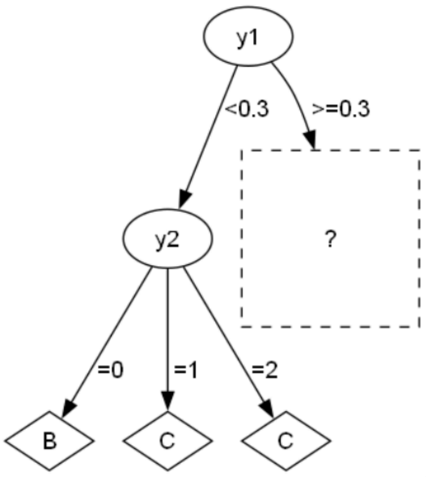
Aprendizagem 2024

HomeWork 1 - Grupo 02

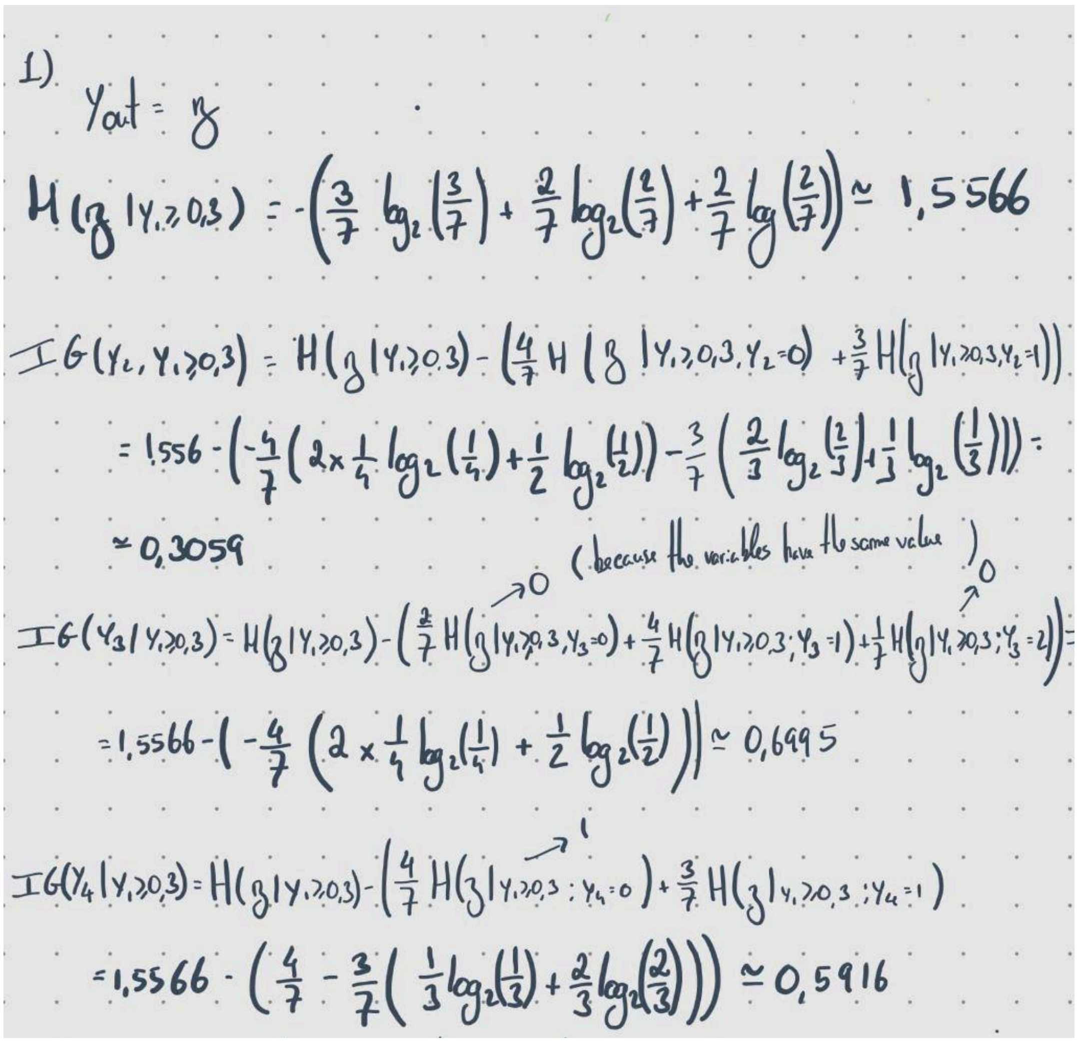
Joana Vaz (106078) e David Quintino (107095)

I. Pen-and-paper

D	y_1	y_2	y_3	y_4	y_{out}
x_1	0.22	2	0	1	C
x_2	0.06	0	0	0	B
x_3	0.16	1	2	2	C
x_4	0.21	0	0	0	B
x_5	0.01	2	2	0	C
x_6	0.3	0	1	0	B
x_7	0.76	0	1	1	A
x_8	0.86	1	0	0	A
x_9	0.93	0	1	1	C
x_{10}	0.47	0	1	1	C
x_{11}	0.73	1	0	0	A
x_{12}	0.89	1	2	0	B



1) [5v] 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.



