Aprendizagem 2024/2025

Homework I – Group 034

(ist199417, ist1106481)

I. Pen-and-paper

1)

Como os cálculos para este tipo de exercícios são algo extensivos e podem dar aso a erro, utilizámos python para computar a entropia e o ganho de informação para um dado conjunto de dados.
Carregamos os dados desta forma. Se quisermos tirar ou adicionar instâncias é só comentar/descomentar as linhas onde figuram.
A classe (yout) é a primeira (índice 0).

```
data  = []
# data += [["C", 0.22, 2, 0, 1]]
# data += [["B", 0.06, 0, 0, 0]]
# data += [["C", 0.16, 1, 2, 2]]
# data += [["B", 0.21, 0, 0, 0]]
data += [["C", 0.01, 2, 2, 0]]
data += [["B", 0.3 , 0, 1, 0]]
data += [["A", 0.76, 0, 1, 1]]
data += [["A", 0.86, 1, 0, 0]]
data += [["C", 0.93, 0, 1, 1]]
data += [["C", 0.47, 0, 1, 1]]
data += [["A", 0.73, 1, 0, 0]]
data += [["B", 0.89, 1, 2, 0]]
```

Calculamos a entropia da variável y_i como se segue:

```
import math

def entropy(data, i):

    hist = []
    for instance in data:
        found = False
        for pair in hist:
            value = pair[0]
            count = pair[1]
            if instance[i] == value:
                pair[1] += 1
                found = True
        if not found:
            hist += [[instance[i], 1]]

    total = len(data)

    entr = 0
    for pair in hist:
        entr -= pair[1]/total * math.log(pair[1]/total, 2)

    return entr
```

Calculamos a entropia da variável y_i em relação à variável y_{cond} da seguinte forma:

```
def entropy_cond(data, i, cond):

    hist = []
    for j in range(len(data)):
        found = False
        for pair in hist:
            value = pair[0]
            count = pair[1]
            if data[j][cond] == value:
                pair[1] += [j]
                found = True
        if not found:
            hist += [[data[j][cond], [j]]]

    total = len(data)

    entr = 0
    for pair in hist:
        data_subset = []
        for j in pair[1]:
            data_subset += [data[j]]
        entr += len(pair[1])/total * entropy(data_subset, i)

    return entr
```

Já temos tudo o que é preciso para calcular o ganho de informação da variável y_i.
Como variáveis discretas e contínuas se calculam de maneiras diferentes, temos duas versões.
Em information_gain_cont tem-se em conta todos os modos de dividir a variável contínua em dois grupos e escolhe-se o modo com maior ganho de informação.

```
def information_gain(data, i):
    return entropy(data, 0) - entropy_cond(data, 0, i)

def information_gain_cont(data, i):

    max_ig = 0
    cut = 0
    data = sorted(data, key=lambda x: x[i])
    for j in range(1, len(data)):
        if data[j - 1][i] != data[j][i]:
            data_subset1 = data[:j]
            data_subset2 = data[j:]
            entr = j / len(data) * entropy(data_subset1, 0) + (len(data) - j) / len(data) * entropy(data_subset2, 0)
            ig = entropy(data, 0) - entr
            if ig > max_ig:
                max_ig = ig
                cut = (data[j][i] + data[j - 1][i]) / 2

    return max_ig, cut
```

Agora basta chamar as funções e observar os resultados.

```
print(f"IG(y{1}) = {round(information_gain_cont(data, 1)[0], 3), information_gain_cont(data, 1)[1]}")
for i in [2, 3, 4]:
    print(f"IG(y{i}) = {round(information_gain(data, i), 3)}")
```

