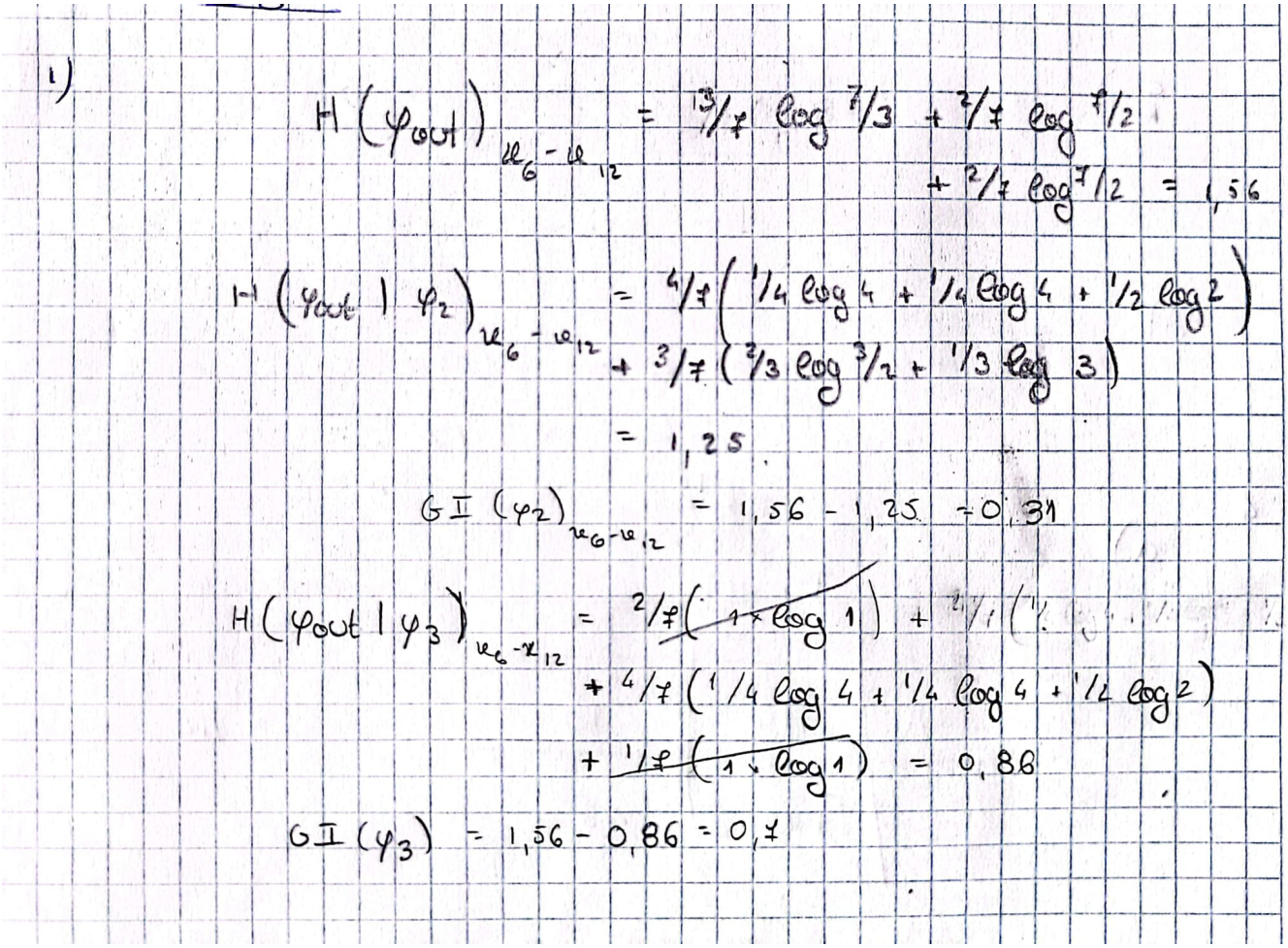
Aprendizagem 2024/25

Homework I – Group 066

(ist1107301, ist1106794)

I. Pen-and-paper

1)



TÉCNICO LISBOA

Aprendizagem 2024/25

Homework I – Group 066

(ist1107301, ist1106794)

