BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho

1. Nesta questão você deve utilizar a base Student Performance, archive.ics.uci.edu/ml/datasets/Student+Performance (ver arquivo student-mat.csv no student.zip).

(a) (5 pontos) Explique qual a forma mais adequada para converter todos os atributos da base para numéricos.

R:

1. atributos binários: **[school, sex, address, famsize, Pstatus, schoolsup, famsup, paid, activities, nursery, higher, internet, romantic]**

2. atributos não-binários e não-ordinais: **[Mjob, Fjob, reason, guardian]**

1. **converter todos os binários para o formato: [-1, 1]**

2. **Para os não-binários, criar uma nova coluna para cada valor na coluna original e atribuir o valor 1 quando o valor do atributo bater com o nome da nova coluna.**

- **Já para a conversão da classe, imaginei uma média de nota no intervalo [7.0…10.0] para aprovado e abaixo disso para reprovado.**

(b) (10 pontos) Converta todos os atributos da base para numéricos (exceto a classe).

```
import numpy as np

import pandas as pd

import math

pd.options.mode.chained_assignment = None

from IPython.display import HTML
```

```
f = open("data/student-mat.csv", "r")

lines = f.readlines()

for x in range(len(lines)):
    lines[x] = lines[x].replace(';',',')
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho


```
f2 = open("data/student-mat-processado.csv", "w")

for x in lines:
    f2.write(x)

df = pd.read_csv("data/student-mat-processado.csv")
```

1°) função - conversão de atributos binários nas colunas.
2°) função - criação de colunas com nomes customizados dos valores únicos das colunas não binárias.
3°) função - alterando os nomes dos valores dentro da própria coluna para facilitar a comparação.

```
def converter_binario(coluna):
    unicos = coluna.unique()
    for x in range(len(coluna)):
        if(coluna[x] == unicos[0]):
            coluna[x] = -1
        else:
            coluna[x] = 1
    return coluna

def criar_novas_colunas(coluna, df):
    for x in range(coluna.unique().size):
        df[str(coluna.name)+ "_" + coluna.unique()[x]] = tamanho
    return df

def acrescentar_nome_valores(coluna):
    for x in range(coluna.size):
        coluna[x] = coluna.name + "_" + coluna[x]

    return coluna
```

primeiro loop criando um array de tamanho das colunas preenchido com zero para ser o valor das novas colunas criadas facilitando a distribuição dos valores após.

segundo loop distribuindo o valor “1” para as colunas que batem com o valor da coluna original

```
df.school = converter_binario(df.school)
```


```
df.sex = converter_binario(df.sex)
df.address = converter_binario(df.address)
df.famsize = converter_binario(df.famsize)
df.Pstatus = converter_binario(df.Pstatus)
df.schoolsup = converter_binario(df.schoolsup)
df.famsup = converter_binario(df.famsup)
df.paid = converter_binario(df.paid)
df.activities = converter_binario(df.activities)
df.nursery = converter_binario(df.nursery)
df.higher = converter_binario(df.higher)
df.internet = converter_binario(df.internet)
df.romantic = converter_binario(df.romantic)

tamanho = []

for y in range(df.Mjob.size):
    tamanho.append(0)

g3 = df.pop("G3")
df = criar_novas_colunas(df.Mjob, df)
df = criar_novas_colunas(df.Fjob, df)
df = criar_novas_colunas(df.reason, df)
df = criar_novas_colunas(df.guardian, df)
df.Mjob = acrescentar_nome_valores(df.Mjob)
df.Fjob = acrescentar_nome_valores(df.Fjob)
df.reason = acrescentar_nome_valores(df.reason)
df.guardian = acrescentar_nome_valores(df.guardian)

df["G3"] = g3

for index, row in df.iterrows():
    df[row['Mjob']][index] = 1
    df[row['Fjob']][index] = 1
    df[row['reason']][index] = 1
    df[row['guardian']][index] = 1