1) $y_{out} = z$

$$H(z \mid y_1 \geq 0{,}3) = -\left(\frac{3}{7}\log_2\left(\frac{3}{7}\right) + \frac{2}{7}\log_2\left(\frac{2}{7}\right) + \frac{2}{7}\log\left(\frac{2}{7}\right)\right) \simeq 1{,}5566$$

$$IG(y_2, y_1 \geq 0{,}3) = H(z \mid y_1 \geq 0{,}3) - \left(\frac{4}{7}H(z \mid y_1 \geq 0{,}3, y_2 = 0) + \frac{3}{7}H(z \mid y_1 \geq 0{,}3, y_2 = 1)\right)$$

$$= 1{,}556 - \left(-\frac{4}{7}\left(2 \times \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right) - \frac{3}{7}\left(\frac{2}{3}\log_2\left(\frac{2}{3}\right) + \frac{1}{3}\log_2\left(\frac{1}{3}\right)\right)\right) =$$

$$\simeq 0{,}3059$$

(because the variables have the same value)

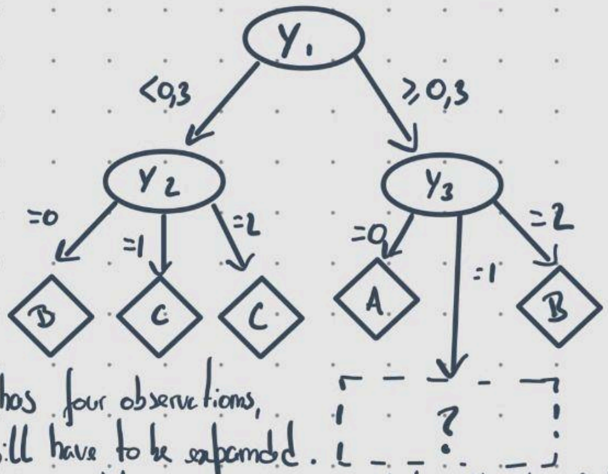
$$IG(y_3 \mid y_1 \geq 0{,}3) = H(z \mid y_1 \geq 0{,}3) - \left(\frac{2}{7}\overset{\nearrow 0}{H(z \mid y_1 \geq 0{,}3, y_3 = 0)} + \frac{4}{7}H(z \mid y_1 \geq 0{,}3; y_3 = 1) + \frac{1}{7}\overset{\nearrow 0}{H(z \mid y_1 \geq 0{,}3; y_3 = 2)}\right) =$$

$$= 1{,}5566 - \left(-\frac{4}{7}\left(2 \times \frac{1}{4}\log_2\left(\frac{1}{4}\right) + \frac{1}{2}\log_2\left(\frac{1}{2}\right)\right)\right) \simeq 0{,}6995$$

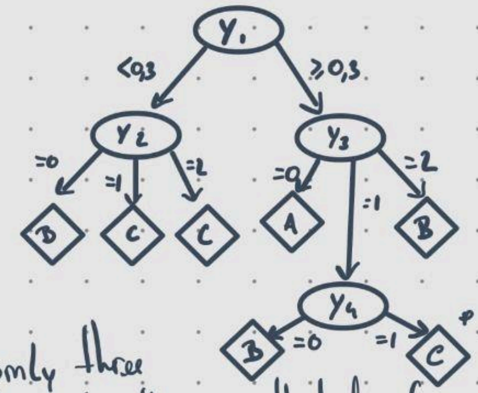
$$IG(y_4 \mid y_1 \geq 0{,}3) = H(z \mid y_1 \geq 0{,}3) - \left(\frac{4}{7}\overset{\nearrow 1}{H(z \mid y_1 \geq 0{,}3; y_4 = 0)} + \frac{3}{7}H(z \mid y_1 \geq 0{,}3; y_4 = 1)\right)$$

$$= 1{,}5566 - \left(\frac{4}{7} - \frac{3}{7}\left(\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right)\right) \simeq 0{,}5916$$

