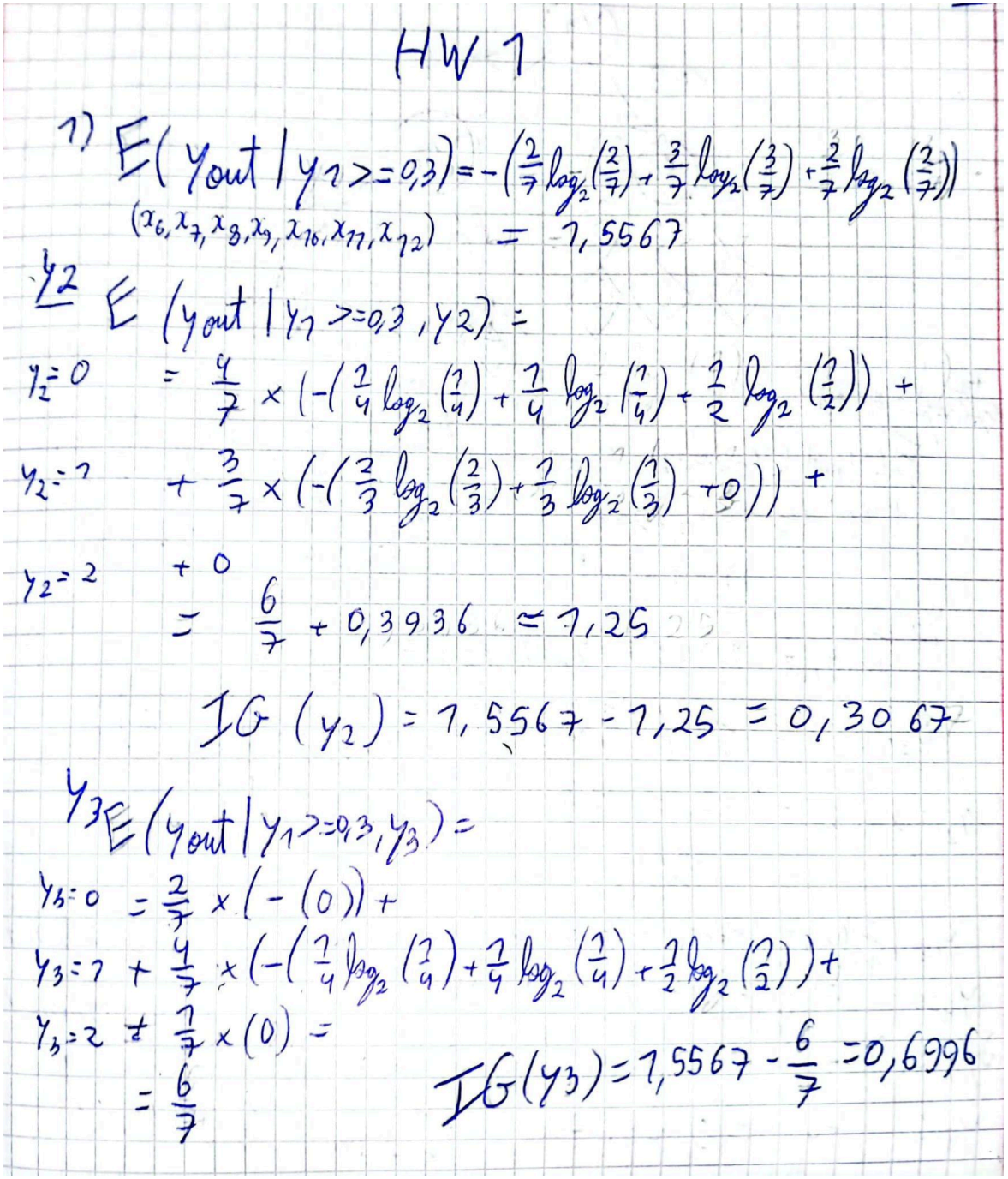
Aprendizagem 2024/25

Homework I – Group 040

(ist106307, ist106559)

I. Pen-and-paper

1) Complete the given decision tree using Shannon entropy (log2) 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.



