Aprendizagem 2023
Homework I — Group 003
(ist1107028, ist1107137)

Part I: Pen and paper

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

The entropy of y_{out} is given by:

$$\begin{aligned} H(y_{\text{out}}|y_1 >= 0.3) = &- P(y_{\text{out}} = A|y_1 >= 0.3)\log_2(P(y_{\text{out}} = A|y_1 >= 0.3)) \\ &- P(y_{\text{out}} = B|y_1 >= 0.3)\log_2(P(y_{\text{out}} = B|y_1 >= 0.3)) \\ &- P(y_{\text{out}} = C|y_1 >= 0.3)\log_2(P(y_{\text{out}} = C|y_1 >= 0.3)) \end{aligned}$$

Replacing by the values is obtained:

$$H(y_{\text{out}}|y_1 >= 0.3) = -\left(\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right)\right) \approx 1.5567$$

We must now calculate $H(y_{\text{out}}|y_1 >= 0.3, y_x)$, in which x will assume the values of 2, 3 and 4:

$$\begin{aligned} H(y_{\text{out}}|y_1 >= 0.3, y_x) = &P(y_x = 0|y_1 >= 0.3)H(y_{\text{out}}|y_1 >= 0.3, y_x = 0) \\ &+P(y_x = 1|y_1 >= 0.3)H(y_{\text{out}}|y_1 >= 0.3, y_x = 1) \\ &+P(y_x = 2|y_1 >= 0.3)H(y_{\text{out}}|y_1 >= 0.3, y_x = 2) \end{aligned} \tag{1}$$

The information gain of y_x is given by:

$$IG(y_{\text{out}}|y_1 >= 0.3, y_x) = H(y_{\text{out}}|y_1 >= 0.3) - H(y_{\text{out}}|y_1 >= 0.3, y_x) \tag{2}$$

x=2:

$$P(y_2 = 0|y_1 >= 0.3) = \frac{4}{7} \quad P(y_2 = 1|y_1 >= 0.3) = \frac{3}{7} \quad P(y_2 = 2|y_1 >= 0.3) = \frac{0}{7}$$

$$H(y_{\text{out}}|y_1 >= 0.3, y_2 = 0) = -\left(\frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 1.5$$

$$H(y_{\text{out}}|y_1 >= 0.3, y_2 = 1) = -\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{0}{3}\log_2\left(\frac{0}{3}\right)\right) \approx 0.9183$$

On account of the fact that $P(y_2 = 2|y_1 >= 0.3) = \frac{0}{7} = 0$, $H(y_{\text{out}}|y_1 >= 0.3, y_2 = 2)$ will also be 0.

Thus, by inserting these values into equation (1), we get:

$$H(y_{\text{out}}|y_1 >= 0.3, y_2) = \frac{4}{7} \times 1.5 + \frac{3}{7} \times 0.9183 + \frac{0}{7} \times 0 = 1.2507$$

Lastly, we calculate the information gain as per (2):

$$IG(y_{\text{out}}|y_1 >= 0.3, y_2) = 1.5567 - 1.2507 = 0.306$$

x=3:

$$P(y_3 = 0|y_1 >= 0.3) = \frac{2}{7} \quad P(y_3 = 1|y_1 >= 0.3) = \frac{4}{7} \quad P(y_3 = 2|y_1 >= 0.3) = \frac{1}{7}$$

$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 0) = -\left(\frac{2}{2}\log_2\left(\frac{2}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right) + \frac{0}{2}\log_2\left(\frac{0}{2}\right)\right) = 0$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1) = -\left(\frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 1.5$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 2) = -\left(\frac{0}{1}\log_2\left(\frac{0}{1}\right) + \frac{1}{1}\log_2\left(\frac{1}{1}\right) + \frac{0}{1}\log_2\left(\frac{0}{1}\right)\right) = 0$$

Thus, by inserting these values into equation (1), we get:

$$H(y_{\text{out}}|y_1 >= 0.3, y_3) = \frac{2}{7} \times 0 + \frac{4}{7} \times 1.5 + \frac{1}{7} \times 0 \approx 0.8571$$