Since y_3 has the greatest information gain, the tree will look like this

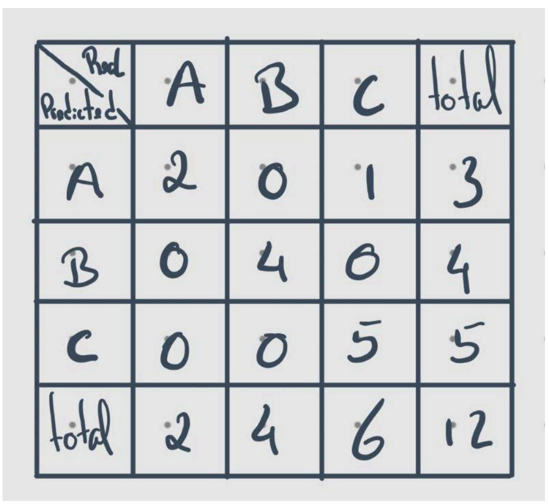


This node has four observations,
therefore it will have to be expanded.
The remaining variables are y_2 and y_4, but in this branches' conditions ($y_1 \geq 0{,}3$ and $y_3 = 1$), y_2 only has zeros and the information gain would be 0. So the tree will look like this:



* Since there are only three observations (A; C; C), we chose the one with higher frequency.

2) [2.5v] Draw the training confusion matrix for the learnt decision tree.



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

3) [1.5v] Identify which class has the lowest training F1 score.

3)

A: Precisiom $= 2/3$, Recal $= 1$ $\quad F_1 = 2 \times \dfrac{2/3 \times 1}{2/3 + 1} = \dfrac{4/3}{5/3} = 0{,}8$

B: Precisiom $= 1$, Recall $= 1$ $\quad F_1 = 2 \times \dfrac{1 \times 1}{1+1} = 1$

C: Precisiom $= 1$, Recall $= 5/6$ $\quad F_1 = 2 \times \dfrac{1 \times 5/6}{1 + 5/6} = \dfrac{\frac{10}{6}}{11/6} = \dfrac{10}{11}$

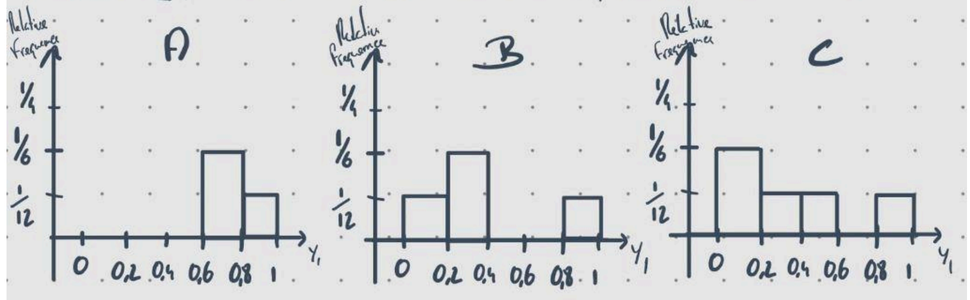
R: A has the lowest training F_1 score

4) [2v] 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.

4)

5 equally spaced bims in $[0,1]$:

			freq				
1	$[0;0{,}2[$	$x_2; x_3, x_5$	3	$P(Y_1 \in 1) = 1/4$	$A: 0$	$B = \frac{1}{12}$	$C = \frac{1}{6}$
2	$[0{,}2;0{,}4[$	x_1, x_4, x_6	3	$P(Y_1 \in 2) = 1/4$	$A - 0$	$B = \frac{1}{6}$	$C = 1/12$
3	$[0{,}4;0{,}6[$	x_{10}	1	$P(Y_1 \in 3) = 1/12$	$A = 0$	$B = 0$	$C = 1/12$
4	$[0{,}6;0{,}8[$	$x_7\ x_{11}$	2	$P(Y_1 \in 4) = 1/6$	$A - 1/6$	$B = 0$	$C = 0$
5	$[0{,}8;1]$	$x_8, x_9\ x_{12}$	3	$P(Y_1 \in 5) = 1/4$	$A = 1/12$	$B : 1/12$	$C - 1/12$