Homework I – Group 040

(ist106307, ist106559)

y_4 - $E(y_{out} | y_1 >= 0,3, y_4) =$

$y_4=0 = \frac{4}{7} \times (-(\frac{1}{2}\log_2(\frac{1}{2}) + \frac{1}{2}\log_2(\frac{1}{2})))$

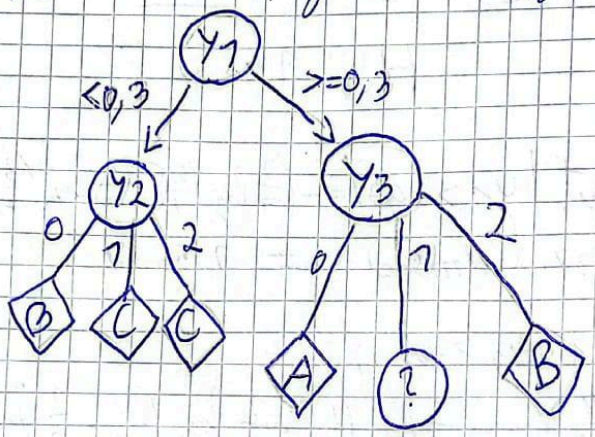
$y_4=1 + \frac{3}{7} \times (-(\frac{1}{3}\log_2(\frac{1}{3}) + \frac{2}{3}\log_2(\frac{2}{3})))$

$y_4=2 + 0$

$= \frac{4}{7} + 0,3936 \simeq 0,9650$

$IG(y_4) = 1,5567 - 0,965 = 0,5917$

IG de y_3 é o maior, logo escolhemos y_3.



Entropia do nó

$E(y_{out} | y_1 >= 0,3, y_3 = 1) = -(\frac{1}{4}\log_2(\frac{1}{4}) + \frac{1}{4}\log_2(\frac{1}{4}) + \frac{1}{2}\log_2(\frac{1}{2})) = 1,5$

$(x_6, x_7, x_9, x_{10}) >= 4$

i) > 4 → split!

y_2 $E(y_{out} | y_1 >= 0,3, y_3 = 1, y_2) =$

$y_2=0 = 1 \times (-(\frac{1}{4}\log_2(\frac{1}{4}) + \frac{1}{4}\log_2(\frac{1}{4}) + \frac{1}{2}\log_2(\frac{1}{2})))$

$y_2=1$ & $y_2=2$ $+0 +0 = 1,5$

$IG(y_2) = 0$

y_4 $E(y_{out} | y_1 >= 0,3, y_3 = 1, y_4) =$

$y_4=0 = \frac{1}{4} \times (-(0))$

$y_4=1 + \frac{3}{4} \times (-(\frac{1}{3}\log_2(\frac{1}{3}) + \frac{2}{3}\log_2(\frac{2}{3})))$

