Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τομέας Ηλεκτρονικής και Υπολογιστών

9ο Εξάμηνο
Φθινόπωρο 2021

Δένδρα Απόφασης στην Python

1 Εισαγωγή

1.1 Εισαγωγή στα Δένδρα Απόφασης

Το δένδρο απόφασης είναι ένας αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται κυρίως σε προβλήματα ταξινόμησης. Εφαρμόζεται σε κατηγορικά και συνεχή δεδομένα. Η βασική ιδέα του αλγορίθμου είναι ο διαδοχικός διαχωρισμός των δεδομένων με βάση κανόνες που κατασκευάζονται σύμφωνα με κάποιο κριτήριο διαχωρισμού.

Τα δένδρα απόφασης είναι υπολογιστικά εύκολα στην κατασκευή και πολύ γρήγορα στην ταξινόμηση νέων εγγραφών. Επιπλέον είναι αρκετά κατανοητά για δένδρα μικρού μεγέθους και για αυτό χρησιμοποιούνται πολλές φορές για εξερεύνηση των δεδομένων. Ένα βασικό μειονέκτημα είναι ότι ενδέχεται να παρουσιαστεί overfitting, το οποίο αντιμετωπίζεται κάνοντας pruning ή γενικότερα θέτοντας περιορισμούς κατά την κατασκευή του δένδρου.

1.2 Δένδρα Απόφασης στην Python

Για την κατασκευή δένδρων απόφασης στην Python, μπορούμε να χρησιμοποιήσουμε τη βιβλιοθήκη `sklearn` και το component `tree`:

```
>>> from sklearn import tree
```

Επιπλέον, χρησιμοποιούμε και τη βιβλιοθήκη `matplotlib` και το component `pyplot` με το οποίο μπορούμε να σχεδιάσουμε τα παραγόμενα δένδρα:

```
>>> import matplotlib.pyplot as plt
```

Για να κατασκευάσουμε ένα δένδρο απόφασης τρέχουμε την εντολή:

```
>>> clf = tree.DecisionTreeClassifier()
```

Και για να κάνουμε fit το δέντρο στα δεδομένα:

```
>>> clf = clf.fit(features, target)
```

Για να κάνουμε plot το δένδρο μπορούμε να τρέξουμε την `plot_tree(model, class_names)`.

2 Κριτήρια Διαχωρισμού και Κατασκευή Δένδρων Απόφασης

Για τον υπολογισμό κριτηρίων διαχωρισμού και την κατασκευή δένδρου απόφασης θα χρησιμοποιήσουμε ως εφαρμογή τα δεδομένα εκπαίδευσης του παρακάτω πίνακα για ένα πρόβλημα δυαδικής ταξινόμησης.

Outlook	Temperature	Humidity	Play
Sunny	Hot	High	No
Sunny	Hot	Low	No
Rainy	Hot	Low	Yes
Rainy	Cool	High	Yes
Rainy	Cool	Low	Yes
Rainy	Hot	Low	No
Rainy	Cool	Low	Yes
Sunny	Hot	High	No
Sunny	Cool	Low	Yes
Rainy	Hot	Low	Yes
Sunny	Cool	Low	Yes
Rainy	Hot	High	Yes
Rainy	Cool	Low	Yes
Sunny	Cool	High	No

Στο παραπάνω dataset επιθυμούμε να εφαρμόσουμε έναν αλγόριθμο δένδρου απόφασης. Θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Ποιο χαρακτηριστικό από τα Outlook, Temperature, Humidity είναι καλύτερο να χρησιμοποιηθεί στον πρώτο διαχωρισμό, με βάση το δείκτη Gini;

β) Ποιο χαρακτηριστικό από τα Outlook, Temperature, Humidity είναι καλύτερο να χρησιμοποιηθεί στον πρώτο διαχωρισμό, με βάση το δείκτη κέρδους πληροφορίας;

γ) Κατασκευάστε και κάντε plot το πλήρες δένδρο απόφασης για το παραπάνω πρόβλημα με βάση το δείκτη Gini.

