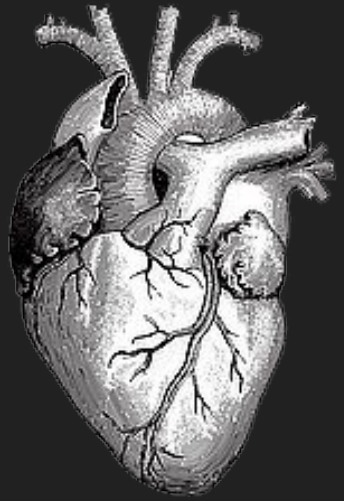
Trabalho de Modulagem

Primeiro Trabalho Aprendizado de Máquina e Modelagem de Conhecimento Incerto

Alunos: João Pedro Alkamim
Sarah Anduca de Oliveira

O Problema



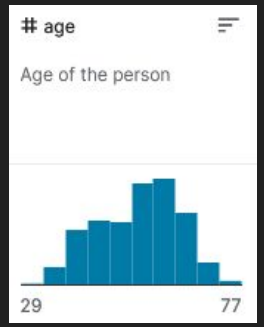
O uso de um banco de dados sobre ataque cardíaco tem várias vantagens. Por exemplo, ele pode ser usado para identificar padrões e tendências em relação aos ataques cardíacos, o que é útil para a prevenção e o tratamento desta doença. Além disso, ele pode ser usado para melhorar a eficácia dos tratamentos e a qualidade dos cuidados prestados aos pacientes com ataque cardíaco. O banco de dados também pode ser utilizado para desenvolver novas terapias e tratamentos para a doença.

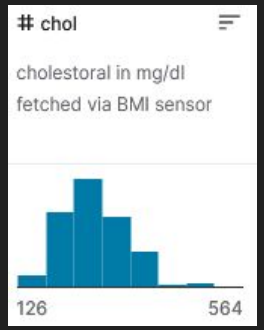
Ele é uma ferramenta valiosa que pode ajudar a entender e tratar a doença. Portanto iremos utilizá-lo para que consigamos desenvolver um programa apto em dizer se a pessoa precisa ou não de tratamento de um médico, evitando assim o ataque cardíaco.

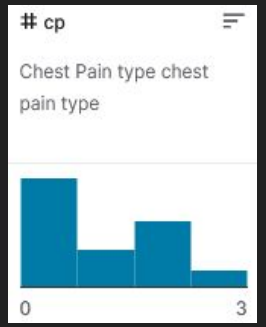
Variáveis

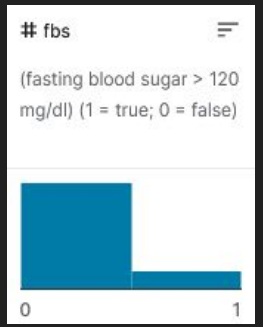
As variáveis escolhidas foram:

- Idade (Age)
- Colesterol (Chol)
- Dores no peito (Chest Pain)
- Açúcar no sangue (Fasting blood sugar)









Variáveis

Cada variável foi distribuída em 3 conjuntos diferentes. As tabelas com cada informação está abaixo

Variável	Conjunto 1	Conjunto 2	Conjunto 3
Age	x < 45	45 <= x < 56	56 <= x
Chol	x < 200	200 <= x < 240	240 <= x
Chest Pain	x = 2 ou x = 3	x = 1	x = 0
FBS	x = 1 (se > 120)	x = 0 (caso contrário)	-

Variáveis

Conjunto 1: Jovem/Adulto;

Conjunto 2: Idade intermediária;

Conjunto 3: Adulto/Idoso.

Variável	Conjunto 1	Conjunto 2	Conjunto 3
Age	x < 45	45 <= x < 56	56 <= x

Variáveis

Conjunto 1: Colesterol ideal;

Conjunto 2: Colesterol superior;

Conjunto 3: Colesterol indesejável.