#HTML(df.to_html(index=False))
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho


```
df = df.drop(columns=['Mjob', 'Fjob', 'reason', 'guardian']) # comentar essa e a próxima linha para comparar mais facilmente antes de retirar essas colunas, retirar o comentário anterior.

HTML(df.to_html(index=False))
```

ures	schoolsup	famsup	paid	activities	nursery	higher	internet	romantic	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	Mjob_at_home	Mjob_health	Mjob_other	Mjob_services	Mjob_teacher	Fjob_teacher	Fjob_other
0	-1	-1	-1	-1	-1	-1	-1	-1	4	3	4	1	1	3	6	5	6	1	0	0	0	0	1	0
0	1	1	-1	-1	1	-1	1	-1	5	3	3	1	1	3	4	5	5	1	0	0	0	0	0	1
3	-1	-1	1	-1	-1	-1	1	-1	4	3	2	2	3	3	10	7	8	1	0	0	0	0	0	1
0	1	1	1	1	-1	-1	1	1	3	2	2	1	1	5	2	15	14	0	1	0	0	0	0	0
0	1	1	1	-1	-1	-1	-1	-1	4	3	2	1	2	5	4	6	10	0	0	1	0	0	0	1
0	1	1	1	1	-1	-1	1	-1	5	4	2	1	2	5	10	15	15	0	0	0	1	0	0	1
0	1	-1	-1	-1	-1	-1	1	-1	4	4	4	1	1	3	0	12	12	0	0	1	0	0	0	1
0	-1	1	-1	-1	-1	-1	-1	-1	4	1	4	1	1	1	6	6	5	0	0	1	0	0	1	0
0	1	1	1	-1	-1	-1	1	-1	4	2	2	1	1	1	0	16	18	0	0	0	1	0	0	1
0	1	1	1	1	-1	-1	1	-1	5	5	1	1	1	5	0	14	15	0	0	1	0	0	0	1
0	1	1	1	-1	-1	-1	1	-1	3	3	3	1	2	2	0	10	8	0	0	0	0	1	0	0
0	1	1	-1	1	-1	-1	1	-1	5	2	2	1	1	4	4	10	12	0	0	0	1	0	0	1
0	1	1	1	1	-1	-1	1	-1	4	3	3	1	3	5	2	14	14	0	1	0	0	0	0	0
0	1	1	1	-1	-1	-1	1	-1	5	4	3	1	2	3	2	10	10	0	0	0	0	1	0	1
0	1	1	-1	-1	-1	-1	1	1	4	5	2	1	1	3	0	14	16	0	0	1	0	0	0	1
0	1	1	-1	-1	-1	-1	1	-1	4	4	4	1	2	2	4	14	14	0	1	0	0	0	0	1
0	1	1	1	1	-1	-1	1	-1	3	2	3	1	2	2	6	13	14	0	0	0	1	0	0	0
0	-1	1	-1	1	-1	-1	-1	-1	5	3	2	1	1	4	4	8	10	0	0	1	0	0	0	1
3	1	1	-1	1	-1	-1	1	-1	5	5	5	2	4	5	16	6	5	0	0	0	1	0	0	0
0	1	-1	1	1	-1	-1	1	-1	3	1	3	1	3	5	4	8	10	0	1	0	0	0	0	1
0	1	-1	-1	-1	-1	-1	1	-1	4	4	1	1	1	1	0	13	14	0	0	0	0	1	0	1
0	1	1	1	-1	-1	-1	1	-1	5	4	2	1	1	5	0	12	15	0	1	0	0	0	0	0
0	1	-1	-1	1	-1	-1	1	-1	4	5	1	1	3	5	2	15	15	0	0	0	0	1	0	1
0	1	1	-1	1	-1	-1	1	-1	5	4	4	2	4	5	0	13	13	0	0	1	0	0	0	1
0	-1	1	1	1	-1	-1	1	-1	4	3	2	1	1	5	2	10	9	0	0	0	1	0	0	0
2	1	1	1	-1	1	-1	1	-1	1	2	2	1	3	5	14	6	9	0	0	0	1	0	0	0
0	1	1	1	-1	-1	-1	1	-1	4	2	2	1	2	5	2	12	12	0	0	1	0	0	0	1
0	1	-1	1	-1	-1	-1	1	-1	2	2	4	2	4	1	4	15	16	0	1	0	0	0	0	0
0	-1	1	-1	1	-1	-1	1	-1	5	3	3	1	1	5	4	11	11	0	0	0	1	0	0	1
0	1	1	1	1	-1	-1	1	1	4	4	5	5	5	5	16	10	12	0	0	0	0	1	1	0
0	1	1	1	-1	1	-1	1	-1	5	4	2	3	4	5	0	9	11	0	1	0	0	0	0	0
0	1	1	-1	1	-1	-1	1	-1	4	3	1	1	1	5	0	17	16	0	0	0	1	0	0	0
0	1	1	-1	1	-1	-1	1	1	4	5	2	1	1	5	0	17	16	0	0	0	0	1	0	0
0	1	-1	-1	1	1	-1	1	-1	5	3	2	1	1	2	0	8	10	0	0	1	0	0	0	1
0	1	1	1	-1	1	-1	1	-1	5	4	3	1	1	5	0	12	14	0	0	1	0	0	0	1