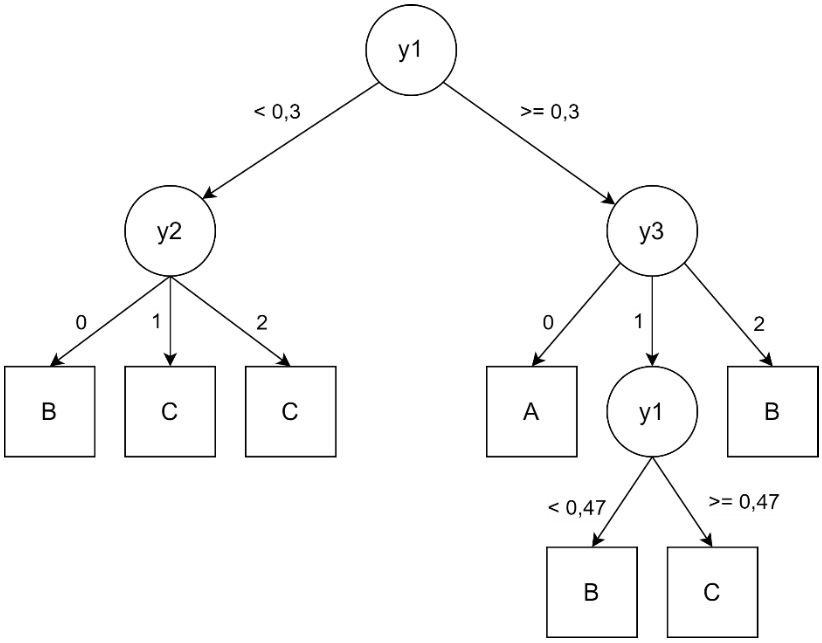
$IG(y_1) = (0.36, 0.6)$
$IG(y_2) = 0.467$
$IG(y_3) = 0.561$
$IG(y_4) = 0.266$
Como y_3 tem ganho de informação maior, escolhe-se a variável y_3. Observemos o subconjunto de dados depois da escolha $y_3 = 1$.

```
data  = []
# data += [["C", 0.22, 2, 0, 1]]
# data += [["B", 0.06, 0, 0, 0]]
# data += [["C", 0.16, 1, 2, 2]]
# data += [["B", 0.21, 0, 0, 0]]
#data += [["C", 0.01, 2, 2, 0]]
data += [["B", 0.3 , 0, 1, 0]]
data += [["A", 0.76, 0, 1, 1]]
# data += [["A", 0.86, 1, 0, 0]]
data += [["C", 0.93, 0, 1, 1]]
data += [["C", 0.47, 0, 1, 1]]
# data += [["A", 0.73, 1, 0, 0]]
# data += [["B", 0.89, 1, 2, 0]]

print(f"IG(y{1}) = {round(information_gain_cont(data, 1)[0], 3), information_gain_cont(data, 1)[1]}")
for i in [2, 3, 4]:
    print(f"IG(y{i}) = {round(information_gain(data, i), 3)}")
```

$IG(y1) = (0.811, 0.385)$
$IG(y2) = 0.0$
$IG(y3) = 0.0$
$IG(y4) = 0.811$
Como y_1 e y_4 têm ambos ganho de informação igual, escolhe-se por ordem alfabética y_1, com corte em 0.385.

Resposta Final:



2)

True / Predicted	A	B	C
A	2	0	0
B	0	4	0
C	1	0	5

3)

$$recall = \frac{TP}{TP + FN} \qquad precision = \frac{TP}{TP + FP}$$

$$recall_A = \frac{2}{3} \qquad precision_A = \frac{2}{2}$$

$$recall_B = \frac{4}{4} \qquad precision_B = \frac{4}{4}$$

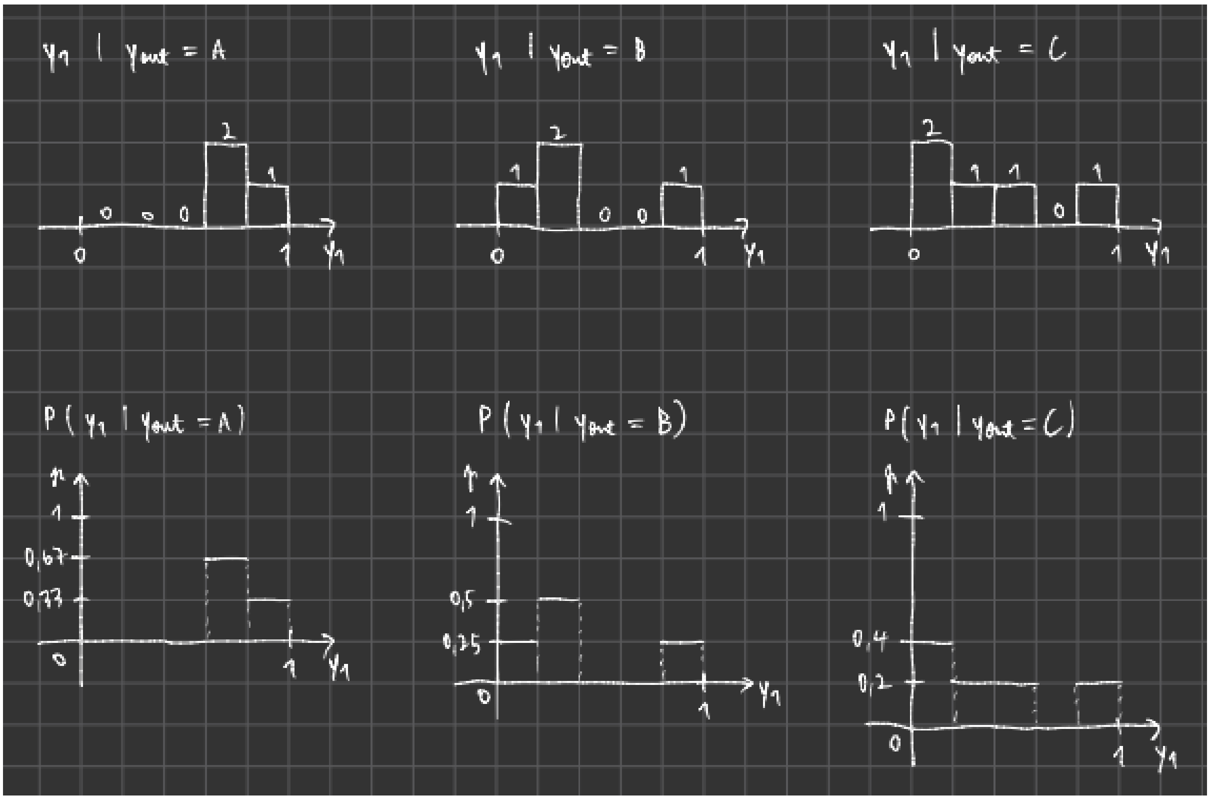
$$recall_C = \frac{5}{5} \qquad precision_C = \frac{5}{6}$$