$y_4=2 + 0 = 0,6887$

$IG(y_4) = 0,8113$

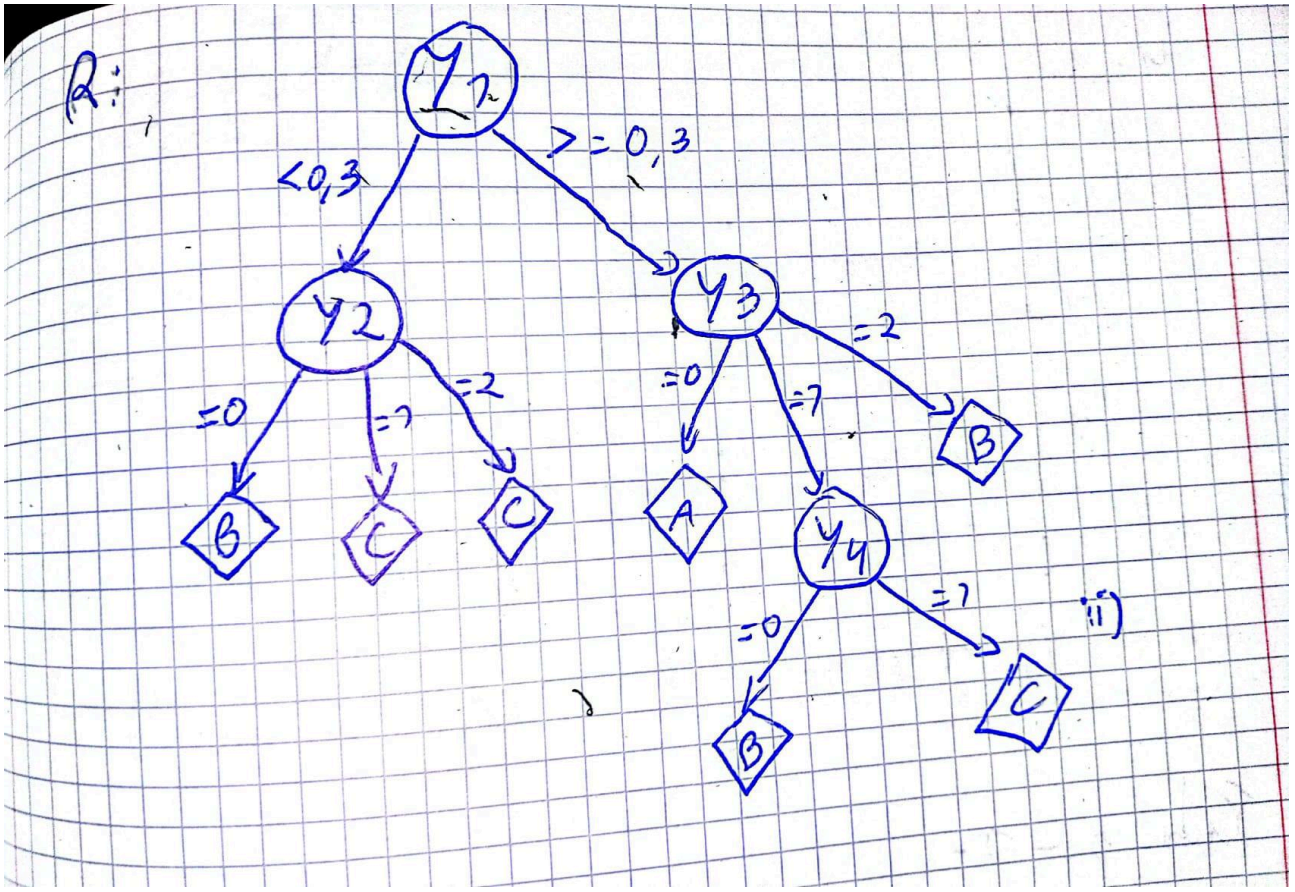
$IG(y_4) > IG(y_2)$

logo escolhemos y_4

Aprendizagem 2024/25

Homework I – Group 040

(ist106307, ist106559)



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

Yout = X

Output Real:

X = [C B C B C B A A C C A B]

Output Previsto:

X^=[C B C B C B C A C C A B]

Matriz de confusão:

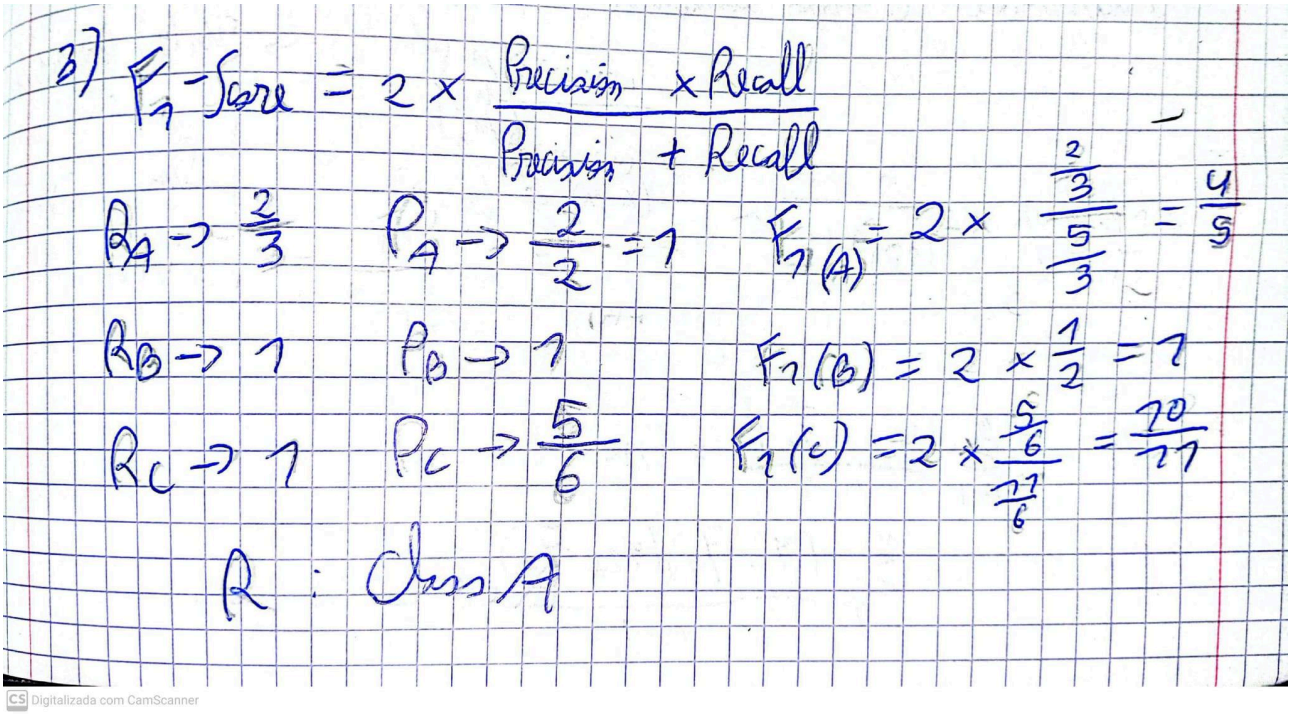
PREVISTOS

REAIS		A	B	C
	A	2	0	1
	B	0	4	0
	C	0	0	5

Homework I – Group 040

(ist106307, ist106559)