$H(y_{out} \mid y_4)_{x_6, x_{12}} = 4/8 \left(1/2 \log 2 + 1/2 \log 2 \right)$

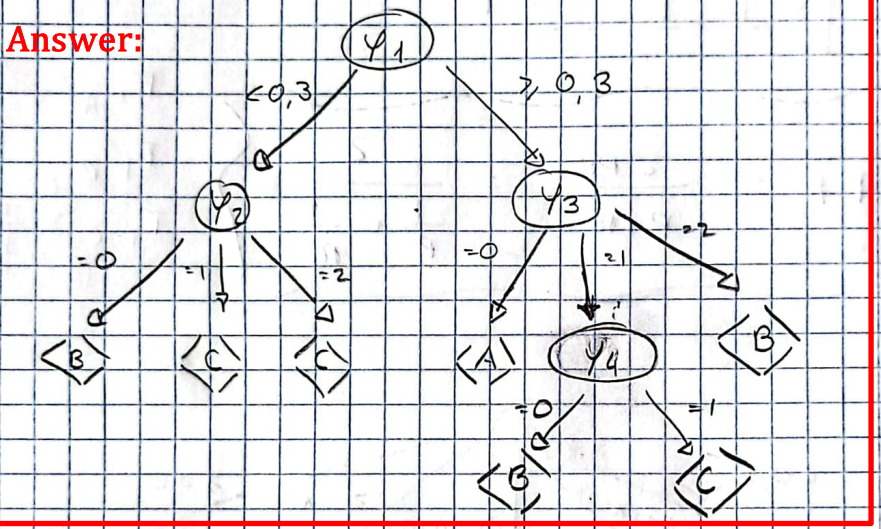
$+ 3/8 \left(1/3 \log 3 + 2/3 \log 3/2 \right)$

$= 0,96$

$GI(y_3) > GI(y_4) > GI(y_2)$

$GI(y_4) = 1,56 - 0,96 = 0,6$

Answer:



$H(y_{out})_{x_6, x_7, x_9, x_{10}} = 1/4 \log 4 + 1/4 \log 4 + 1/2 \log 2 = 1,5$

$H(y_{out} \mid y_2) = 1 \left(1/4 \log 4 + 1/4 \log 4 + 1/2 \log 2 \right) = 1,5$

$GI(y_2) = 1,5 - 1,5 = 0$

$H(y_{out} \mid y_4) = 1/4 \left(1 \log 1 \right) + \frac{3}{4} \left(1/3 \log 3 + 2/3 \log 3/2 \right)$