Lastly, we calculate the information gain as per (2):

$$IG(y_{\text{out}}|y_1 >= 0.3, y_3) = 1.5567 - 0.8571 = \mathbf{0.6996}$$

x=4:

$$P(y_4 = 0|y_1 >= 0.3) = \frac{4}{7} \quad P(y_4 = 1|y_1 >= 0.3) = \frac{3}{7} \quad P(y_4 = 2|y_1 >= 0.3) = \frac{0}{7}$$

$$H(y_{\text{out}}|y_1 >= 0.3, y_4 = 0) = -\left(\frac{2}{4}\log_2\left(\frac{2}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right) + \frac{0}{4}\log_2\left(\frac{0}{4}\right)\right) = 1$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_4 = 1) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{0}{3}\log_2\left(\frac{0}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) \approx 0.9183$$

Once again, $P(y_4 = 2|y_1 >= 0.3) = \frac{0}{7} = 0$. Therefore $H(y_{\text{out}}|y_1 >= 0.3, y_4 = 2)$ will also equal 0.

Thus, by inserting these values into equation (1), we get:

$$H(y_{\text{out}}|y_1 >= 0.3, y_4) = \frac{4}{7} \times 1 + \frac{3}{7} \times 0.9183 + \frac{0}{7} \times 0 \approx 0.9650$$

Lastly, we calculate the information gain as per (2):

$$IG(y_{\text{out}}|y_1 >= 0.3, y_4) = 1.5567 - 0.9650 = 0.5917$$

After calculating the information gains for each attribute, we can observe that the attribute y_3 has the highest value of 0.6996. Accordingly, we chose it as the next node. Since there are at least four observations with $y_1 >= 0.3$, we split the new node.

If we fix $y_1 >= 0.3$ with $y_3 = 0$, $y_3 = 1$ and $y_3 = 2$, we will obtain 2, 4 and 1 observations, respectively. This gives us three new leaves for the branch in question. The node corresponding to $y_3 = 0$ will be class A, and the one corresponding to $y_3 = 2$ will be from class B, since these classes are the ones that appear most frequently for the respective conditions in the data set. For the $y_3 = 1$ branch with the $y_1 >= 0.3$ condition we get exactly four observations, as we said previously. That being the case, we must split the next node with the higher information gain.

$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_2 = 0) = -\left(\frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{2}{4}\log_2\left(\frac{2}{4}\right)\right) = 1.5$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_2 = 1) = -\left(\frac{0}{0}\log_2\left(\frac{0}{0}\right) + \frac{0}{0}\log_2\left(\frac{0}{0}\right) + \frac{0}{0}\log_2\left(\frac{0}{0}\right)\right) = 0$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_2 = 2) = -\left(\frac{0}{0}\log_2\left(\frac{0}{0}\right) + \frac{0}{0}\log_2\left(\frac{0}{0}\right) + \frac{0}{0}\log_2\left(\frac{0}{0}\right)\right) = 0$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_4 = 0) = -\left(\frac{0}{1}\log_2\left(\frac{0}{1}\right) + \frac{1}{1}\log_2\left(\frac{1}{1}\right) + \frac{0}{1}\log_2\left(\frac{0}{1}\right)\right) = 0$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_4 = 1) = -\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{0}{3}\log_2\left(\frac{0}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right) \approx 0.9183$$
$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_4 = 2) = -\left(\frac{0}{0}\log_2\left(\frac{0}{0}\right) + \frac{0}{0}\log_2\left(\frac{0}{0}\right) + \frac{0}{0}\log_2\left(\frac{0}{0}\right)\right) = 0$$

From other calculations above, we know that:

$$H(y_{\text{out}}|y_1 >= 0.3, y_3 = 1) = 1.5$$

Therefore, replacing the above values on equation (1), gives us:

$$E(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_2) = \frac{4}{4} \times 1.5 + \frac{0}{4} \times 0 + \frac{0}{4} \times 0 = 1.5$$
$$E(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_4) = \frac{1}{4} \times 0 + \frac{3}{4} \times 0.9183 + \frac{0}{4} \times 0 = 0.6887$$