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



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

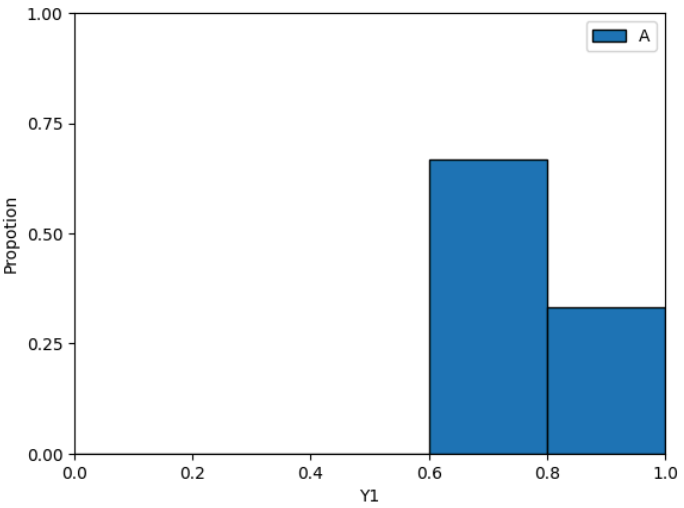


Fig1. Histograma 1 condicionado à Classe A

Aprendizagem 2024/25

Homework I – Group 040

(ist106307, ist106559)

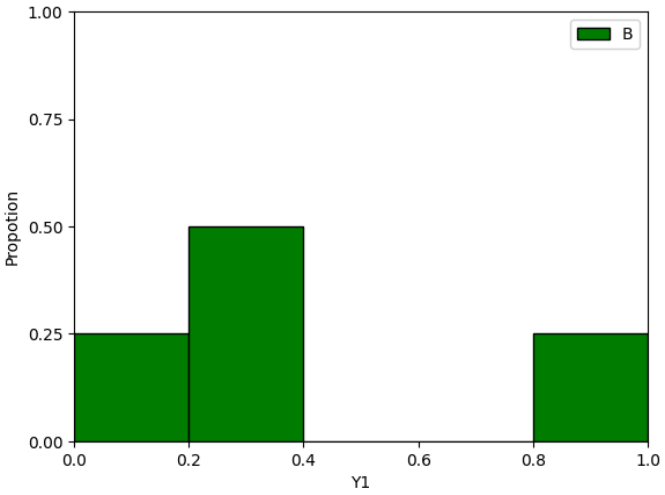


Fig2. Histograma 1 condicionado à Classe B

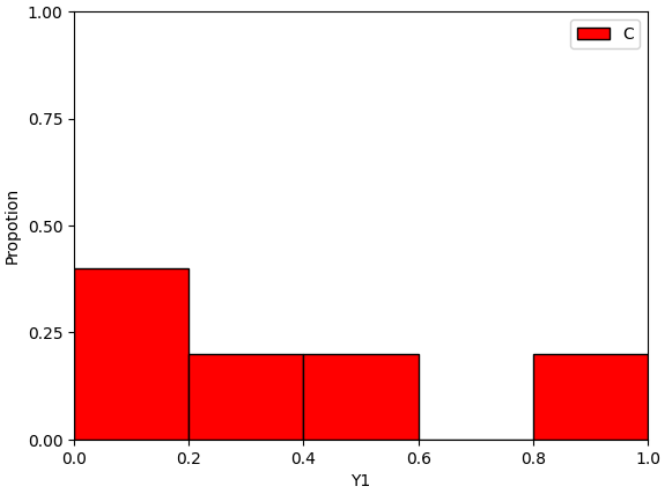
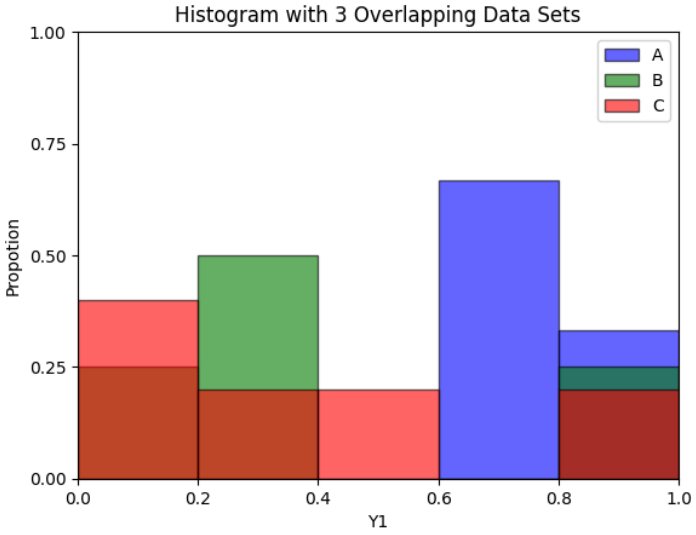