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}}\ , P = precision, R = recall$$

$$F1_A = \frac{2}{\frac{1}{1} + \frac{3}{2}} = 0{,}80 \qquad F1_B = \frac{2}{\frac{1}{1} + \frac{1}{1}} = 1 \qquad F1_C = \frac{2}{\frac{6}{5} + \frac{1}{1}} = 0{,}91$$

R: A classe com um F1 score mais baixo é a classe A.

4)

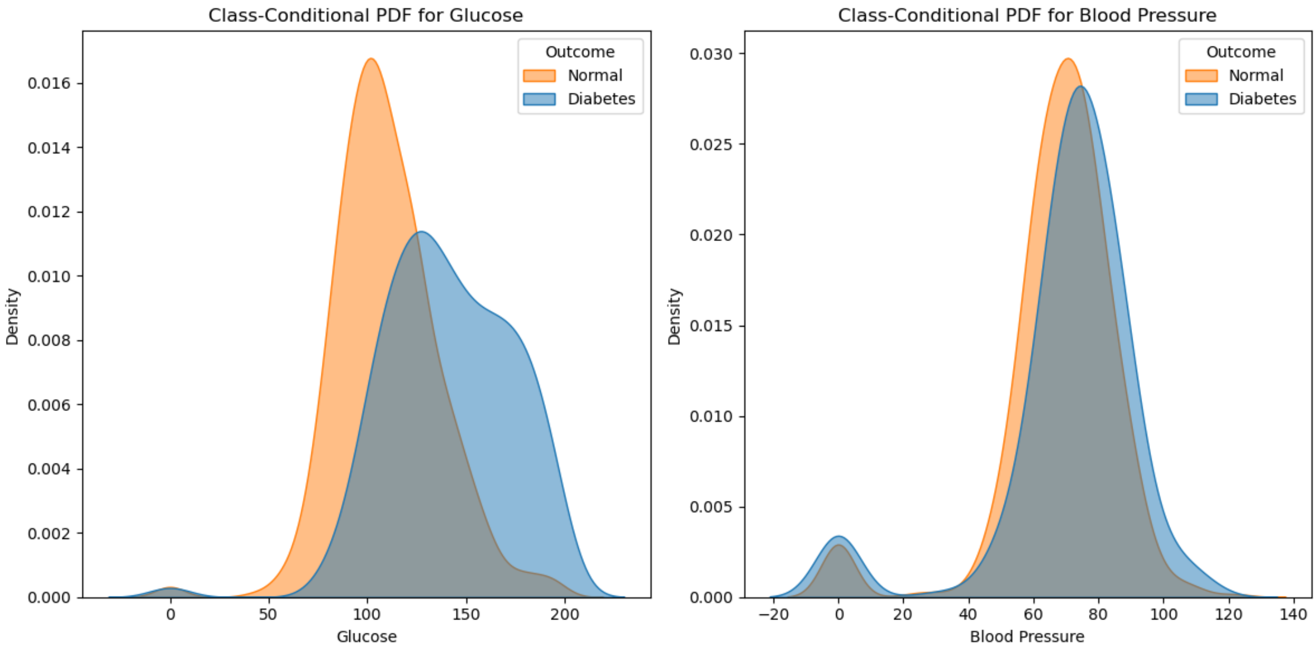


$$y_1 < 0{,}2 => C$$
$$0{,}2 \leq y_1 < 0{,}4 => B$$
$$0{,}4 \leq y_1 < 0{,}6 => C$$
$$0{,}6 \leq y_1 \quad => A$$