2.1 Εισαγωγή Δεδομένων και Βιβλιοθηκών

Αρχικά διαβάζουμε τα δεδομένα και φορτώνουμε τις απαραίτητες βιβλιοθήκες:

```
>>> import pandas as pd
>>> from sklearn import tree
>>> from sklearn.preprocessing import OneHotEncoder
>>> import matplotlib.pyplot as plt
>>> weather = pd.read_csv("./weather.txt")
```

2.2 Κριτήρια Διαχωρισμού

Για να δούμε το διαχωρισμό για τη μεταβλητή Outlook τρέχουμε την παρακάτω εντολή:

```
>>> encoder.fit(weather.loc[:, ['Outlook']])

>>> transformedOutlook = encoder.transform(weather.loc[:,
['Outlook']])

>>> clf = tree.DecisionTreeClassifier()

>>> clf = clf.fit(transformedOutlook, weather.loc[:, 'Play'])
```

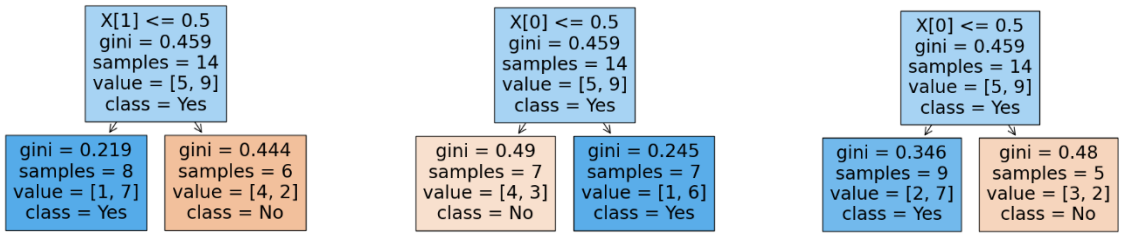
Αντίστοιχα, αν τρέξουμε και για τις Temperature και Humidity, και κάνουμε plot την καθεμία με τις παρακάτω εντολές:

```
>>> fig = plt.figure()

>>> tree.plot_tree(clf, class_names=['No', 'Yes'], filled=True)

>>> plt.show()
```

προκύπτουν τα παρακάτω σχήματα:



Εμπειρικά, μπορούμε ίσως ήδη να καταλάβουμε ποιος από τους τρεις διαχωρισμούς πρέπει να επιλεγεί για το πρώτο επίπεδο;

2.2.1 Gini Index

Υπολογίζουμε το Gini index για το Outlook με τους παρακάτω τύπους:

$$\boldsymbol{GINI}(\boldsymbol{Sunny}) = 1 - Freq(Play = No|Outlook = Sunny)^2 - Freq(Play = Yes|Outlook = Sunny)^2$$
$$= 1 - (4/6)^2 - (2/6)^2 = 0.444$$

$$\boldsymbol{GINI}(\boldsymbol{Rainy}) = 1 - Freq(Play = No|Outlook = Rainy)^2 - Freq(Play = Yes|Outlook = Rainy)^2$$
$$= 1 - (1/8)^2 - (7/8)^2 = 0.219$$

$$\boldsymbol{GINI_{Outlook}} = Freq(Outlook = Sunny) \cdot GINI(Sunny) + Freq(Outlook = Rainy) \cdot GINI(Rainy)$$
$$= (6/14) \cdot 0.444 + (8/14) \cdot 0.219 = 0.315$$

Αντίστοιχα, για τα Temperature και Humidity, το GINI είναι $\boldsymbol{GINI_{Temperature}} = 0.367$ και $\boldsymbol{GINI_{Humidity}} = 0.394$. Άρα, απαντώντας στο ερώτημα (α), ο βέλτιστος πρώτος διαχωρισμός σύμφωνα με το Gini index είναι στο Outlook.