$= 0,69$

$GI(y_4) = 1,5 - 0,69 = 0,81$

$GI(y_4) > GI(y_2)$

Aprendizagem 2024/25

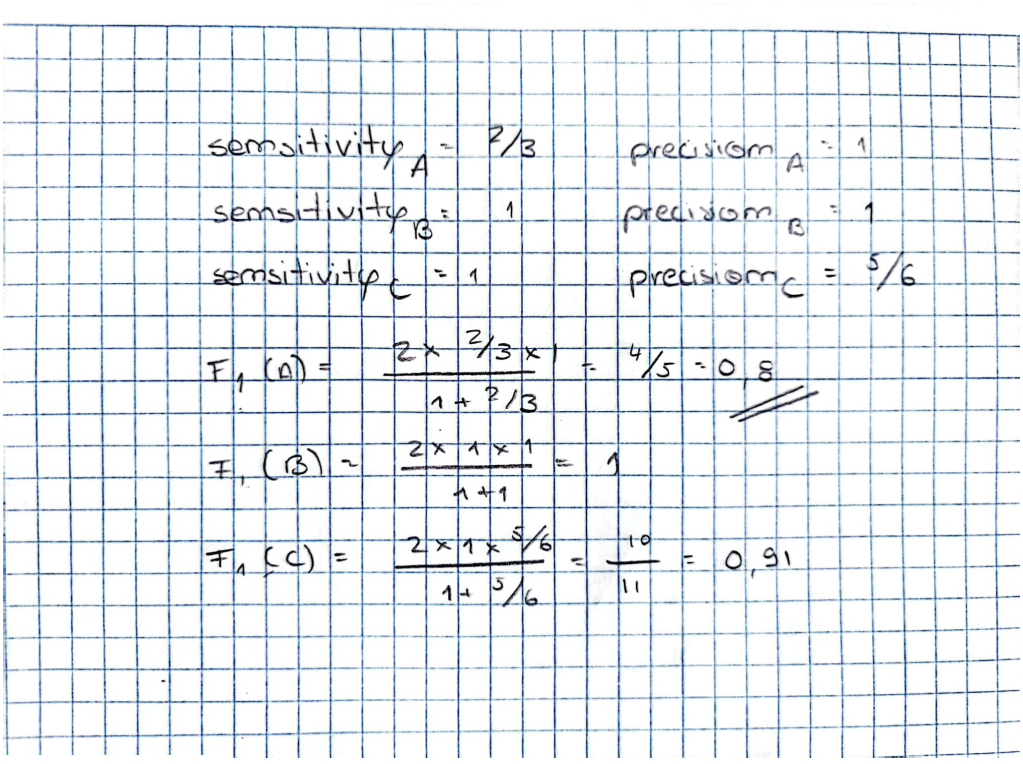
Homework I – Group 066

(ist1107301, ist1106794)

2)

		Truth		
Predictions		A	B	C
	A	2	0	0
	B	0	4	0
	C	1	0	5

3)

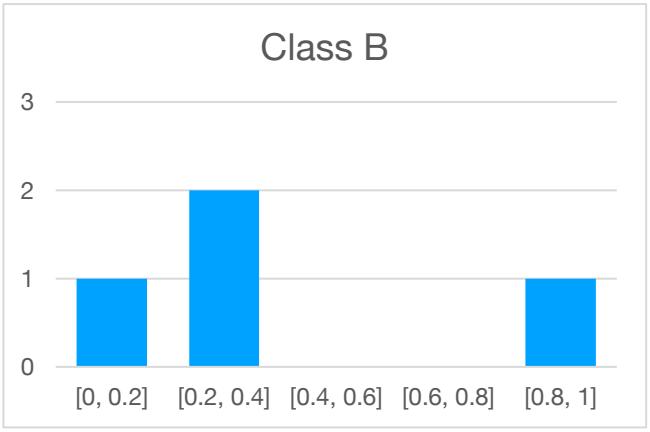


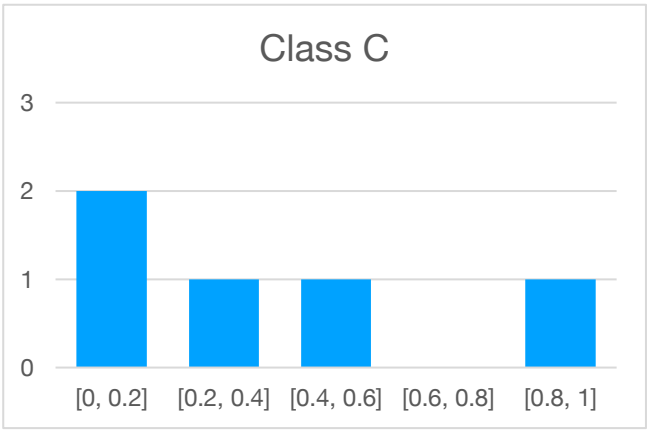
Answer: The class with the lowest F1 score is Class A.

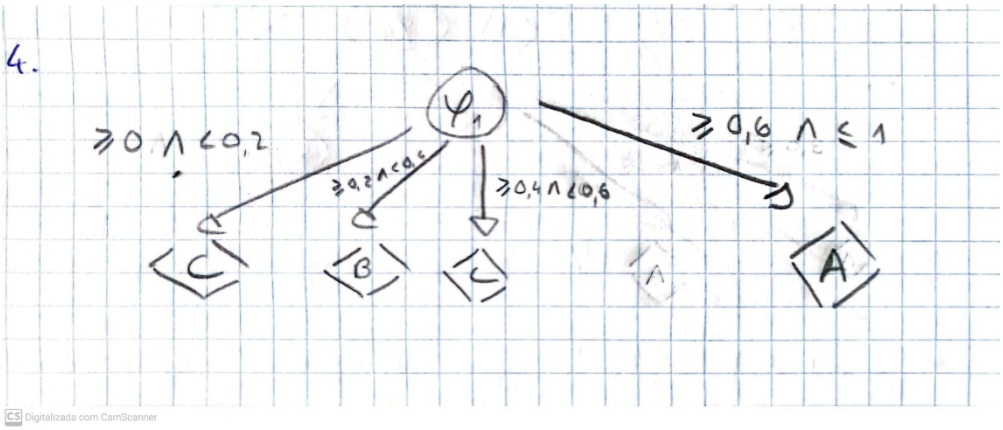
Homework I – Group 066

(ist1107301, ist1106794)

4)







TÉCNICO LISBOA

Aprendizagem 2024/25

Homework I – Group 066

(ist1107301, ist1106794)