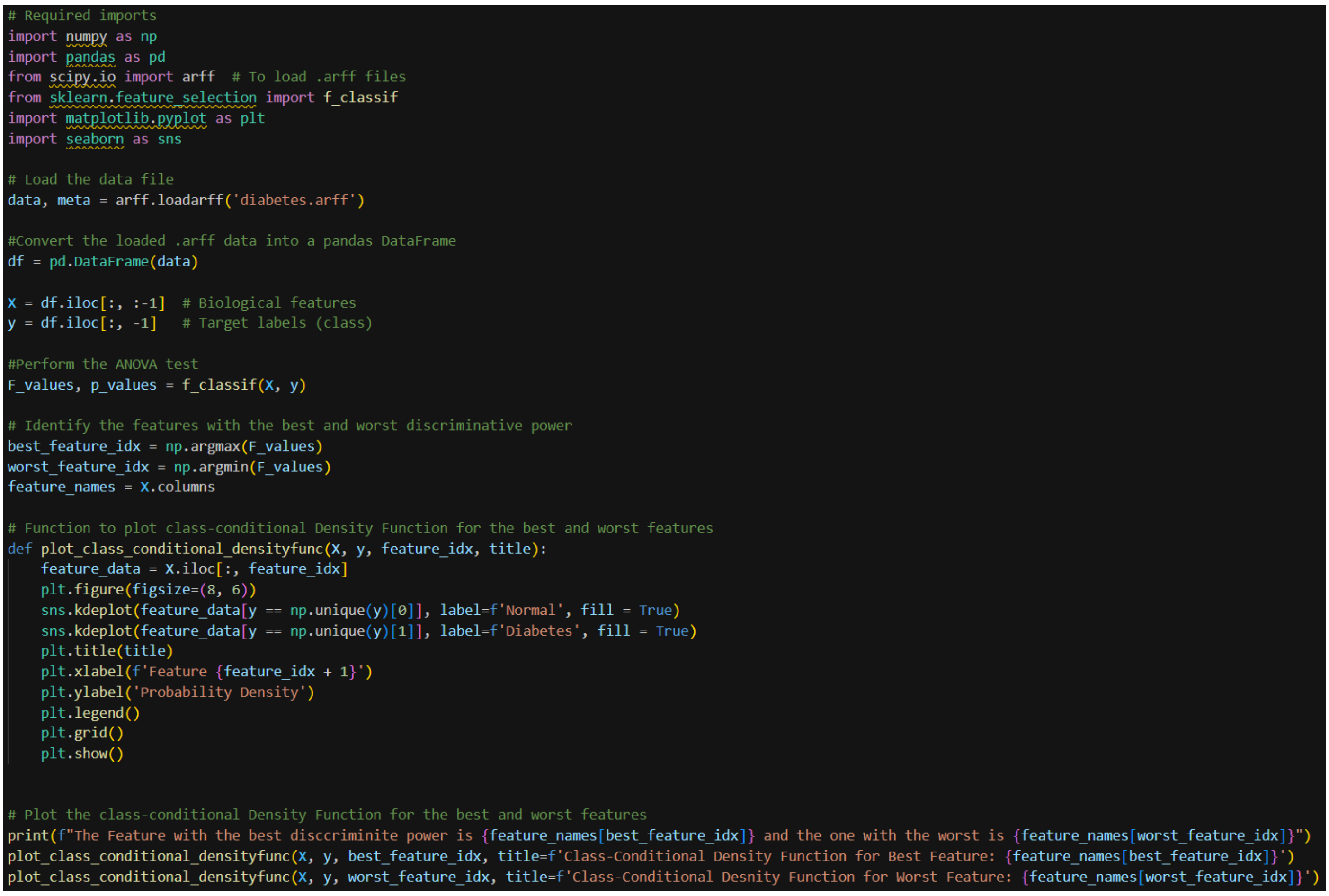
According to the histogram

- $Y_1 \leq 0{,}2 \rightarrow C$
- $0{,}2 < Y_1 \leq 0{,}4 \rightarrow B$
- $0{,}4 < Y_1 < 0{,}6 \rightarrow C$
- $0{,}6 < Y_1 \leq 0{,}8 \rightarrow A$
- $Y_1 > 0{,}8 \rightarrow A$ *

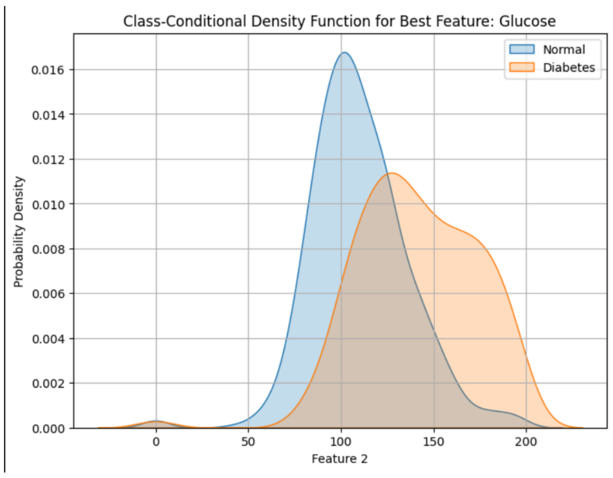
* Todos tem a mesma probabilidade então desempatamos por ordem alfabética

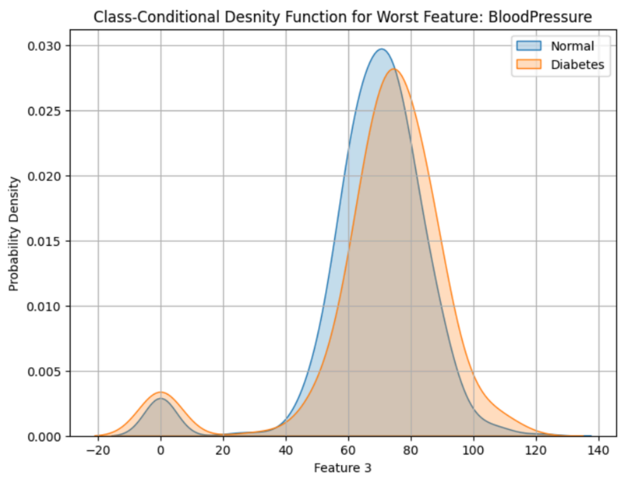
II. Programming [9v]

1) [1v] 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.