Fig3. Histograma 1 condicionado à Classe C

Para encontrar o *n-ary root split,* basta sobrepor os 3 histogramas:



As root split serão os pontos que separam os locais em que a classe condicional predominante muda:

$$y1 < 0.2 \Rightarrow C;$$
$$0.2 <= y1 < 0.4 \Rightarrow B;$$
$$0.4 <= y1 < 0.6 \Rightarrow C;$$
$$y1 >= 0.6 \Rightarrow A$$

II. Programming and critical analysis

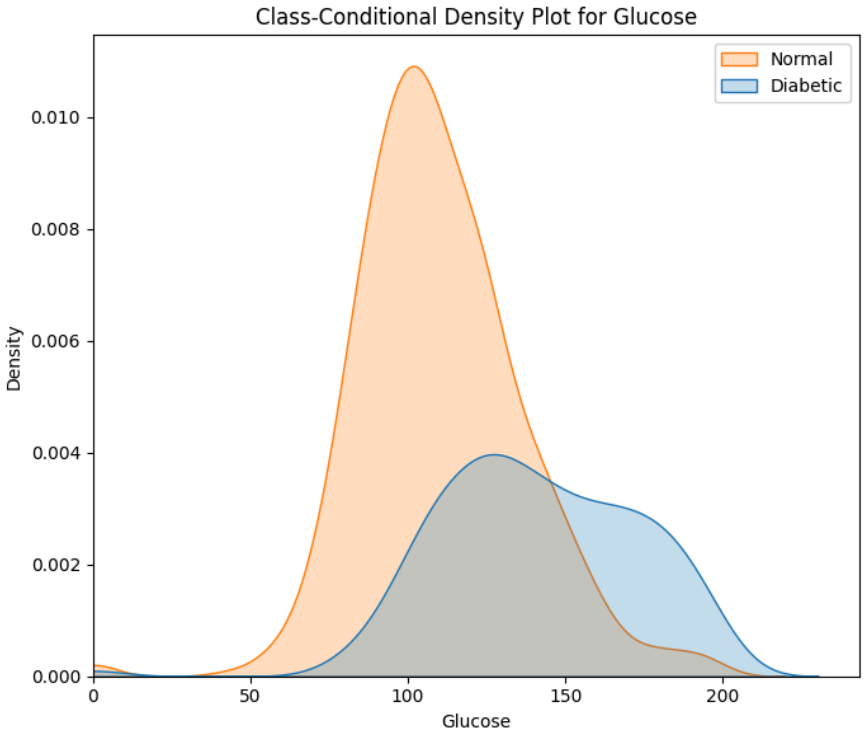
5) ANOVA is a statistical test that can be used to assess the discriminative power of a single input variable. Using f_classif from sklearn, identify the input variables with the worst and best discriminative power. Plot their class-conditional probability density functions:

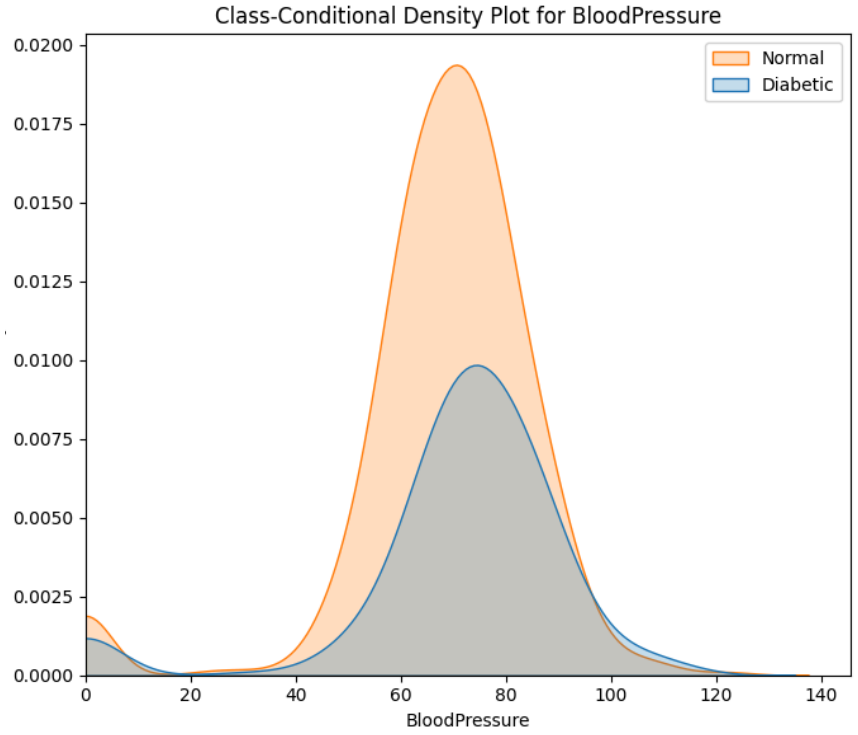
```python
import seaborn as sns
import pandas as pd
from scipy.io import arff
from sklearn.feature_selection import f_classif
import matplotlib.pyplot as plt
import numpy as np

data = arff.loadarff('diabetes.arff')
df = pd.DataFrame(data[0])

df['Outcome'] = df['Outcome'].str.decode('utf-8')
x = df.drop('Outcome', axis=1)
y = df['Outcome']

fimportance, p_value = f_classif(x, y)

feature_scores = sorted(zip(x.columns, fimportance), key=lambda x: x[1], reverse=True)
best_feature = feature_scores[0][0]
worst_feature = feature_scores[-1][0]
print(f"Variable with the best discriminative power: {best_feature}:{feature_scores[0][1]}")
print(f"Variable with the worst discriminative power: {worst_feature}:{feature_scores[-1][1]}")

plt.figure(figsize=(14, 6))

# Plot for the best feature
plt.subplot(1, 2, 1)
sns.kdeplot(data=df, x=best_feature, hue='Outcome', fill=True)
plt.title(f'Class-Conditional Density Plot for {best_feature}')
plt.xlim(left=0)
plt.legend(title='', labels=["Normal", "Diabetic"])
# Plot for the worst feature
plt.subplot(1, 2, 2)
sns.kdeplot(data=df, x=worst_feature, hue='Outcome', fill=True)
plt.title(f'Class-Conditional Density Plot for {worst_feature}')
plt.xlim(left=0)
plt.legend(title='', labels=["Normal", "Diabetic"])

plt.tight_layout()
```

```
Variable with the best discriminative power: Glucose:213.16175217803828
Variable with the worst discriminative power:BloodPressure:3.256950397889028
```

Para descobrir as variáveis com melhor e pior poder discriminativo é necessário calcular o valor da F-statistic para cada característica, sendo Glucose a variável com o valor mais alto e BloodPressure o menos elevado Logo a variável com maior poder discriminativo é Glucose e com menor poder discriminativo é BloodPressure.

As classes deste data set são Normal ou Diabetic, podemos então desenhar os gráficos das funções de densidade de probabilidade das variáveis referidas anteriormente:





6) 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.

```python
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

sample_split = [2, 5, 10, 20, 30, 50, 100]
X_train, X_test, y_train, y_test = train_test_split(x, y, train_size=0.8, stratify=y, random_state=1)

accuracies_train = []
accuracies_test = []

for split in sample_split:
    split_accuracies_train = []
    split_accuracies_test = []
    for i in range(10):
        predictor = DecisionTreeClassifier(min_samples_split=split, random_state=1)
        predictor.fit(X_train, y_train)
        y_pred = predictor.predict(X_train)
        y_pred1 = predictor.predict(X_test)
        split_accuracies_train.append(metrics.accuracy_score(y_train, y_pred))
        split_accuracies_test.append(metrics.accuracy_score(y_test, y_pred1))
    avg_accuracy_train = np.mean(split_accuracies_train)
    accuracies_train.append(avg_accuracy_train)
    avg_accuracy_test = np.mean(split_accuracies_test)
    accuracies_test.append(avg_accuracy_test)

accuracies_train = [round(acc, 2) for acc in accuracies_train]
accuracies_test = [round(acc, 2) for acc in accuracies_test]

print("Train accuracies: ", accuracies_train, "\nTest accuracies: ", accuracies_test)
plt.figure(figsize=(14, 6))
plt.plot(sample_split, accuracies_train, marker='o', label='Train Accuracy')
plt.plot(sample_split, accuracies_test, marker='o', label='Test Accuracy')
plt.grid()
plt.xticks(sample_split)
plt.title('Accuracy vs min_samples_split')
plt.xlabel('min_samples_split')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Train accuracies: [1.0, 0.97, 0.94, 0.88, 0.85, 0.84, 0.79]

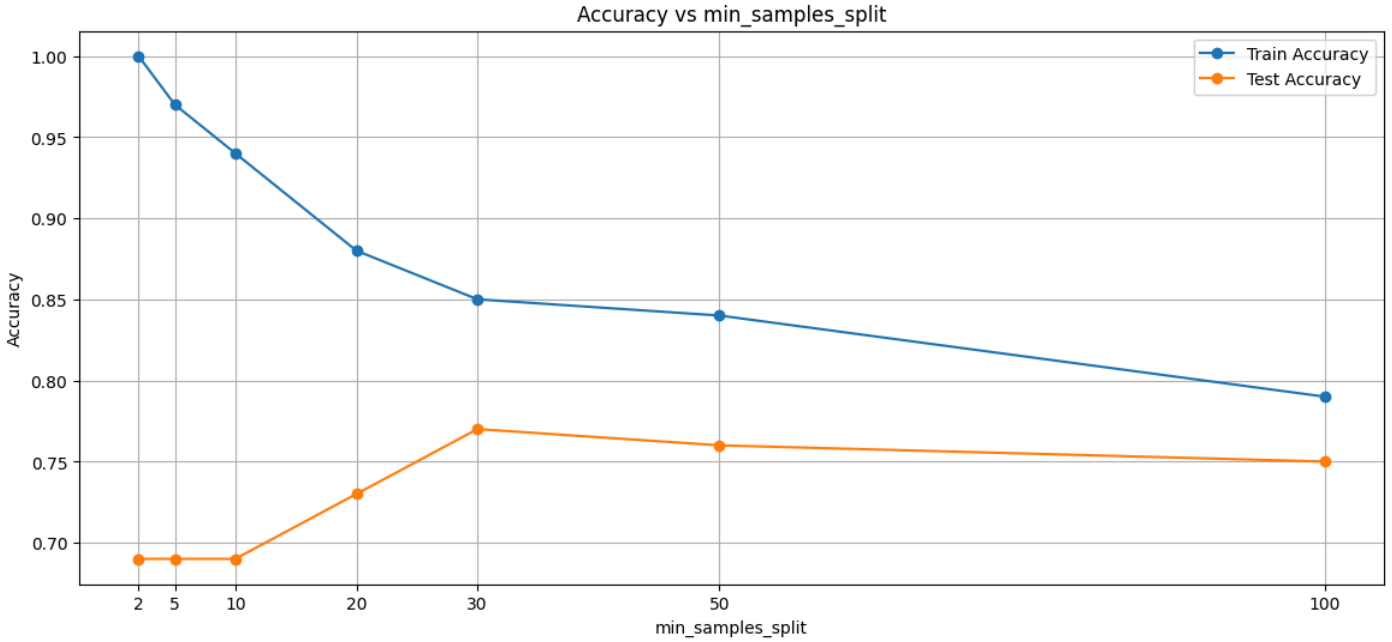
Test accuracies: [0.69, 0.69, 0.69, 0.73, 0.77, 0.76, 0.75]

Aprendizagem 2024/25

Homework I – Group 040

(ist106307, ist106559)

Como pedido, fizemos o split o dataset em 80% para dados de treino e 20% para dados de teste. Para obter a training accuracy e a test accuracy para um minimum sample split de {2,5,10, 20, 50, 100} é criada uma árvore com esses critérios e o modelo é treinado com os dados de treino e são obtidos os valores para as target variables. Por fim, são calculados os valores de accuracy de teste e de treino. De forma a obter resultados mais fidedignos, este processo é executado 10 vezes para cada valor do minimum sample split e é feito a média dos valores de accuracy, produzindo o seguinte gráfico de accuracies em função do minimum sample split:



7) Critically analyze these results, including the generalization capacity across settings.

Para valores baixos do minimum sample split ({2, 5, 10}), os resultados mostram sintomas de "overfitting", uma vez que o modelo tem um bom desempenho nos dados de treino ({1,0, 0,97, 0,94}), mas um desempenho fraco no conjunto de teste ({0,69, 0,69, 0,69}). Isso acontece porque o modelo se ajusta demais ao conjunto de treino e não se generaliza bem para novos dados.