Finally, we can calculate the information gain, as per (2):

$$IG(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_2) = 1.5 - 1.5 = 0$$
$$IG(y_{\text{out}}|y_1 >= 0.3, y_3 = 1, y_4) = 1.5 - 0.6887 = \mathbf{0.8113}$$

Since y_4 is the node with the highest information gain of 0.8113, it is the node we chose to split. With the conditions of $y_1 >= 0.3$ and $y_3 = 1$, we can check that the conditions $y_4 = 0$, $y_4 = 1$ and $y_4 = 2$ all have less than 4 observations each. Therefore, we take the class with the highest frequency for each respective conditions on the dataset. Finally, we can construct the following decision tree:

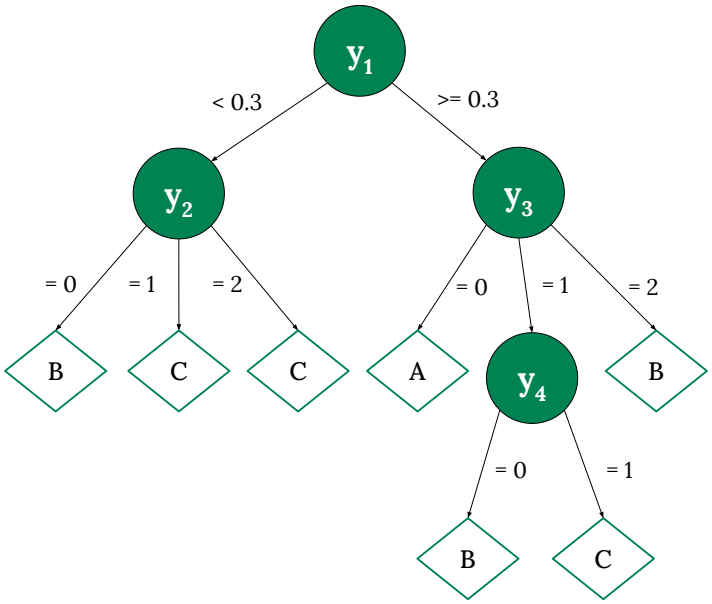


Figure 1: Decision Tree for exercise 1

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

To construct the confusion matrix, we must first predict the values for each observation. We start at the initial node of the tree and follow its branches. When we reach the next node, the process is repeated for the succeeding variable and the following ones until a leaf is reached. The class present in this leaf will correspond to the predicted value for the respective variable in the dataset (column D in the given table). The real value will be the one in the y_{out} column corresponding to that same variable. This results in the following values:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
real = [	C	B	C	B	C	B	A	A	C	C	A	B]
predicted = [	C	B	C	B	C	B	C	A	C	C	A	B]

Finally, we set up the confusion matrix by counting the pairs of observations previously mentioned.

		Real			
		A	B	C	
Predicted	A	2	0	0	2
	B	0	4	0	4
	C	1	0	5	6
		3	4	5	12

Figure 2: Confusion matrix for exercise 2

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

F1_{score} is given by the following equation:

$$\text{F1}_{\text{score}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

And precision and recall are given by:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5}$$

We can calculate the precision by replacing the according values for each class in equation (4):

$$\text{Precision}_A = \frac{2}{2+0} = 1 \qquad \text{Precision}_B = \frac{4}{4+0} = 1 \qquad \text{Precision}_C = \frac{5}{5+1} = \frac{5}{6}$$

Same goes for recall, using equation (5):

$$\text{Recall}_A = \frac{2}{2+1} = \frac{2}{3} \qquad \text{Recall}_B = \frac{4}{4+0} = 1 \qquad \text{Recall}_C = \frac{5}{5+0} = 1$$

