Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language.
It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

We can install scikit-learn separately using pip or we can install anaconda which contains scikit-learn in it.

Components of scikit-learn :

Supervised learning algorithms:
Starting from Generalized linear models (e.g Linear Regression), Support Vector Machines (SVM), Decision Trees to Bayesian methods – all of them are part of scikit-learn toolbox.

Cross-validation:
There are various methods to check the accuracy of supervised models on unseen data

Unsupervised learning algorithms:
Again there is a large spread of algorithms in the offering – starting from clustering, factor analysis, principal component analysis to unsupervised neural networks.

Feature extraction:
Useful for extracting features from images and text (e.g. Bag of words)

Problem statement :

There are multiple types of balls in our data set as cricket ball and tennis ball.
This types of balls are classified on the basis of its weight and its surface.
We have to design application using machine learning strategy which is used to classify the balls.

Consider below data set which contains two features as Weight and Pattern.

Marvellous Infosystems Training Data set

Weight	Pattern	Label
35	Rough	Tennis
47	Rough	Tennis
90	Smooth	Cricket
48	Rough	Tennis
90	Smooth	Cricket
35	Rough	Tennis
92	Smooth	Cricket
35	Rough	Tennis
35	Rough	Tennis
35	Rough	Tennis
96	Smooth	Cricket
43	Rough	Tennis
110	Smooth	Cricket
35	Rough	Tennis
95	Smooth	Cricket

Tennis Ball

Cricket Ball

To deal with this problem statement we create below application which loads data set in our application

In this application we import tree module from sklearn library ie scikit-learn.
BallsFeatures list contains all examples from data set with weight and surface as a feature.

```
from sklearn import tree

BalllsFeatures = [[35,"Rough"],[47,"Rough"],[90,"Smooth"],[48,"Rough"],
    [90,"Smooth"],[35,1],[92,"Smooth"],[35,"Rough"],[35,"Rough"],
    [35,"Rough"],[96,"Smooth"],[43,"Rough"],[110,"Smooth"],[35,"Rough"
    ],[95,"Smooth"]]

Names =
    ["Tenis","Tenis","Cricket","Tenis","Cricket","Tenis","Cricket","Tenis","Teni
    s","Tenis","Cricket","Tenis","Cricket","Tenis","Cricket"]

clf = tree.DecisionTreeClassifier()

clf = clf.fit(BalllsFeatures,Names)

print(clf.predict([[44,1]]))

```

Names list contains expected labels which are based on data set.
As we can store integral values in list we can consider

Rough as 1
Smooth as 0

Tenis as 1
Cricket as 2

after replacing this values our application becomes

```
from sklearn import tree

BalllsFeatures = [[35,1],[47,1],[90,0],[48,1],[90,0],[35,1],[92,0],[35,1],
    [35,1],[35,1],[96,0],[43,1],[110,0],[35,1],[95,0]]

Names = [1,1,2,1,2,1,2,1,1,1,2,1,2,1,2]

clf = tree.DecisionTreeClassifier()

clf = clf.fit(BalllsFeatures,Names)

print(clf.predict([[44,1]]))

```

Output of above application

```
(base) MBPdeMARVELLOUS:Ball marvellous$ python3 cls
.py
[1]
(base) MBPdeMARVELLOUS:Ball marvellous$
```

Output displays value 1 means our object is identified as tennis ball.