Aprendizagem 2023
Homework I – Group 28

Gonçalo Bárias (ist1103124) & Raquel Braunschweig (ist1102624)

Part I: Pen and Paper

Consider the partially learnt decision tree from the dataset D. D is described by four input variables – one numeric with values in $[0, 1]$ and 3 categorical – and a target variable with three classes.

D	y_1	y_2	y_3	y_4	y_{out}
$\mathbf{x}_1$	0.24	1	1	0	A
$\mathbf{x}_2$	0.06	2	0	0	B
$\mathbf{x}_3$	0.04	0	0	0	B
$\mathbf{x}_4$	0.36	0	2	1	C
$\mathbf{x}_5$	0.32	0	0	2	C
$\mathbf{x}_6$	0.68	2	2	1	A
$\mathbf{x}_7$	0.9	0	1	2	A
$\mathbf{x}_8$	0.76	2	2	0	A
$\mathbf{x}_9$	0.46	1	1	1	B
$\mathbf{x}_{10}$	0.62	0	0	1	B
$\mathbf{x}_{11}$	0.44	1	2	2	C
$\mathbf{x}_{12}$	0.52	0	2	0	C

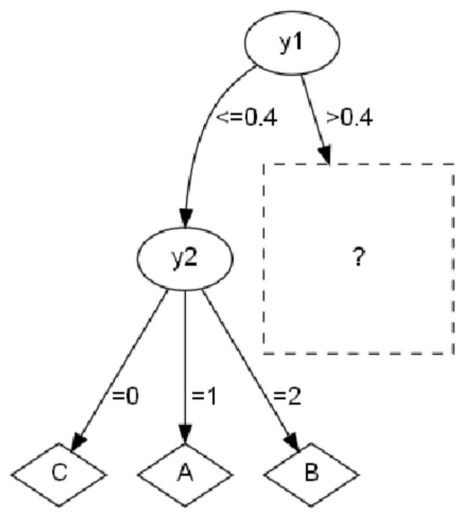

Figure 1: Partially Learnt Decision Tree and Dataset D from Part I

1. **Complete the given decision tree using Information gain with Shannon entropy (log_2). Consider that: i) a minimum of 4 observations is required to split an internal node, and ii) decisions by ascending alphabetic order should be placed in case of ties.**

   The entropy of y_{out} is given by:

$$\begin{aligned} E(y_{out}|y_1 > 0.4) = \quad & p(\mathrm{A}, y_1 > 0.4)\log_2(p(\mathrm{A}, y_1 > 0.4)) \\ & + p(\mathrm{B}, y_1 > 0.4)\log_2(p(\mathrm{B}, y_1 > 0.4)) \\ & + p(\mathrm{C}, y_1 > 0.4)\log_2(p(\mathrm{C}, y_1 > 0.4)) \end{aligned} \tag{1}$$

   We can calculate $E(y_{out})$:

$$\begin{aligned} E(y_{out}) &= -\left(\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right)\right) \\ &= 1.5567 \end{aligned}$$

The next step is calculating $E(y_{out}|y_1 > 0.4, y_x)$, in which x will take the values of 2, 3 or 4:

$$\begin{aligned} E(y_{out}|y_1 > 0.4, y_x) = \quad & p(y_x = 0)E(y_{out}|y_x > 0.4, y_2 = 0) \\ & + p(y_x = 1)E(y_{out}|y_x > 0.4, y_2 = 1) \\ & + p(y_x = 2)E(y_{out}|y_x > 0.4, y_2 = 2) \end{aligned} \tag{2}$$

And the information gain of variable y_x is given by

$$IG(y_x) = E(y_{out}) - E(y_{out}|y_1 > 0.4, y_x) \tag{3}$$

Let's start with x = 2:

$$\begin{aligned} p(y_2 = 0, y_1 > 0.4) &= \frac{3}{7} \\ p(y_2 = 1, y_1 > 0.4) &= \frac{2}{7} \\ p(y_2 = 1, y_1 > 0.4) &= \frac{2}{7} \\ E(y_{out}|y_x > 0.4, y_2 = 0) &= -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right) = 1.5849 \\ E(y_{out}|y_x > 0.4, y_2 = 1) &= -\left(\frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1 \\ E(y_{out}|y_x > 0.4, y_2 = 2) &= -\left(\frac{2}{2}\log_2\left(\frac{2}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right)\right) = 0 \end{aligned}$$

Therefore, replacing these values on equation (2), gives us:

$$\begin{aligned} E(y_{out}|y_1 > 0.4, y_2) &= \frac{3}{7} \times 1.5849 + \frac{2}{7} \times 1 + \frac{2}{7} \times 0 \\ &= 0.965. \end{aligned}$$