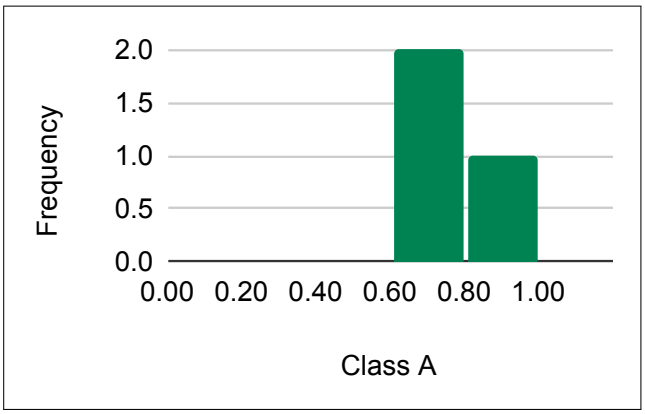
At Last, we determine each F1_{score} with (3):

$$\text{F1}_{\text{score}_A} = 2 \times \frac{1 \times \frac{2}{3}}{1 + \frac{2}{3}} = 0.8000 \qquad \text{F1}_{\text{score}_B} = 2 \times \frac{1 \times 1}{1 + 1} = 1 \qquad \text{F1}_{\text{score}_C} = 2 \times \frac{\frac{5}{6} \times 1}{\frac{5}{6} + 1} \approx 0.9091$$

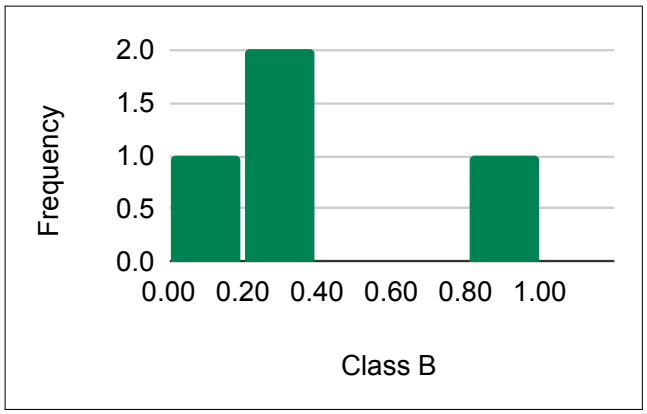
The **class with the lowest training F1 score is A**, with a score of 0.8000.

4. **Draw the class-conditional relative histograms of y1 using 5 equally spaced bins in [0,1]. Find the n-ary root split using the discriminant rules from these empirical distributions.**

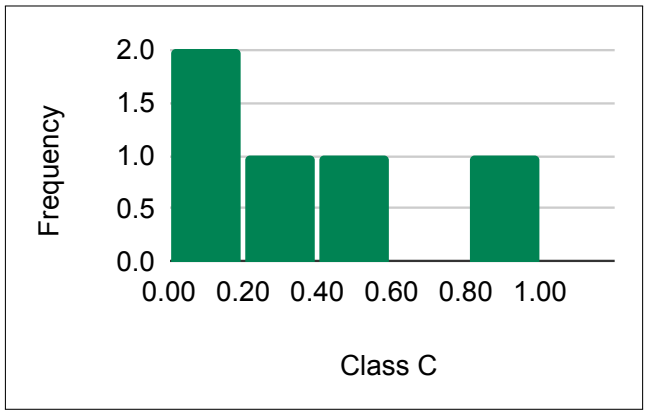
   1. Firstly we must create a histogram for every class present (A, B and C). For each, when a value of y_1 falls into a bin, we add one to the value of that bin. Each bin has a size of 0.2, since we are using 5 equal spaced bins in the interval [0,1].



(a) Histogram for Class A



(b) Histogram for Class B



(c) Histogram for Class C

Figure 3: Class-conditional relative histograms of y_1 using 5 equal spaced bins in [0,1]

2. To find the root split using the discriminating rules for the empirical distributions, we draw a decision tree with just one root split on y_1. Each branch contains the values of one bin, and it's corresponding leaf represents the class with highest value for that bin. Since we are faced with a tie in branch $0.8 <= y_1 <= 1$, we will use the alphabetic ascending rule to make this decision (like in the first exercise).