Variável	Conjunto 1	Conjunto 2	Conjunto 3
Chol	x < 200	200 <= x < 240	x <= 240

Variáveis

Conjunto 1: Assintomático;

Conjunto 2: Angina atípica;

Conjunto 3: Angina

Variável	Conjunto 1	Conjunto 2	Conjunto 3
Chest Pain	x = 2 ou x = 3	x = 1	x = 0

Variáveis

Conjunto 1: Caso o açúcar no sangue esteja superior a 120

Conjunto 2: Caso contrário

Variável	Conjunto 1	Conjunto 2	Conjunto 3
FBS	x = 1 (se > 120)	x = 0 (caso contrário)	-

Variáveis

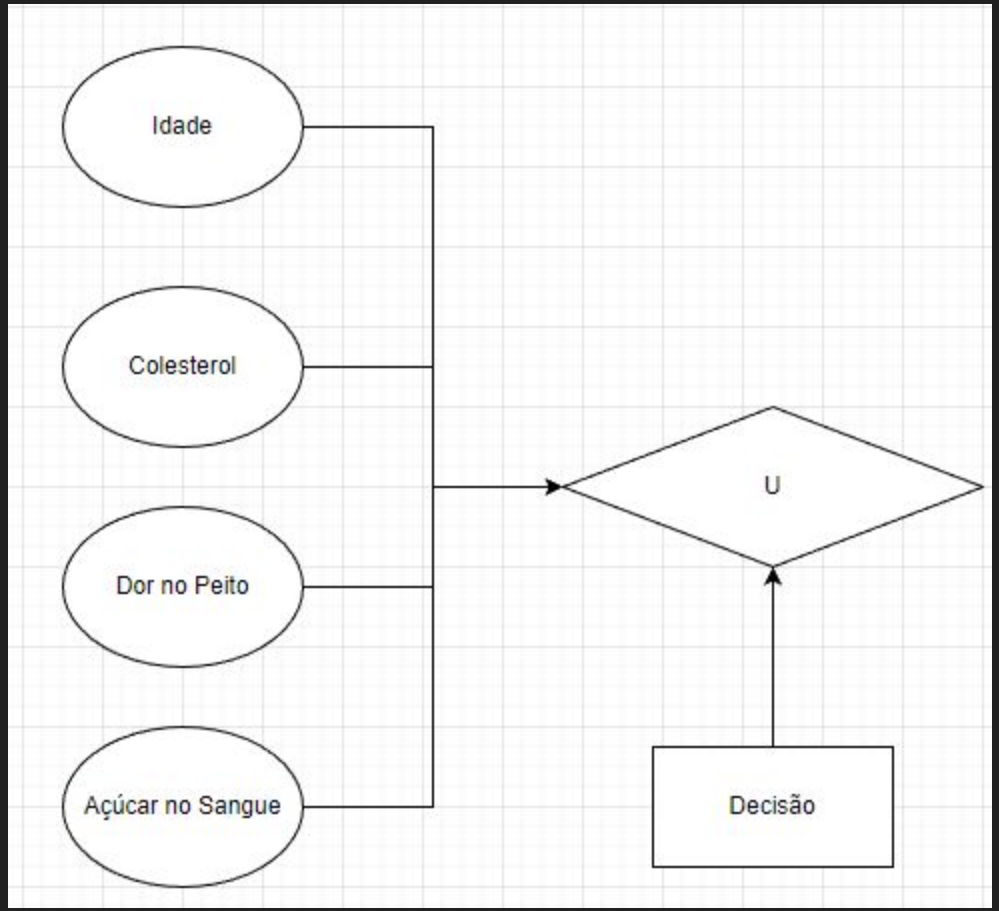
Todas as probabilidades de ter um ataque cardíaco foram tiradas da seguinte *database:*

https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset?resource=download

E são as seguintes:

Variável	Conjunto 1	Conjunto 2	Conjunto 3
Age	0,138613	0,138613	0,267323
Chol	0,280528	0,135313	0,128712
Chest Pain	0,099009	0,194719	0,250825
FBS	0,468646	0,759075	-

Modelo



Decisões

Escolhemos 3 decisões baseadas nessas 4 variáveis, que são:

- Coração saudável, sem problemas de saúde;
- É recomendável melhorar a alimentação;
- Procurar um médico.

Baseada nessas decisões foram criadas 4 tabelas de utilidade em uma escala de 0 a 150 para cada variável.