Μπορούμε επίσης να κάνουμε τους υπολογισμούς χρησιμοποιώντας την Python. Για το Outlook, κατασκευάζουμε τους παρακάτω πίνακες συχνοτήτων:

```
>>> absfreq = pd.crosstab(weather.Outlook, weather.Play)

>>> freq = pd.crosstab(weather.Outlook, weather.Play,
normalize='index')

>>> freqSum = pd.crosstab(weather.Outlook, weather.Play,
normalize='all').sum(axis=1)
```

Υπολογίζουμε το Gini index για το Sunny και το Rainy:

```
>>> GINI_Sunny = 1 - freq.loc["Sunny", "No"]**2 - freq.loc["Sunny",
"Yes"]**2

>>> GINI_Rainy = 1 - freq.loc["Rainy", "No"]**2 - freq.loc["Rainy",
"Yes"]**2
```

Το συνολικό Gini για το Outlook υπολογίζεται με την εντολή:

```
>>> GINI_Outlook = freqSum.loc["Sunny"] * GINI_Sunny +
freqSum["Rainy"] * GINI_Rainy

>>> print(GINI_Outlook)
```

2.2.2 Information Gain

Υπολογίζουμε το Information Gain για το Outlook με τους παρακάτω τύπους:

$$\boldsymbol{Entropy(All)} = -Freq(No) \cdot lg(Freq(No)) - Freq(Yes) \cdot lg(Freq(Yes))$$
$$= -(5/14) \cdot lg(5/14) - (9/14) \cdot lg(9/14) = 0.940$$

$$\boldsymbol{Entropy(Sunny)} = -Freq(No|Sunny) \cdot lg(Freq(No|Sunny)) - Freq(Yes|Sunny) \cdot lg(Freq(Yes|Sunny))$$
$$= -(4/6) \cdot lg(4/6) - (2/6) \cdot lg(2/6) = 0.918$$

$$\boldsymbol{Entropy(Rainy)} = -Freq(No|Rainy) \cdot lg(Freq(No|Rainy)) - Freq(Yes|Rainy) \cdot lg(Freq(Yes|Rainy))$$
$$= -(1/8) \cdot lg(1/8) - (7/8) \cdot lg(7/8) = 0.544$$

$$\boldsymbol{GAIN_{Outlook}} = Entropy(All) - Freq(Sunny) \cdot Entropy(Sunny) - Freq(Rainy) \cdot Entropy(Rainy)$$
$$= 0.940 - (6/14) \cdot 0.918 + (8/14) \cdot 0.544 = 0.236$$

Αντίστοιχα, για τα Temperature και Humidity, το GAIN είναι $GAIN_{Temperature} = 0.152$ και $GAIN_{Humidity} = 0.102$. Άρα, απαντώντας στο ερώτημα (β), ο βέλτιστος πρώτος διαχωρισμός σύμφωνα με το κέρδος πληροφορίας είναι στο Outlook.

Μπορούμε επίσης να κάνουμε τους υπολογισμούς χρησιμοποιώντας την Python. Αρχικά υπολογίζουμε την εντροπία για το σύνολο των δεδομένων:

```
>>> import math

>>> freq = pd.crosstab("Play", weather.Play, normalize="index")

>>> EntropyAll = - freq.No * math.log2(freq.No) – freq.Yes *
math.log2(freq.Yes)
```

Κατόπιν, για το Outlook κατασκευάζουμε τους παρακάτω πίνακες συχνοτήτων:

```
>>> absfreq = pd.crosstab(weather.Outlook, weather.Play)

>>> freq = pd.crosstab(weather.Outlook, weather.Play,
normalize='index')

>>> freqSum = pd.crosstab(weather.Outlook, weather.Play,
normalize='all').sum(axis=1)
```

Υπολογίζουμε την εντροπία για το Sunny και το Rainy:

```
>>> EntropySunny = - freq.loc['Sunny', 'No'] *
math.log2(freq.loc['Sunny', 'No']) - freq.loc['Sunny', 'Yes'] *
math.log2(freq.loc['Sunny', 'Yes'])

>>> EntropyRainy = - freq.loc['Rainy', 'No'] *
math.log2(freq.loc['Rainy', 'No']) - freq.loc['Rainy', 'Yes'] *
math.log2(freq.loc['Rainy', 'Yes'])
```