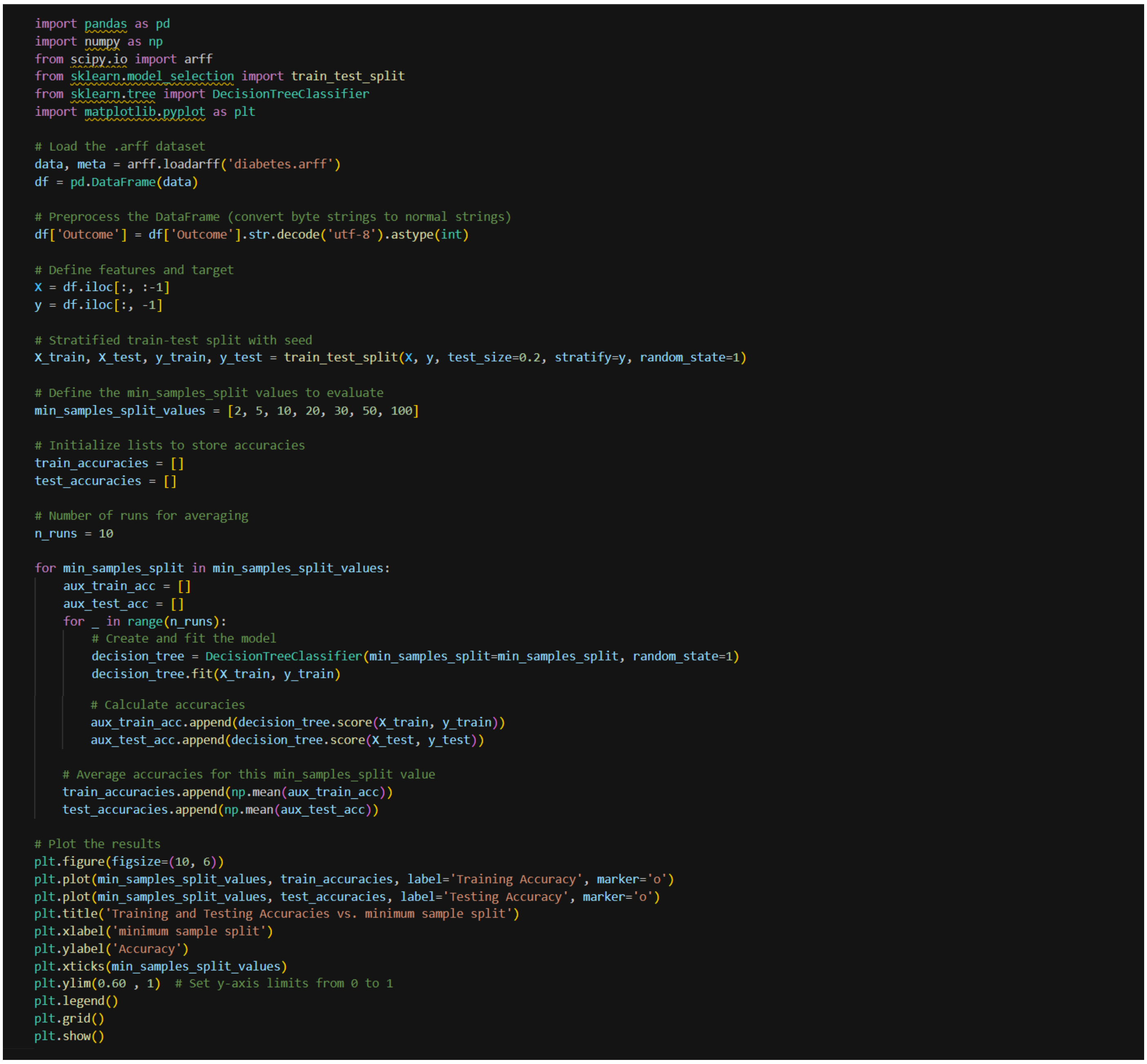
```python
# Required imports
import numpy as np
import pandas as pd
from scipy.io import arff  # To load .arff files
from sklearn.feature_selection import f_classif
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data file
data, meta = arff.loadarff('diabetes.arff')

#Convert the loaded .arff data into a pandas DataFrame
df = pd.DataFrame(data)

X = df.iloc[:, :-1]  # Biological features
y = df.iloc[:, -1]   # Target labels (class)

#Perform the ANOVA test
F_values, p_values = f_classif(X, y)

# Identify the features with the best and worst discriminative power
best_feature_idx = np.argmax(F_values)
worst_feature_idx = np.argmin(F_values)
feature_names = X.columns

# Function to plot class-conditional Density Function for the best and worst features
def plot_class_conditional_densityfunc(X, y, feature_idx, title):
    feature_data = X.iloc[:, feature_idx]
    plt.figure(figsize=(8, 6))
    sns.kdeplot(feature_data[y == np.unique(y)[0]], label=f'Normal', fill = True)
    sns.kdeplot(feature_data[y == np.unique(y)[1]], label=f'Diabetes', fill = True)
    plt.title(title)
    plt.xlabel(f'Feature {feature_idx + 1}')
    plt.ylabel('Probability Density')
    plt.legend()
    plt.grid()
    plt.show()


# Plot the class-conditional Density Function for the best and worst features
print(f"The Feature with the best disccriminite power is {feature_names[best_feature_idx]} and the one with the worst is {feature_names[worst_feature_idx]}")
plot_class_conditional_densityfunc(X, y, best_feature_idx, title=f'Class-Conditional Density Function for Best Feature: {feature_names[best_feature_idx]}')
plot_class_conditional_densityfunc(X, y, worst_feature_idx, title=f'Class-Conditional Desnity Function for Worst Feature: {feature_names[worst_feature_idx]}')
```



