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. 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.

2. 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

3. 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]
```

4. 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'] )
```

5. 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.

6. 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.

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

```
decision_tree.score( features_test, label_test )
The output is as follows:
0.9884393063583815
```

8. 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.