Finally, we can calculate the information gain, as per (3),

$$IG(y_2) = 1.5567 - 0.965 = 0.5917$$

Now, let's calculate for x = 3:

$$
\begin{aligned}
p(y_3 = 0, y_1 > 0.4) &= \frac{1}{7} \\
p(y_3 = 1, y_1 > 0.4) &= \frac{2}{7} \\
p(y_3 = 2, y_1 > 0.4) &= \frac{4}{7} \\
E(y_{out}|y_x > 0.4, y_3 = 0) &= -\left(\frac{0}{1}\log_2\left(\frac{0}{1}\right) + \frac{1}{1}\log_2\left(\frac{1}{1}\right) + \frac{0}{1}\log_2\left(\frac{0}{1}\right)\right) = 0 \\
E(y_{out}|y_x > 0.4, y_3 = 1) &= -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right)\right) = 1 \\
E(y_{out}|y_x > 0.4, y_3 = 2) &= -\left(\frac{2}{4}\log_2\left(\frac{2}{4}\right) + \frac{0}{4}\log_2\left(\frac{0}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 1
\end{aligned}
$$

Therefore, replacing these values on equation (2), gives us:

$$
\begin{aligned}
E(y_{out}|y_1 > 0.4, y_3) &= \frac{1}{7} \times 0 + \frac{2}{7} \times 1 + \frac{4}{7} \times 1 \\
&= 0.8571.
\end{aligned}
$$

Finally, we can calculate the information gain, as per (3),

$$IG(y_3) = 1.5567 - 0.8571 = 0.6996$$

Finally, let's calculate for x = 4:

$$
\begin{aligned}
p(y_4 = 0, y_1 > 0.4) &= \frac{2}{7} \\
p(y_4 = 1, y_1 > 0.4) &= \frac{3}{7} \\
p(y_4 = 1, y_1 > 0.4) &= \frac{2}{7} \\
E(y_{out}|y_x > 0.4, y_4 = 0) &= -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{3}\right)\right) = 1 \\
E(y_{out}|y_x > 0.4, y_4 = 1) &= -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{0}{3}\log_2\left(\frac{0}{3}\right)\right) = 0.9183 \\
E(y_{out}|y_x > 0.4, y_4 = 2) &= -\left(\frac{1}{2}\log_2\left(\frac{1}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) = 1
\end{aligned}
$$

Therefore, replacing these values on equation (2), gives us:

$$
\begin{aligned}
E(y_{out}|y_1 > 0.4, y_4) &= \frac{2}{7} \times 1.5849 + \frac{3}{7} \times 1 + \frac{2}{7} \times 0 \\
&= 0.965.
\end{aligned}
$$

Finally, we can calculate the information gain, as per (3),

$$IG(y_4) = 1.5849 - 0.965 = 0.5917$$

Upon computing the information gains for each attribute, it is evident that y_3 yields the highest value of 0.6996. Consequently, it is selected as the next node, resulting in the construction of the following decision tree:

Por arvore aqui (ns bem como fazer help)

2. **Draw the training confusion matrix for the learnt decision tree.**

   Blah

3. **Identify which class has the lowest training F1 score.**

   Blah

4. **Considering y_2 to be ordinal, assess if y_1 and y_2 are correlated using the Spearman coefficient.**

   Blah

5. **Draw the class-conditional relative histograms of y_1 using 5 equally spaced bins in $[0, 1]$. Find the root split using the discriminant rules from these empirical distributions.**

   Blah

Part II: Programming

Consider the `column_diagnosis.arff` data available at the homework tab, comprising 6 biomechanical features to classify 310 orthopaedic patients into 3 classes (`normal`, `disk hernia`, `spondilolysthesis`).