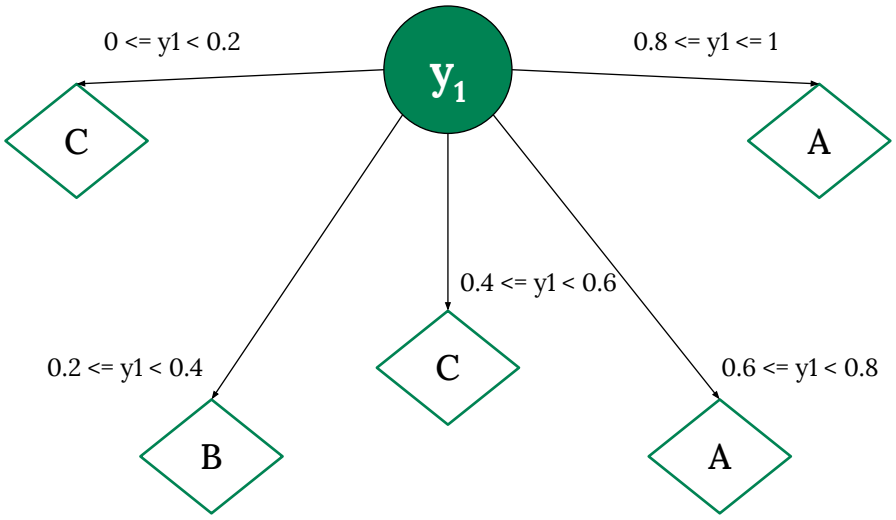


Figure 4: Decision Tree using the discriminant rules from the empirical distributions

Part II: Programming

Consider the diabetes.arff data available at the homework tab, comprising 8 biological features to classify 768 patients into 2 classes (normal, diabetes).

1. **ANOVA is a statistical test that can be used to assess the discriminative power of a single input variable. Using f_classif from sklearn, identify the input variables with the worst and best discriminative power. Plot their class-conditional probability density functions.**

```
import pandas as pd, seaborn as sns, matplotlib.pyplot as plt
from scipy.io.arff import loadarff
from sklearn.feature_selection import f_classif

# Load the dataset
data = loadarff('./diabetes.arff')
df = pd.DataFrame(data[0])
```

```
# Separate features and target variable
df["Outcome"] = df["Outcome"].str.decode("utf-8")
X, y = df.drop("Outcome", axis=1), df["Outcome"]
df["Outcome"] = df["Outcome"].map({'1': 'Diabetic', '0': 'Normal'})

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

# Find best and worst discriminative features
best_feature = X.columns[f_values.argmax()]
worst_feature = X.columns[f_values.argmin()]

plt.figure(figsize=(12, 6))
custom_palette = ["#1f77b4", "#E40071"]

# Plot for the best discriminative features
plt.subplot(1, 2, 1)
sns.kdeplot(data=df, x=best_feature, hue='Outcome',
            fill=False, common_norm=False, palette=custom_palette)
plt.title(f'Best Discriminative Feature: {best_feature}')
plt.grid(True)

# Plot for the worst discriminative features
plt.subplot(1, 2, 2)
sns.kdeplot(data=df, x=worst_feature, hue='Outcome',
            fill=False, common_norm=False, palette=custom_palette)
plt.title(f'Worst Discriminative Feature: {worst_feature}')
plt.grid(True)

plt.tight_layout()
plt.show()
```

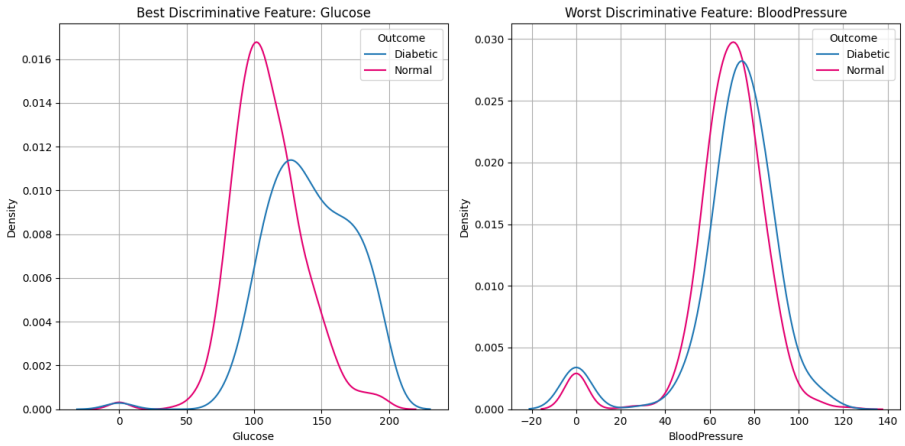


Figure 5: Class-conditional probability density functions plot

As we can see from the graphic above:

- The variable with the best discriminative power is **Glucose**: It's visible that the distribution for 'Diabetic' (blue line) has a significant shift compared to 'Normal' (pink line), showing that glucose levels are a strong differentiating factor between the two classes.
- The variable with the worst discriminative power is **BloodPressure**: The distributions for both 'Diabetic' and 'Normal' classes overlap significantly, meaning this feature doesn't separate the two classes well.

2. **Using a stratified 80–20 training-testing split with a fixed seed (random_state=1), assess in a single plot both the training and testing accuracies of a decision tree with minimum sample split in and the remaining parameters as default.**

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

# Load the dataset
data = loadarff('./diabetes.arff')
df = pd.DataFrame(data[0])

# Separate features and target variable
df["Outcome"] = df["Outcome"].str.decode("utf-8")
X, y = df.drop("Outcome", axis=1), df["Outcome"]
df["Outcome"] = df["Outcome"].map({'1': 'Diabetic', '0': 'Normal'})

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    stratify=y, random_state=1)
min_samples_splits = [2, 5, 10, 20, 30, 50, 100]
train_acc, test_acc = [], []

for min_samples in min_samples_splits:
    predictor = tree.DecisionTreeClassifier(min_samples_split=min_samples,
    random_state=1)
    predictor.fit(X_train, y_train)

    # Calculate training and testing accuracy
    train_acc.append(metrics.accuracy_score(y_train, predictor.predict(X_train
    )))
    test_acc.append(metrics.accuracy_score(y_test, predictor.predict(X_test)))

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(
    min_samples_splits,
    train_acc,
    label='Training Accuracy',
    marker='o',
)
plt.plot(
```

```
    min_samples_splits,
    test_acc,
    label='Testing Accuracy',
    marker='D',
    color='#E40071'
)
plt.xlabel('Minimum Sample Split')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Testing accuracy of a decision tree')
plt.grid(True)
plt.show()
```

Figure 6: Training and testing accuracies of a decision tree with various depth limits

3. **Critically analyze these results, including the generalization capacity across settings**

As shown in the graph above, increasing the `min_samples_split` parameter leads to a roughly 20% decrease in accuracy. This is expected because higher values of `min_samples_split` result in simpler trees, which reduces the model's ability to fit the training data accurately.

Regarding testing accuracy, we observe a slight initial decrease, indicating overfitting (high training accuracy but low testing accuracy). As `min_samples_split` increases, testing accuracy improves, reaching 75%-80%, which indicates better generalization. Beyond this point, testing accuracy slightly declines, suggesting the model is becoming underfit due to the tree's reduced complexity.

In terms of generalization, we can observe:

- *Underfitting* for higher `min_samples_split`, since both training and testing accuracy drop. This indicates the model is becoming too simplistic and fails to capture the underlying patterns in the data, leading to underfitting.

- Optimal generalization around 20–30 `min_samples_split`, where testing accuracy peaks, balancing bias and variance. The model complexity is reduced enough to avoid overfitting while retaining predictive power.
- Overfitting at Low `min_samples_split`, since The model fits the training data perfectly but performs poorly on test data, indicating poor generalization to new, unseen data.

The ideal minimal sample split appears to be around 30, which is the point in the graphic where the testing accuracy reaches its maximum, striking a balance between model complexity and generalization to new data. This would help us prevent the model from being too minimalistic (*underfitting*) or overly complex (*overfitting*).

4. **To deploy the predictor, a healthcare provider opted to learn a single decision tree (random_state=1) using all available data and ensuring that the maximum depth would be 3 in order to avoid overfitting risks.**

(i) Plot the decision tree.