II. Programming and critical analysis

Here we have an overview of all the imports used and how we imported the dataset, which are both common to all questions. Therefore, we will be omitting this code from the beginning of each task to avoid redundancy.

```
import pandas as pd, matplotlib.pyplot as plt, numpy as np
import seaborn as sns
from scipy.io.arff import loadarff
from sklearn.feature_selection import f_classif
from sklearn.metrics import accuracy_score
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

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

1)

Finding the input variable with the highest and lowest discriminative power

```
X = df.drop('Outcome', axis = 1)
df['Outcome'] = df['Outcome'].astype(int)
y = df['Outcome']

f_values, p_values = f_classif(X, y)

# Identify the best and worst input variables based on f-values
best = np.argmax(f_values)
worst = np.argmin(f_values)

fig = plt.figure(figsize=(12, 5))

print(f"The input variables with the worst discriminative power is: {X.columns[worst]} and the best is: {X.columns[best]}")
```

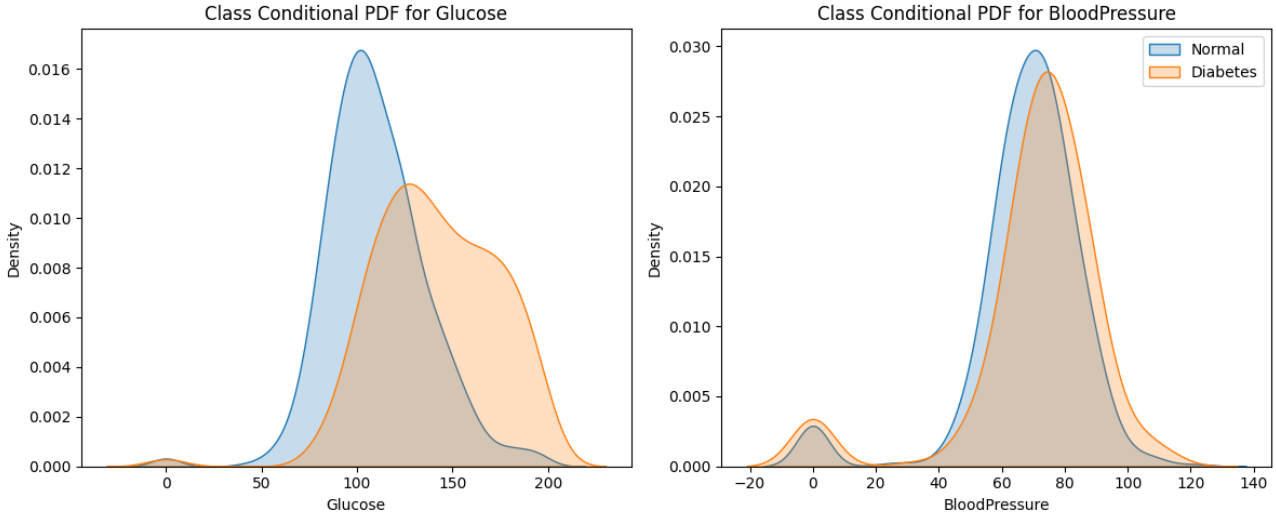
Printing the plot

```
# Best input variable
plt.subplot(1, 2, 1)
sns.kdeplot(df[df['Outcome'] == 0][X.columns[best]], label= 'Normal', fill= True)
sns.kdeplot(df[df['Outcome'] == 1][X.columns[best]], label= 'Diabetes', fill= True)
plt.xlabel(f'{X.columns[best]}')
plt.ylabel('Density')
plt.title(f'Class Conditional PDF for {X.columns[best]}')

# Worst input variable
plt.subplot(1, 2, 2)
sns.kdeplot(df[df['Outcome'] == 0][X.columns[worst]], label= 'Normal', fill= True)
sns.kdeplot(df[df['Outcome'] == 1][X.columns[worst]], label= 'Diabetes', fill= True)
plt.xlabel(f'{X.columns[worst]}')
plt.ylabel('Density')
plt.title(f'Class Conditional PDF for {X.columns[worst]}')

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

Homework I – Group 066

(ist1107301, ist1106794)

The input variables with the worst discriminative power is BloodPressure and the best is Glucose



2)

Processing the data and initializing all variables needed

```
X = df.drop('Outcome', axis = 1)
df['Outcome'] = df['Outcome'].astype(int)
y = df['Outcome']

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)

avg_train_accuracies, avg_test_accuracies = [], []
min_sample_splits = [2,5,10,20,30,50,100]
```