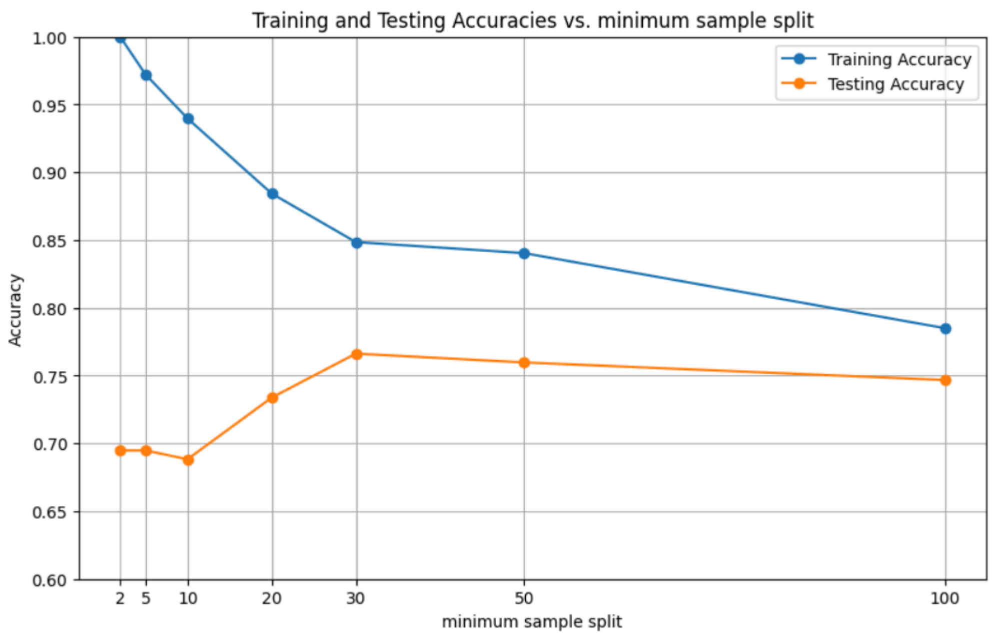


2) [4v] Using a stratified 80-20 training-testing split with a fixed seed (`random_state=1`), assess in a single plot both the training and testing accuracies of a decision tree with minimum sample split in $\{2, 5, 10, 20, 30, 50, 100\}$ and the remaining parameters as default.

[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
import numpy as np
from scipy.io import arff
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt

# Load the .arff dataset
data, meta = arff.loadarff('diabetes.arff')
df = pd.DataFrame(data)

# Preprocess the DataFrame (convert byte strings to normal strings)
df['Outcome'] = df['Outcome'].str.decode('utf-8').astype(int)

# Define features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Stratified train-test split with seed
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=1)

# Define the min_samples_split values to evaluate
min_samples_split_values = [2, 5, 10, 20, 30, 50, 100]

# Initialize lists to store accuracies
train_accuracies = []
test_accuracies = []

# Number of runs for averaging
n_runs = 10

for min_samples_split in min_samples_split_values:
    aux_train_acc = []
    aux_test_acc = []
    for _ in range(n_runs):
        # Create and fit the model
        decision_tree = DecisionTreeClassifier(min_samples_split=min_samples_split, random_state=1)
        decision_tree.fit(X_train, y_train)