Decisões

Tabela de utilidade para a idade:

Decisão	Conjunto 1	Conjunto 2	Conjunto 3
Saudável	60	20	5
Alimentação	25	50	20
Médico	15	30	75

Decisões

Tabela de utilidade para o colesterol:

Decisão	Conjunto 1	Conjunto 2	Conjunto 3
Saudável	70	20	5
Alimentação	20	50	15
Médico	10	30	120

Decisões

Tabela de utilidade para dor no peito:

Decisão	Conjunto 1	Conjunto 2	Conjunto 3
Saudável	75	21	7
Alimentação	15	50	28
Médico	10	29	65

Decisões

Tabela de utilidade para açúcar no sangue:

Decisão	Conjunto 1	Conjunto 2	Conjunto 3
Saudável	10	3	-
Alimentação	15	10	-
Médico	3	15	-

Código implementado

Foi criada uma classe *Dados* que recebe todas as variáveis e calcula de acordo com a probabilidade a sua utilidade.

```
class Dados:

    def __init__(self, age, cp, chol, fbs):
        self.age = age
        self.cp = cp
        self.chol = chol
        self.fbs = fbs
        self.classAge = None
        self.classCp = None
        self.classChol = None
        self.classFbs = None
```

```
def setClassAge(self):
    if self.age < 45:
        result = [probAge1 * utilAge1[0], probAge1 *
                  utilAge1[1], probAge1 * utilAge1[2]]
        self.classAge = 1
        return result
    elif self.age < 53:
        result = [probAge2 * utilAge2[0], probAge2 *
                  utilAge2[1], probAge2 * utilAge2[2]]
        self.classAge = 2
        return result
    else:
        result = [probAge3 * utilAge3[0], probAge3 *
                  utilAge3[1], probAge3 * utilAge3[2]]
        self.classAge = 3
        return result
```

Código implementado

```
def setClassCp(self):
    if self.cp >= 2:
        result = [probCp1 * utilCp1[0], probCp1 *
                  utilCp1[1], probCp1 * utilCp1[2]]
        self.classCp = 1
        return result
    elif self.cp == 1:
        result = [probCp2 * utilCp2[0], probCp2 *
                  utilCp2[1], probCp2 * utilCp2[2]]
        self.classCp = 2
        return result
    else:
        result = [probCp3 * utilCp3[0], probCp3 *
                  utilCp3[1], probCp3 * utilCp3[2]]
        self.classCp = 3
        return result

def setClassChol(self):
    if self.chol < 200:
        result = [probChol1 * utilChol1[0], probChol1 *
                  utilChol1[1], probChol1 * utilChol1[2]]
        self.classChol = 1
        return result
    elif self.chol < 240:
        result = [probChol2 * utilChol2[0], probChol2 *
                  utilChol2[1], probChol2 * utilChol2[2]]
        self.classChol = 2
        return result
    else:
        result = [probChol3 * utilChol3[0], probChol3 *
                  utilChol3[1], probChol3 * utilChol3[2]]
        self.classChol = 3
        return result
```

```
def setClassFbs(self):
    if self.fbs == 0:
        result = [probFbs1 * utilFbs1[0], probFbs1 *
                  utilFbs1[1], probFbs1 * utilFbs1[2]]
        self.classFbs = 1
        return result
    else:
        result = [probFbs2 * utilFbs2[0], probFbs2 *
                  utilFbs2[1], probFbs2 * utilFbs2[2]]
        self.classFbs = 2
        return result
```

Código implementado

Por fim ele compara todos os resultados e devolve a decisão que tem a maior utilidade:

```
results = [resultAge, resultCp, resultChol, resultFbs]

big_value = -1
big_list = None

for i in range(len(results)):
    max_value = max(results[i])
    if max_value > big_value:
        big_value = max_value
        index_list = i
```

```
if index_list == 0:
    if d.classAge == 1:
        print("Coração ta saudável, sem problemas de saúde.")
    elif d.classAge == 2:
        print("É recomendável melhorar sua alimentação.")
    else:
        print("Procure um médico imediatamente.")
```

```
elif index_list == 1:
    if d.classCp == 1:
        print("Coração ta saudável, sem problemas de saúde.")
    elif d.classCp == 2:
        print("É recomendável melhorar sua alimentação.")
    else:
        print("Procure um médico imediatamente.")
elif index_list == 2:
    if d.classChol == 1:
        print("Coração ta saudável, sem problemas de saúde.")
    elif d.classChol == 2:
        print("É recomendável melhorar sua alimentação.")
    else:
        print("Procure um médico imediatamente.")
else:
    if d.classFbs == 1:
        print("Coração ta saudável, sem problemas de saúde.")
    else:
        print("É recomendável melhorar sua alimentação.")
```

Referências

https://www.hermespardini.com.br/blog/?p=76

https://www.cdc.gov/mmwr/volumes/71/wr/mm7123a4.htm

https://www.textbookofcardiology.org/wiki/Chest_Pain_/_Angina_Pectoris