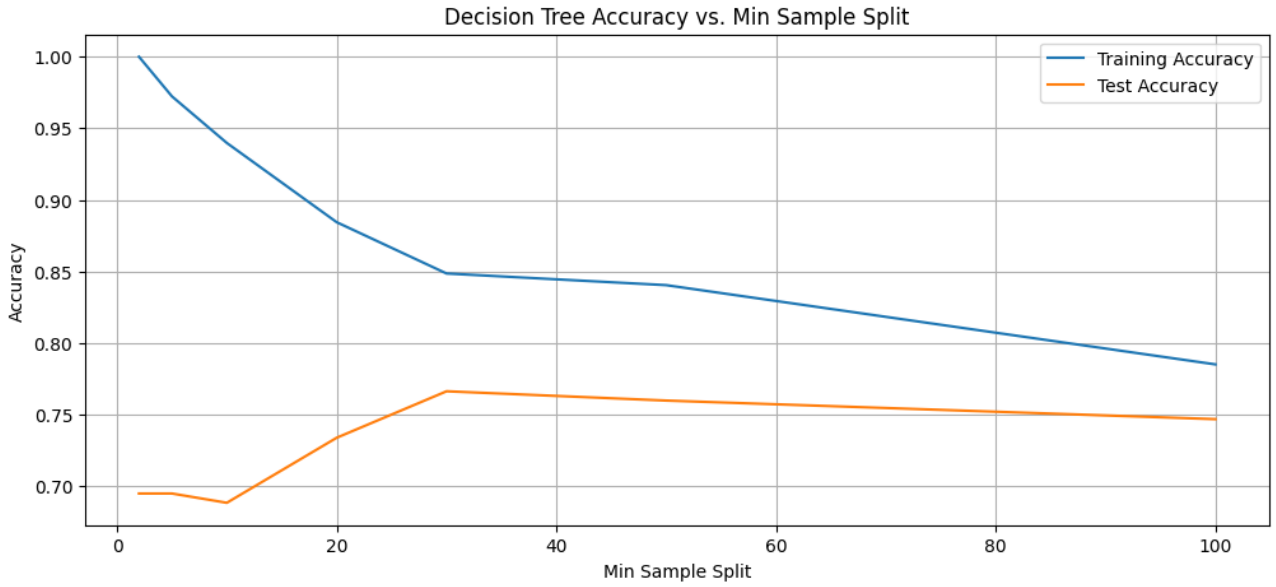
Training 10 times per minimum sample splits value to get a better accuracy and storing the average

```
for min_sample_split in min_sample_splits:
    train_acc_scores = []
    test_acc_scores = []

    # Run 10 times to improve accuracy
    for i in range(10):
        #Create a decision tree classifier
        clf = DecisionTreeClassifier(min_samples_split= min_sample_split, random_state=1)
        clf.fit(x_train, y_train)


        train_accuracy = accuracy_score(y_train, clf.predict(x_train))
        test_accuracy = accuracy_score(y_test, clf.predict(x_test))
        train_acc_scores.append(train_accuracy)
        test_acc_scores.append(test_accuracy)

    avg_train_accuracies.append(np.mean(train_acc_scores))
    avg_test_accuracies.append(np.mean(test_acc_scores))
```

Plotting the average accuracies

```
fig = plt.figure(figsize=(12, 5))
plt.plot(min_sample_splits, avg_train_accuracies, label='Training Accuracy')
plt.plot(min_sample_splits, avg_test_accuracies, label='Test Accuracy')
plt.title('Decision Tree Accuracy vs. Min Sample Split')
plt.xlabel('Min Sample Split')
plt.ylabel('Accuracy')
plt.legend()
plt.grid()
plt.show()
```

Results:



3) For better analysis, we can separate the results in two different ranges in the X axis (Min Sample Split).

- Min Sample Split < ≈30

The train accuracy starts at 100% for a Min Sample Split of 2. That makes perfect sense as it means that we always split the decision tree whenever we get two different outcomes for differencing attribute values. This will inevitably result in a perfect classifying decision tree for every instance in the training data set, explaining the perfect accuracy score. However, it scores lower than 70% accuracy in unseen data (test data).

As the Min Sample Split setting increases along this range, the train accuracy decreases and the test accuracy increases due to a better generalization of the results.