        # Calculate accuracies
        aux_train_acc.append(decision_tree.score(X_train, y_train))
        aux_test_acc.append(decision_tree.score(X_test, y_test))

    # Average accuracies for this min_samples_split value
    train_accuracies.append(np.mean(aux_train_acc))
    test_accuracies.append(np.mean(aux_test_acc))

# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(min_samples_split_values, train_accuracies, label='Training Accuracy', marker='o')
plt.plot(min_samples_split_values, test_accuracies, label='Testing Accuracy', marker='o')
plt.title('Training and Testing Accuracies vs. minimum sample split')
plt.xlabel('minimum sample split')
plt.ylabel('Accuracy')
plt.xticks(min_samples_split_values)
plt.ylim(0.60 , 1)  # Set y-axis limits from 0 to 1
plt.legend()
plt.grid()
plt.show()
```



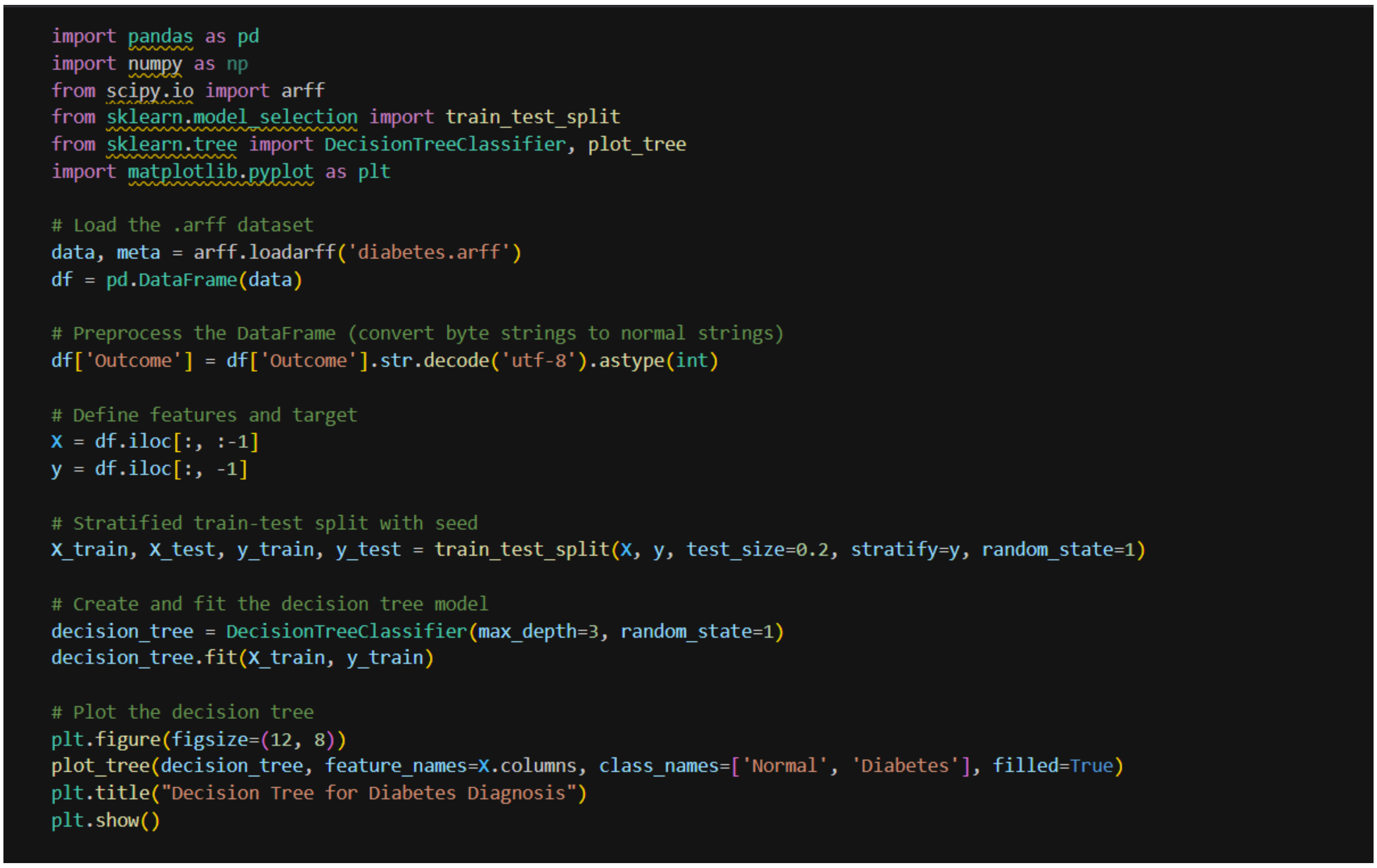
3) [2v] Critically analyze these results, including the generalization capacity across settings.

Overall, the model exhibits poor performance on the testing data while performing well on the training data. At the beginning, the training and testing accuracy curves are noticeably far apart, indicating that the model is not generalizing well. As the minimum sample split increases, particularly around a value of 20, we observe that the testing accuracy begins to improve, and the gap between the training and testing accuracy narrows significantly. This suggests that the model is starting to generalize better as we increase the minimum sample split, moving away from overfitting. However, if we continue to increase the minimum sample split beyond a certain point, we might start to see a decline in both training and testing accuracy, indicating potential underfitting.

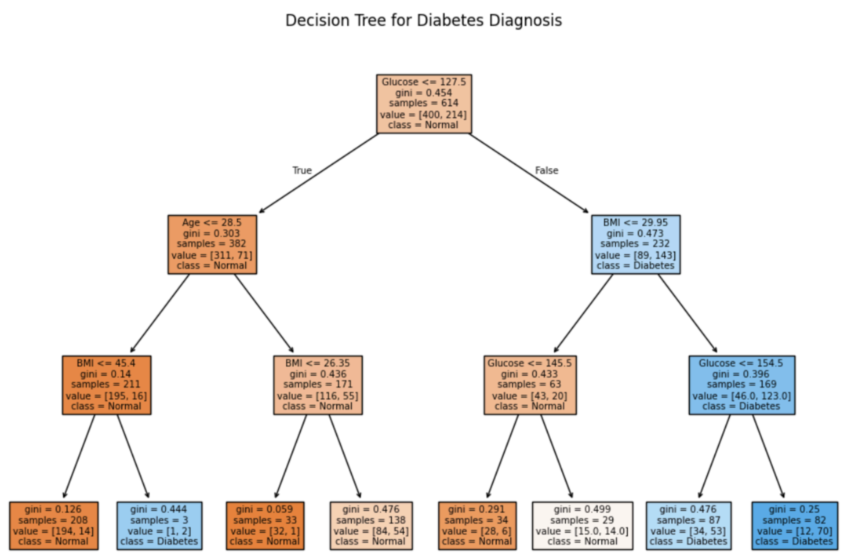
4) [2v] To deploy the predictor, a healthcare provider opted to learn a single decision tree (`random_state=1`) using *all* available data and ensuring that the maximum depth would be 3 in order to avoid overfitting risks.

   i. Plot the decision tree.

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



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

# Load the .arff dataset
data, meta = arff.loadarff('diabetes.arff')
df = pd.DataFrame(data)

# Preprocess the DataFrame (convert byte strings to normal strings)
df['Outcome'] = df['Outcome'].str.decode('utf-8').astype(int)

# Define features and target
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Stratified train-test split with seed
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=1)

# Create and fit the decision tree model
decision_tree = DecisionTreeClassifier(max_depth=3, random_state=1)
decision_tree.fit(X_train, y_train)

# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(decision_tree, feature_names=X.columns, class_names=['Normal', 'Diabetes'], filled=True)
plt.title("Decision Tree for Diabetes Diagnosis")
plt.show()
```