À medida que a minimum sample split aumenta para valores intermédios ({20, 30}), a precisão do treino diminui ({0,88, 0,85}), mas a precisão do teste melhora ({0,73, 0,77}), atingindo o pico com uma minimum sample split de 30. Isto acontece porque, embora a precisão do treino diminua, a capacidade do modelo para generalizar para novos dados, possivelmente não abrangidos pelos dados de treino, aumenta, levando a uma melhor precisão do teste.

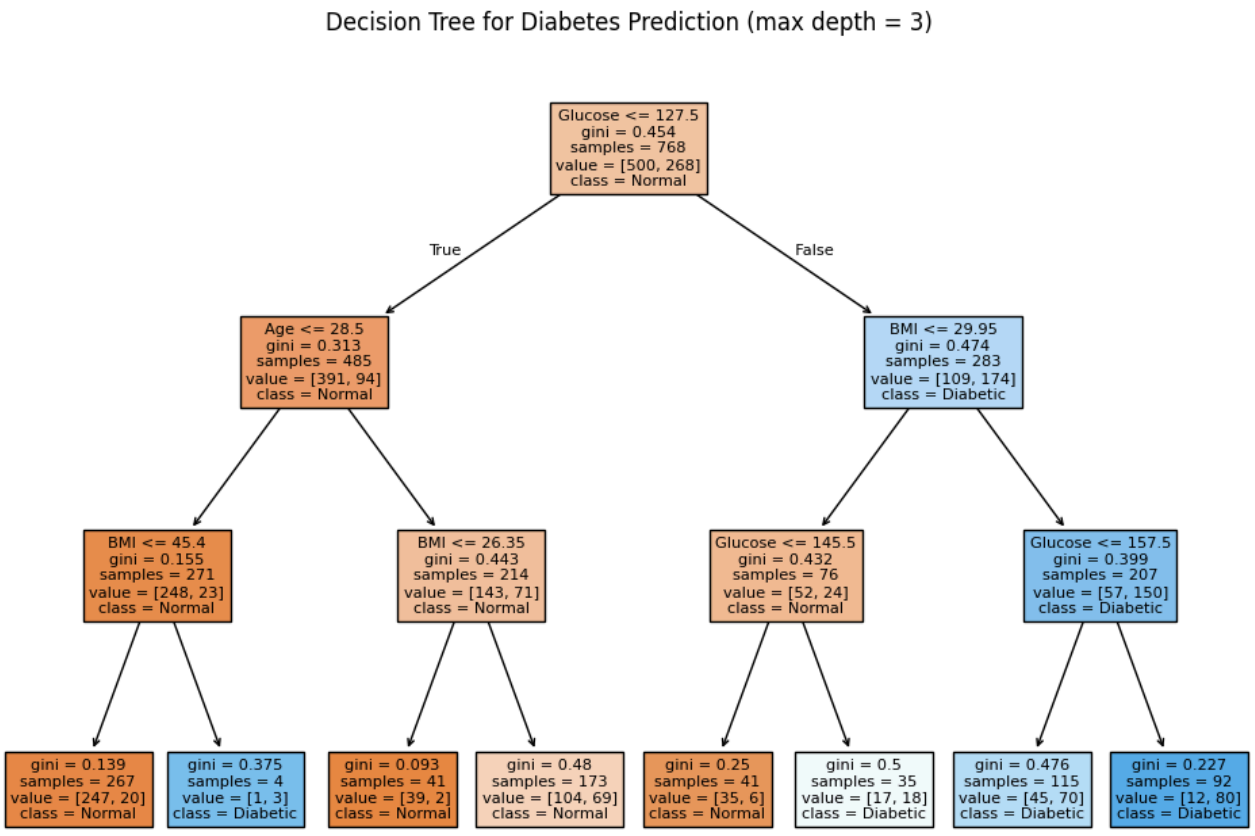
Para valores mais elevados do minimum sample split ({50, 100}), as accuracies de treino e de teste diminuem ({0,84, 0,79} para o treino e {0,76, 0,75} para o teste). Isto acontece porque o modelo se torna demasiado geral e pouco preciso, o que significa que a sua capacidade de previsão diminui ligeiramente tanto para os dados de treino como para os de teste.

Em conclusão, devem ser evitados valores baixos de minimum sample split para evitar "overfitting", mas também devem ser evitados valores excessivamente altos para evitar que o modelo se torne demasiado generalizado. O equilíbrio entre as precisões de treino e de teste é fundamental para escolher a divisão mínima de amostras ideal.

8) 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.

a. Plot the decision tree.

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
clf = DecisionTreeClassifier(max_depth=3, random_state=1)
clf.fit(x, y)

plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=x.columns, class_names=['Normal', 'Diabetic'], filled=True)
plt.title('Decision Tree for Diabetes Prediction (max depth = 3)')
plt.show()
```



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

Através da árvore, podemos observar que existem três grupos de pacientes diagnosticados com Diabetes, cada um com condições específicas e probabilidades posteriores associadas.

Para o primeiro grupo, os doentes têm de ter um nível de Glucose inferior ou igual a 127,5, ter idade igual ou inferior a 28,5 anos e ter um IMC superior a 45,4. Este grupo tem uma probabilidade posterior de 75% (3 em cada 4 doentes que correspondem a estas condições são diabéticos).

No segundo grupo, as condições incluem ter um nível de Glucose superior a 127,5 e um IMC superior a 29,95, com uma probabilidade posterior de 72,5% (150 de 207 doentes que correspondem a estas condições são diabéticos).

Finalmente, para o terceiro grupo, os doentes devem ter um nível de Glucose superior a 145,5 e um IMC inferior ou igual a 29,95, o que resulta numa probabilidade posterior de 51,4% (18 dos 35 doentes que correspondem a estas condições são diabéticos).

END