We can come to the conclusion that the decision trees created from smaller Min Sample Split values suffer from overfitting. They become too complex, overly learning patterns and noise in the training data and then scoring poorly in the test data.

- Min Sample Split > ≈30

As the setting for Min Sample Split values increase from 30, the test accuracy starts slowly decreasing along with the train accuracy. By this point the decision trees created are starting to become too simple, translating in worse accuracies for both data sets. They now indicate underfitting problems.

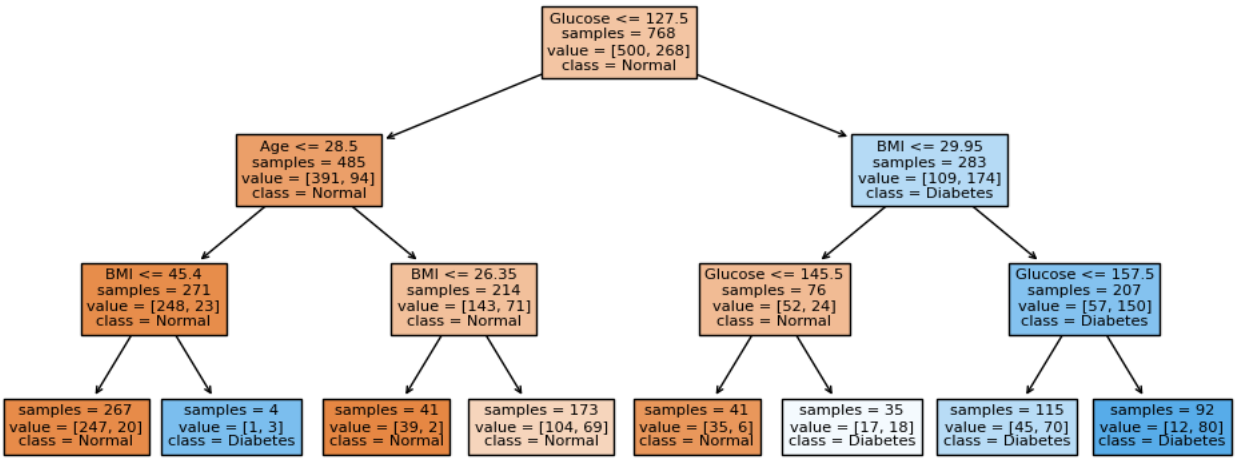
The best balance between complexity and generalization capacity is found setting the Min Sample Split value at around 30, where the maximum test accuracy of about 77% is reached. At this point, the model catches enough patterns in the data to perform well with unseen data, without overly fitting the training set.

4)

i. Creating, training and plotting the decision tree

```
X = df.drop('Outcome', axis = 1)
df['Outcome'] = df['Outcome'].astype(int)
y = df['Outcome']

predictor = tree.DecisionTreeClassifier(random_state=1, max_depth=3)
predictor.fit(X, y)

feature_names = X.columns
class_names = ['Normal', 'Diabetes']

plt.figure(figsize=(12, 5))
tree.plot_tree(predictor, feature_names=feature_names, class_names=class_names, impurity=False, filled=True)
plt.show()
```

Results:



ii. According to the decision tree created, diabetes is mostly characterized by high levels of glucose. However, other factors such as BMI and Age play a significant role in the matter too.

We can see that individuals with Glucose > 127.5 have a higher likelihood of being diabetic, as 174 out of 283 people with glucose levels above this threshold have diabetes. $P_{(Diabetes|Glucose>127,5)} = \frac{174}{174+109} = 61.5\%$. Inside this population, those with BMI > 29.95 have an even higher chance of having diabetes. $P_{(Diabetes|Glucose>127,5 \wedge BMI>29,5)} = \frac{150}{150+57} = 72\%$, while those with BMI ≤ 29,95 are classified as Normal most of the cases. $P_{(Normal|Glucose>127,5 \wedge BMI\leq 29,5)} = \frac{52}{52+24} = 68\%$.

On the other side of the tree, the population with Glucose ≤ 127,5 has a probability of only $\frac{94}{94+391} = 19\%$ of having diabetes. The probability only gets higher than 50% for people with Age ≤ 28,5 and BMI > 45,4. In this event, the posterior probability of having diabetes is $\frac{3}{3+1} = 75\%$.

In conclusion, the primary factor in distinguishing people with diabetes from people classified as normal is Glucose levels. For people with high levels of Glucose, their BMI is the next critical factor in the classification, and for people with lower levels, Age is the secondary decider.

END