1. **Apply `f_classif` from `sklearn` to assess the discriminative power of the input variables. Identify the input variable with the highest and lowest discriminative power. Plot the class-conditional probability density functions of these two input variables.**

```python
import numpy as np, matplotlib.pyplot as plt, pandas as pd
from scipy.io.arff import loadarff
from sklearn.feature_selection import f_classif

# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
X, y = df.drop("class", axis=1), df["class"]

# Apply f_classif
f_scores, p_values = f_classif(X, y)

# Obtains the variables with the highest and lowest discriminative power.
```

```
highest_discriminative_power_idx = np.argmax(f_scores)
lowest_discriminative_power_idx = np.argmin(f_scores)

highest_discriminative_power_variable = X.columns[
    highest_discriminative_power_idx
]
lowest_discriminative_power_variable = X.columns[
    lowest_discriminative_power_idx
]

# Identifies the input variables requested
print(
    f"Highest discriminative power variable: {highest_discriminative_power_variable}"
)
print(
    f"Lowest discriminative power variable: {lowest_discriminative_power_variable}"
)

plt.figure(figsize=(10, 6))

# Plot for the highest discriminative power variable
for class_label in np.unique(y):
    class_data = X.loc[y == class_label, highest_discriminative_power_variable]
    density, bins = np.histogram(class_data, bins=20, density=True)
    plt.plot(
        bins[:-1],
        density,
        label=f"Class {class_label} - {highest_discriminative_power_variable}",
        linewidth=2,
    )

# Plot for the lowest discriminative power variable
for class_label in np.unique(y):
    class_data = X.loc[y == class_label, lowest_discriminative_power_variable]
    density, bins = np.histogram(class_data, bins=20, density=True)
    plt.plot(
        bins[:-1],
        density,
        linestyle="--",
        label=f"Class {class_label} - {lowest_discriminative_power_variable}",
        linewidth=2,
    )

plt.xlabel("Value")
plt.ylabel("Density")

plt.legend()
plt.grid(True)
```

```python
plt.savefig("./report/class_conditional_probability.svg")
plt.show()
```

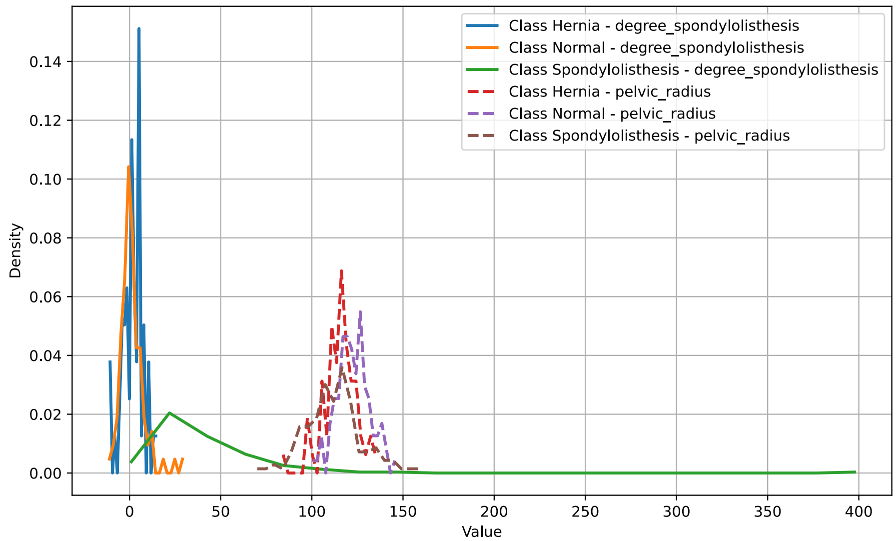

Figure 2: Class-conditional probability density functions of the highest and lowest discriminative power variables.

2. **Using a stratified 70-30 training-testing split with a fixed seed (`random_state=0`), assess in a single plot both the training and testing accuracies of a decision tree with depth limits in $\{1, 2, 3, 4, 5, 6, 8, 10\}$ and the remaining parameters as default.**

   ***[Optional]* Note that split thresholding of numeric variables in decision trees is non-deterministic in sklearn, hence you may opt to average the results using 10 runs per parameterization.**

```python
import pandas as pd, matplotlib.pyplot as plt, numpy as np
from scipy.io.arff import loadarff
from sklearn import metrics, tree
from sklearn.model_selection import train_test_split

# Read the ARFF file and prepare data
data = loadarff("./data/column_diagnosis.arff")
df = pd.DataFrame(data[0])
df["class"] = df["class"].str.decode("utf-8")
X, y = df.drop("class", axis=1), df["class"]

DEPTH_LIMIT = [1, 2, 3, 4, 5, 6, 8, 10]
training_accuracy, test_accuracy = [], []

# Split the dataset into a testing set (30%) and a training set (70%)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=0
)

```

```python
for depth_limit in DEPTH_LIMIT:
    # Create and fit the decision tree classifier
    predictor = tree.DecisionTreeClassifier(
        max_depth=depth_limit, random_state=0
    )
    predictor.fit(X_train, y_train)

    # Use the decision tree to predict the outcome of the given observations
    y_train_pred = predictor.predict(X_train)
    y_test_pred = predictor.predict(X_test)

    # Get the accuracy of each test
    train_acc = metrics.accuracy_score(y_train, y_train_pred)
    test_acc = metrics.accuracy_score(y_test, y_test_pred)

    training_accuracy.append(train_acc)
    test_accuracy.append(test_acc)

plt.plot(
    DEPTH_LIMIT,
    training_accuracy,
    label="Training Accuracy",
    marker="+",
    color="#f8766d",
)
plt.plot(
    DEPTH_LIMIT,
    test_accuracy,
    label="Test Accuracy",
    marker=".",
    color="#00bfc4",
)

plt.xlabel("Depth Limit")
plt.ylabel("Accuracy")

plt.legend()
plt.grid(True)
plt.savefig("./report/training_testing_accuracies.svg")
plt.show()
```