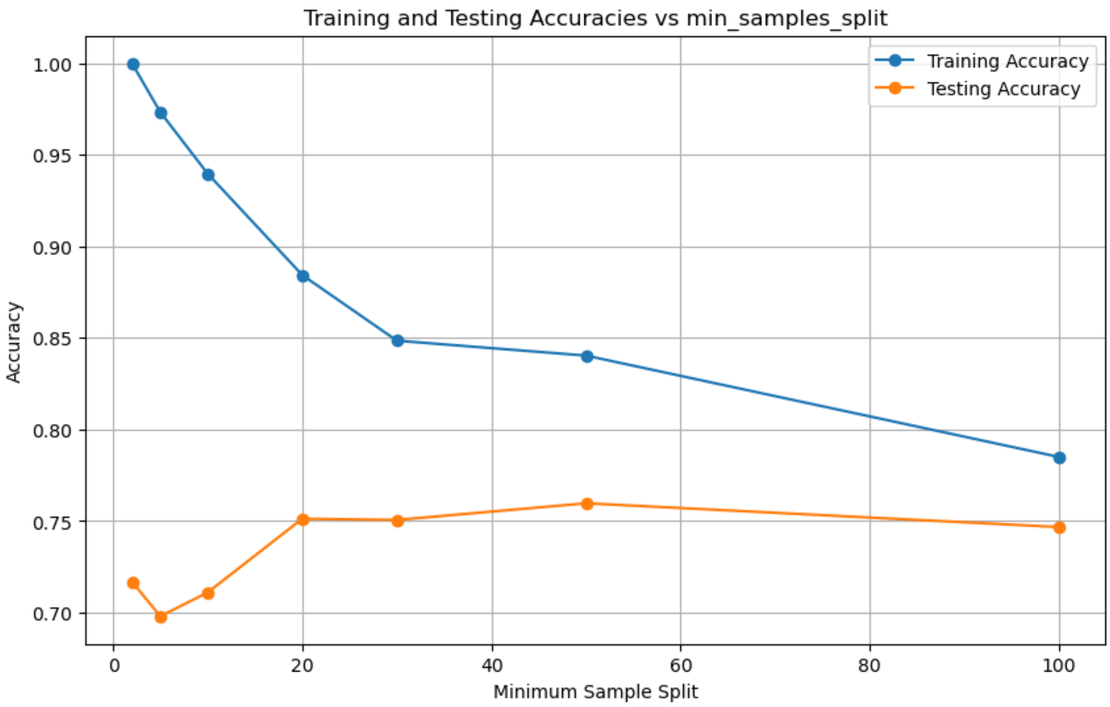
II. Programming

1)

Característica menos discriminativa: Pressão Arterial.
Característica mais discriminativa: Glicose.



2)



3)