Το συνολικό κέρδος πληροφορίας για το Outlook υπολογίζεται με την εντολή:

```
>>> GAINOutlook = EntropyAll - freqSum.loc['Sunny'] * EntropySunny -
freqSum.loc['Rainy'] * EntropyRainy
```

2.3 Κατασκευή Δένδρου Απόφασης

Για να κατασκευάσουμε ένα πλήρες δένδρο απόφασης (ερώτημα (γ)) τρέχουμε την DecisionTreeClassifier:

```
>>> encoder = OneHotEncoder(handle_unknown="ignore", sparse=False)

>>> encoder.fit(weather.loc[:, ['Outlook', 'Temperature',
'Humidity']])

>>> transformed = encoder.transform(weather.loc[:, ['Outlook',
'Temperature', 'Humidity']])

>>> clf = tree.DecisionTreeClassifier()

>>> clf = clf.fit(transformed, weather.loc[:, 'Play'])
```

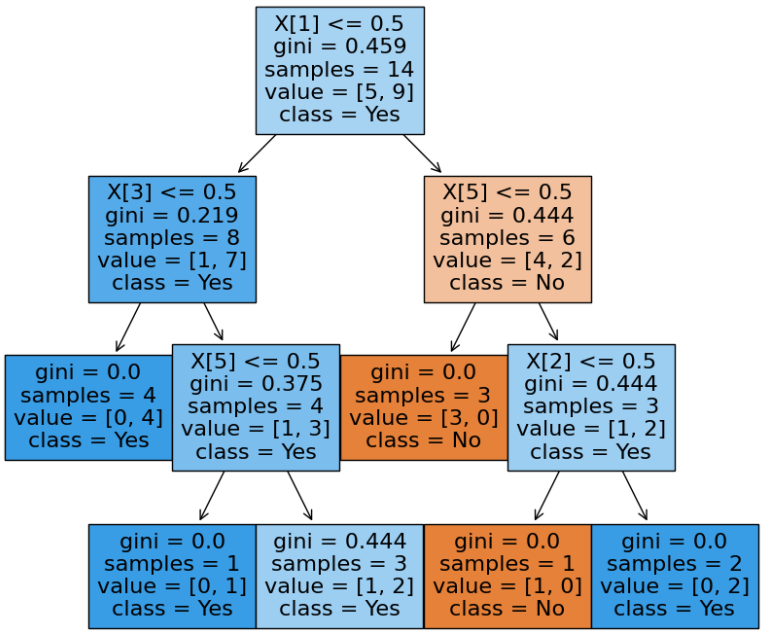
Επίσης, για να κάνουμε plot το δένδρο μπορούμε να τρέξουμε τις παρακάτω εντολές:

```
>>> fig = plt.figure(figsize=(10, 9))

>>> tree.plot_tree(clf, class_names=['No', 'Yes'], filled=True)

>>> plt.show()
```

Τελικά προκύπτει το παρακάτω δένδρο:



Μπορούμε να τυπώσουμε το δένδρο και σε μορφή κειμένου με τις παρακάτω εντολές:

```
>>> text_representation = tree.export_text(clf)
```

```
>>> print(text_representation)
```

Οπότε προκύπτει το παρακάτω δένδρο:

```
|--- feature_0 <= 0.50
|   |--- feature_2 <= 0.50
|   |   |--- class: No
|   |--- feature_2 >  0.50
|   |   |--- feature_4 <= 0.50
|   |   |   |--- class: Yes
|   |   |--- feature_4 >  0.50
|   |   |   |--- class: No
|--- feature_0 >  0.50
|   |--- feature_3 <= 0.50
|   |   |--- class: Yes
|   |--- feature_3 >  0.50
|   |   |--- feature_4 <= 0.50
|   |   |   |--- class: Yes
|   |   |--- feature_4 >  0.50
|   |   |   |--- class: Yes
```