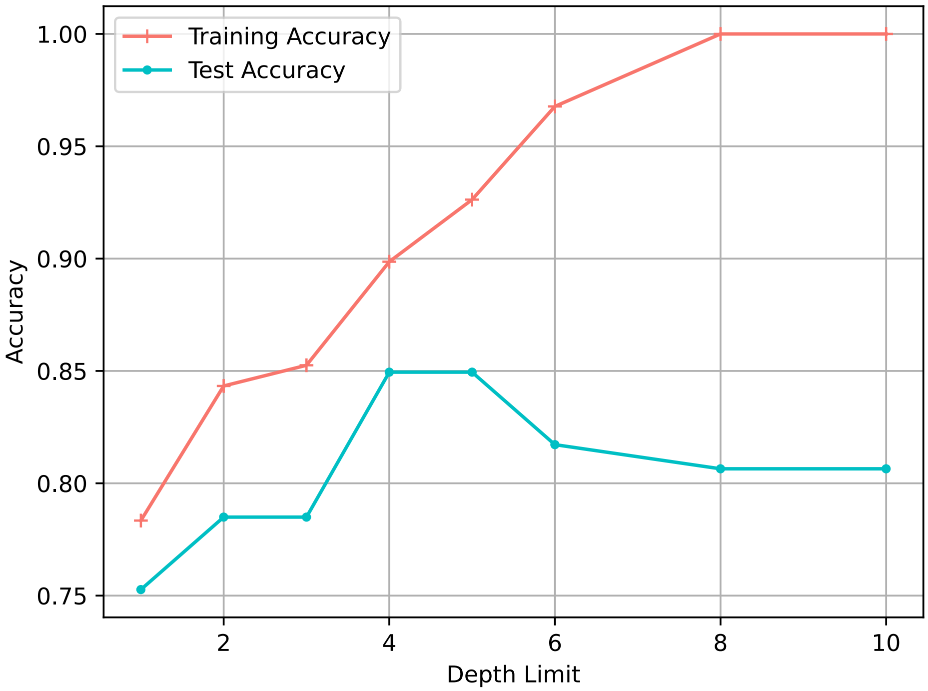

Figure 3: Accuracy of the trained decision tree, applied to both a test and training sets, for varying depth limits.

3. **Comment on the results, including the generalization capacity across settings.**

   Blah

4. **To deploy the predictor, a healthcare team opted to learn a single decision tree (`random_state=0`) using *all* available data as training data, and further ensuring that each leaf has a minimum of 20 individuals in order to avoid overfitting risks.**

   (a) **Plot the decision tree.**

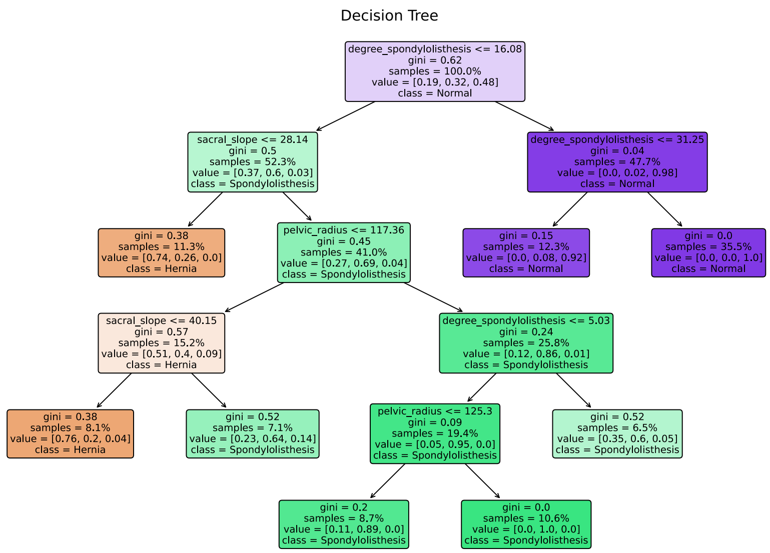

Figure 4: Accuracy of trained decision tree, applied to both a test and training sets, for varying depth limits.

   (b) **Characterize a hernia condition by identifying the hernia-conditional associations.**

   Blah

END