# **Artificial Intelligence and Machine Learning Fundamentals**

**Activity 10**: Car Data Classification

This section will discuss how to build a reliable decision tree model capable of aiding your company in finding cars clients are likely to buy. We will be assuming that you are employed by a car rental agency focusing on building a lasting relationship with its clients. Your task is to build a decision tree model classifying cars into one of four categories: unacceptable, acceptable, good, very good.

The data set can be accessed here: [https://archive.ics.uci.edu/ml/datasets/ Car+Evaluation](https://archive.ics.uci.edu/ml/datasets/%20Car+Evaluation). Click the Data Folder link to download the data set. Click the Data

Set Description link to access the description of the attributes.

Evaluate the utility of your decision tree model.

1. Download the car data file from here: <https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data.> Add a header line to the front of the CSV file to reference it in Python more easily:

**Buying,Maintenance,Doors,Persons,LuggageBoot,Safety,Class**

We simply call the label Class. We named the six features after their descriptions in <https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.names>.

1. Load the data set into Python import pandas

data\_frame = pandas.read\_csv('car.data')

Let's check if the data got loaded correctly:

data\_frame.head()

Buying Maintenance Doors Persons LuggageBoot Safety Class

0 vhigh vhigh 2 2 small low unacc

1 vhigh vhigh 2 2 small med unacc

2 vhigh vhigh 2 2 small high unacc

3 vhigh vhigh 2 2 med low unacc

4 vhigh vhigh 2 2 med med unacc

1. As classification works with numeric data, we have to perform label encoding as seen in previous lesson.

labels = {

'Buying': ['vhigh', 'high', 'med', 'low'],

'Maintenance': ['vhigh', 'high', 'med', 'low'],

'Doors': ['2', '3', '4', '5more'],

'Persons': ['2', '4', 'more'],

'LuggageBoot': ['small', 'med', 'big'],

'Safety': ['low', 'med', 'high'],

'Class': ['unacc', 'acc', 'good', 'vgood']

}

from sklearn import preprocessing

label\_encoders = {}

data\_frame\_encoded = pandas.DataFrame()

for column in data\_frame:

if column in labels:

label\_encoders[column] = preprocessing.LabelEncoder()

label\_encoders[column].fit(labels[column])

data\_frame\_encoded[column] = label\_encoders[column].

transform(data\_frame[column])

else:

data\_frame\_encoded[column] = data\_frame[column]

1. Let's separate features from labels:

import numpy as np

features = np.array(data\_frame\_encoded.drop(['Class'], 1))

label = np.array( data\_frame\_encoded['Class'] )

1. It is time to separate training and testing data with the cross-validation (in newer versions model-selection) feature of scikit-learn. We will use 10% test data:

from sklearn import model\_selection

features\_train, features\_test, label\_train, label\_test = model\_

selection.train\_test\_split(

features,

label,

test\_size=0.1

)

Note that the train\_test\_split method will be available in model\_selection module, not in the cross\_validation module starting in scikit-learn 0.20. In previous versions, model\_selection already contains the train\_test\_split method.

1. We have everything to build the decision tree classifier:

from sklearn.tree import DecisionTreeClassifier

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(features\_train, label\_train)

The output of the fit method is as follows:

DecisionTreeClassifier(

class\_weight=None,

criterion='gini',

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=False,

random\_state=None,

splitter='best'

)

You can see the parametrization of the decision tree classifier. There are quite a few options we could set to tweak the performance of the classifier model.

1. Let's score our model based on the test data:

decision\_tree.score( features\_test, label\_test )

The output is as follows:

0.9884393063583815

1. This is the point where your knowledge up until lesson 4 would take you on model evaluation. We will now go a bit further and create a deeper evaluation of the model based on the classification\_report feature we learned in this topic:

from sklearn.metrics import classification\_report

print(

classification\_report(

label\_test,

decision\_tree.predict(features\_test)

)

)

The output is as follows:

Precision recall f1-score support

0 0.97 0.97 0.97 36

1 1.00 1.00 1.00 5

2 1.00 0.99 1.00 127

3 0.83 1.00 0.91 5

avg / total 0.99 0.99 0.99 173

The model has been proven to be quite accurate. In case of such a high accuracy score, suspect the possibility of overfitting.