Για να προβλέψουμε την κλάση στην οποία ταξινομείται ένα νέο δείγμα, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση `predict` και `predict_proba`:

```
>>> new_data = pd.DataFrame({“Outlook”: [“Sunny”], “Temperature”: [“Cold”], “Humidity”: [“High”]})
>>> transformed_new_data = encoder.transform(new_data)
>>> print(clf.predict(transformed_new_data))
>>> print(clf.predict_proba(transformed_new_data))
```

3 Εφαρμογή με Pruning και Μετρικές Αξιολόγησης

Ένας ταξινομητής δένδρο απόφασης μπορεί να εφαρμοστεί επίσης σε datasets με συνεχείς μεταβλητές, όπως το iris dataset που θα χρησιμοποιήσουμε ως εφαρμογή. Αρχικά, εισάγουμε το dataset, επιλέγοντας ωστόσο μόνο τις 2 πρώτες στήλες και αλλάζοντας τις τελευταίες 50 τιμές του Species ώστε να κάνουμε το πρόβλημα binary classification:

```
>>> from sklearn import datasets
>>> import numpy as np
>>> iris = datasets.load_iris()
>>> data = iris.data[:, [0, 1]]
>>> target = iris.target
>>> target[100:125] = 0
>>> target[125:150] = 1
```

Κατόπιν, κάνουμε split το dataset σε training και testing data:

```
>>> xtrain = np.concatenate((data[0:40], data[50:90], data[100:140]))
>>> ytrain = np.concatenate((target[0:40], target[50:90],
target[100:140]))
>>> xtest = np.concatenate((data[40:50], data[90:100],
data[140:150]))
>>> ytest = np.concatenate((target[40:50], target[90:100],
target[140:150]))
```

Στη συνέχεια, θα απαντήσουμε στα παρακάτω ερωτήματα:

α) Κατασκευάστε το δένδρο απόφασης χρησιμοποιώντας τα δεδομένα εκπαίδευσης (με την παράμετρο min_samples_split της DecisionTreeClassifier ίση με 20).

β) Εφαρμόστε το μοντέλο στα test data και υπολογίστε precision, recall και f-measure για τις δύο κλάσεις.

γ) Κατασκευάστε τα δένδρα για min_samples_split ίσο με 10 και για min_samples_split ίσο με 30 και υπολογίστε precision, recall και f-measure για τις δύο κλάσεις αφού τα εφαρμόσετε στα test data.

δ) Συγκρίνετε τα τρία μοντέλα ως προς το f-measure για την κλάση versicolor.

3.1 Κατασκευή και Εφαρμογή Δένδρου Απόφασης

Κάνουμε training το δένδρο και το σχεδιάζουμε με τις παρακάτω εντολές (ερώτημα (α), (γ)):

```
>>> clf = tree.DecisionTreeClassifier(min_samples_split=20)
>>> clf = clf.fit(xtrain, ytrain)
```

Μπορούμε στη συνέχεια να εκτελέσουμε το δένδρο στο test set:

```
>>> pred = clf.predict(xtest)
```

3.2 Υπολογισμός Μετρικών Αξιολόγησης

Μπορούμε να δούμε το confusion matrix και να υπολογίσουμε χρήσιμες μετρικές με τις παρακάτω εντολές (ερώτημα (β), (γ)):

```
>>> from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
>>> print(confusion_matrix(ytest, pred))
>>> print(accuracy_score(ytest, pred))
>>> print(precision_score(ytest, pred, pos_label=1))
>>> print(recall_score(ytest, pred, pos_label=1))
>>> print(f1_score(ytest, pred, pos_label=1))
```

(Εναλλακτικά μπορούμε να υπολογίσουμε τα TP, FP, TN, FN και να υπολογίσουμε το precision ως TP/(TP + FP) και το recall ως TP/(TP + FN))

Τέλος, το f-measure για τα 3 μοντέλα φαίνεται στον πίνακα (ερώτημα (δ)):

Decision Tree	F-Measure
minsplit = 10	0.865
minsplit = 20	0.923
minsplit = 30	0.950