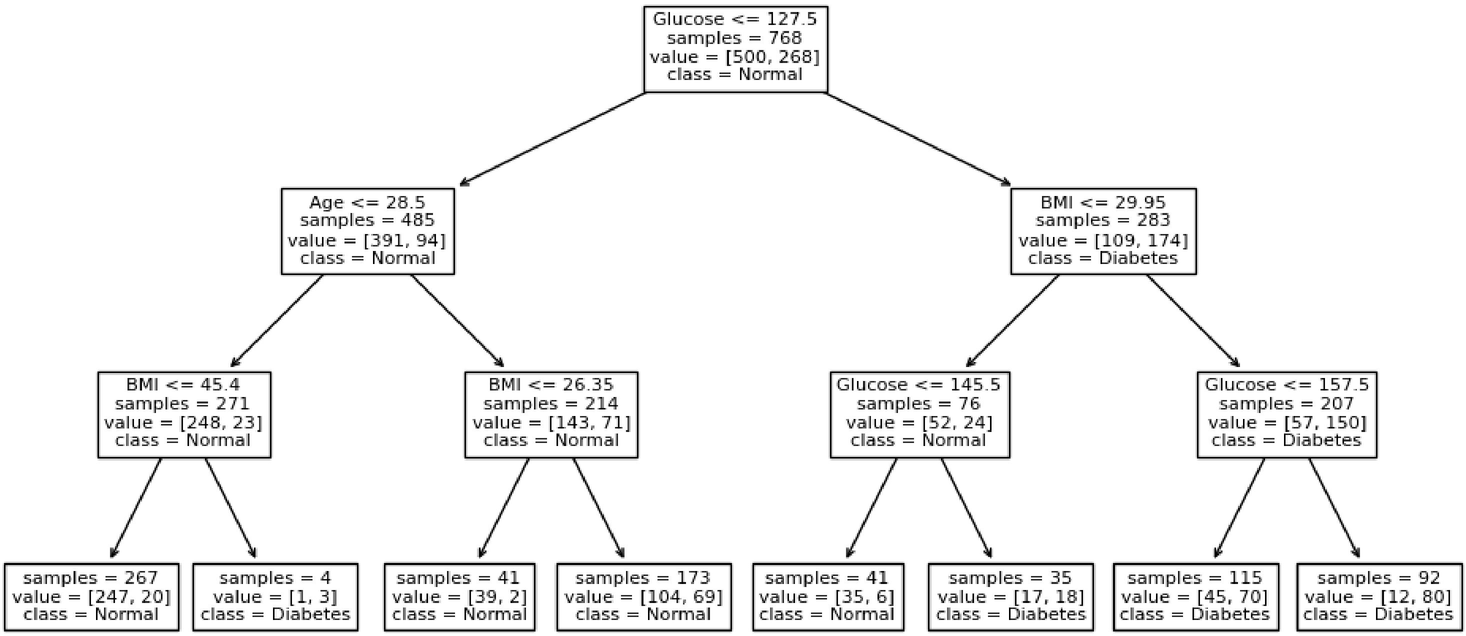
Em primeiro lugar, quando o minimum_sample_split é 2, a training accuracy é obviamente 1 porque todas as instâncias são classificadas nas suas verdadeiras classes. Não há nenhuma folha que contenha duas instâncias de classes diferentes. Mas isto tem um problema: o overfitting. Como vemos no gráfico, para o grupo de teste, a accuracy pode ser melhorada. O modelo tem um desempenho excelente no grupo de treino, mas tem pouca capacidade de generalização.

Para mitigar este problema, uma ideia é aumentar o minimum_sample_split. Podemos então olhar para o outro lado do espetro. Dado minimum_sample_split igual a 100, vemos uma diminuição na accuracy a nível do grupo de treino, embora a accuracy no grupo de teste tenha aumentado. Esta perda de accuracy pode ser evidência de underfitting.

Por isso, há que encontrar um balanço da accuracy nos dois grupos. Analisando o gráfico, observamos que ter o minimum_sample_split entre 30 e 50 é o que nos dá esse tal balanço. A testing accuracy estabiliza daí para a frente e a training accuracy desce.

4)

i.



ii.

$\text{Glucose} \leq 127{,}5 \ \& \ \text{Age} \leq 28{,}5 \ \& \ \text{BMI} \leq 45{,}4 \rightarrow P(\text{Diabetes}) = 7{,}5\%$

$\text{Glucose} \leq 127{,}5 \ \& \ \text{Age} \leq 28{,}5 \ \& \ \text{BMI} > 45{,}4 \rightarrow P(\text{Diabetes}) = 75\%$

$\text{Glucose} \leq 127{,}5 \ \& \ \text{Age} > 28{,}5 \ \& \ \text{BMI} \leq 26{,}35 \rightarrow P(\text{Diabetes}) = 4{,}9\%$

$\text{Glucose} \leq 127{,}5 \ \& \ \text{Age} > 28{,}5 \ \& \ \text{BMI} > 26{,}35 \rightarrow P(\text{Diabetes}) = 39{,}9\%$

$127{,}5 < \text{Glucose} \leq 145{,}5 \quad \& \ \text{BMI} \leq 29{,}95 \rightarrow P(\text{Diabetes}) = 14{,}6\%$

$\text{Glucose} > 145{,}5 \quad \& \ \text{BMI} \leq 29{,}95 \rightarrow P(\text{Diabetes}) = 51{,}4\%$

$127{,}5 < \text{Glucose} \leq 157{,}5 \quad \& \ \text{BMI} > 29{,}95 \rightarrow P(\text{Diabetes}) = 60{,}9\%$

$\text{Glucose} > 157{,}5 \quad \& \ \text{BMI} > 29{,}95 \rightarrow P(\text{Diabetes}) = 87\%$