Decision Tree for Diabetes Diagnosis
Glucose <= 127.5 gini = 0.454 samples = 614 value = [400, 214] class = Normal
True
False
Age <= 28.5 gini = 0.303 samples = 382 value = [311, 71] class = Normal
BMI <= 29.95 gini = 0.473 samples = 232 value = [89, 143] class = Diabetes
BMI <= 45.4 gini = 0.14 samples = 211 value = [195, 16] class = Normal
BMI <= 26.35 gini = 0.436 samples = 171 value = [116, 55] class = Normal
Glucose <= 145.5 gini = 0.433 samples = 63 value = [43, 20] class = Normal
Glucose <= 154.5 gini = 0.396 samples = 169 value = [46.0, 123.0] class = Diabetes
gini = 0.126 samples = 208 value = [194, 14] class = Normal
gini = 0.444 samples = 3 value = [1, 2] class = Diabetes
gini = 0.059 samples = 33 value = [32, 1] class = Normal
gini = 0.476 samples = 138 value = [84, 54] class = Normal
gini = 0.291 samples = 34 value = [28, 6] class = Normal
gini = 0.499 samples = 29 value = [15.0, 14.0] class = Normal
gini = 0.476 samples = 87 value = [34, 53] class = Diabetes
gini = 0.25 samples = 82 value = [12, 70] class = Diabetes

ii) The decision tree shows that having a Glucose<= 127,5 and BMI <= 29,95 or Glucose<= 127,5, Age<=28,5 and BMI<= 45,4, probability means you have Diabetes.