```
import matplotlib.pyplot as plt, pandas as pd
from scipy.io.arff import loadarff
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Load the dataset
data = loadarff('./diabetes.arff')
df = pd.DataFrame(data[0])

# Separate features and target variable
df["Outcome"] = df["Outcome"].str.decode("utf-8")
X, y = df.drop("Outcome", axis=1), df["Outcome"]

# Create and train the decision tree classifier
tree = DecisionTreeClassifier(max_depth=3, random_state=1)
tree.fit(X, y)

# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(tree, feature_names=X.columns, class_names=['normal', 'diabetic'],
          filled=True, rounded=True)
plt.show()
```

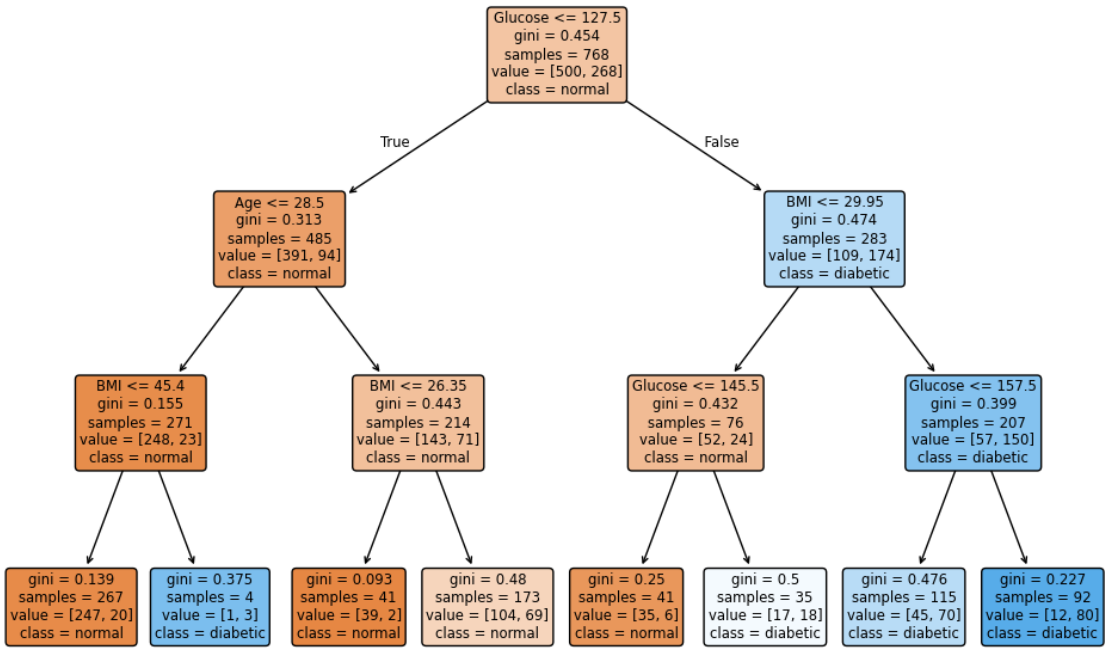


Figure 7: Decision tree for exercise 4

(ii) Explain what characterizes diabetes by identifying the conditional associations together with their posterior probabilities.

The diabetic condition is characterized by:

- Glucose <= 127.5, Age <= 28.5 and BMI >= 45.4
- Glucose <= 127.5 and BMI >= 29.95

The posterior probabilities are calculated using the following expression:

$$PP = \frac{\text{node\_samples\_current\_class}}{\text{total\_node\_samples}}$$

As we can see, the most critical features in determining diabetes are Glucose and BMI.
Glucose levels exceeding 127.5 significantly increase the likelihood of developing diabetes. For individuals with glucose levels surpassing 157.5, the posterior probability is 87%, indicating a high likelihood of suffering from a diabetic condition. Even with a lower BMI, high glucose levels still pose a considerable risk for diabetes. Specifically, a BMI above a certain threshold is strongly associated with an increased chance of being diabetic.