(c) (10 pontos) Assuma a última coluna (G3, que representa a nota final de cada estudante) como classe. Converta esta coluna (atributo numérico) para uma variável categórica binária. Após esta conversão é possível realizar a tarefa a seguir.

```
for x in range(df.G3.size):
    if df.G3[x] < 14:
        df.G3[x] = "reprovado"
    else:
        df.G3[x] = "aprovado"

classe = df[df.columns[-1::]]

classe
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho

UFAPE

```
classe
✓ 0.7s
```

	G3
0	reprovado
1	reprovado
2	reprovado
3	aprovado
4	reprovado
...	...
390	reprovado
391	aprovado
392	reprovado
393	reprovado
394	reprovado

395 rows × 1 columns

(d) (5 pontos) Calcule o intervalo de confiança da acurácia para o 100 repetições de holdout 50/50 utilizando o classificador 1-NN com distância Euclidiana.

Todo esse trecho de código já foi explicado em atividades anteriores

```
atributos = df[df.columns[:df.columns.size-1:]]

classe = np.array(df[df.columns[-1::]]).flatten()
```

```
def calcular_taxas_para_knn(k, qtd_taxas, porcent_test, atributos,
classe):
    array_taxas = []

    for n in range(k):
        arrays = []
```


```
        for i in range(qtd_taxas):
            x_treino, x_teste, y_treino, y_teste =
train_test_split(atributos, classe,test_size=porcent_test,
random_state=i)
            classificador = KNeighborsClassifier(n_neighbors=n+1)
            classificador.fit(x_treino, y_treino)
            taxas = classificador.score(x_teste, y_teste)
            arrays.append(taxas)

        array_taxas.append(arrays)

    return array_taxas

array_taxas = calcular_taxas_para_knn(1, 100, 0.5, atributos, classe)
```

```
def calcular_intervalo_confianca(hash_medias, hash_diferencas_total):
    for i in range(len(hash_medias)):
        for x in range(len(hash_medias[i+1])):

            desvio_padrao = np.std(hash_diferencas_total[i+1][x])

            erro_padrao =
desvio_padrao/np.sqrt(len(hash_diferencas_total[i+1][x]))

            multiplicador = abs(stats.distributions.norm.ppf(0.025))

            print(i+1, (hash_medias[i+1][x] -
multiplicador*erro_padrao, hash_medias[i+1][x] +
multiplicador*erro_padrao)
            )
```

```
def calcular_medias_separadas(array):

    hash_saida = {}
    array_aux = []
    for x in range(len(array)):
        array_aux.append(np.mean(array[x]))
        hash_saida[x+1] = array_aux
        array_aux = []
```

```
    return hash_saida

medias_separadas = calcular_medias_separadas(array_taxas)
```

```
hash_taxas = {}
array_aux = []
for x in range(len(array_taxas)):
    array_aux.append(array_taxas[x])
    hash_taxas[x+1] = array_aux
    array_aux = []

calcular_intervalo_confianca(medias_separadas, hash_taxas)
```

```
✓ 0.4s
1 (0.9000670424237401, 0.9066395444026071)
```

2. Utilizando a base Forest Fires. archive.ics.uci.edu/ml/datasets/Forest+Fires

(a) (5 pontos) Indique a forma mais adequada de converter para numéricos cada um dos atributos da base.

R:

Os atributos a serem convertidos no dataset são cíclicos, portanto há uma forma “especial” de converter esses valores, já que um valor anterior ao atual na realidade é mais próximo do mesmo do que um valor duas vezes a frente por exemplo, então para manter essa proximidade de valores é feito a transformação do atributo em dois atributos, sen e cos.

(b) (10 pontos) Realize a conversão da base conforme a resposta indicada.

```
import numpy as np
import pandas as pd
import math
pd.options.mode.chained_assignment = None
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho

Os appends são feitos já de maneira a deixar o array ordenado para que os valores cíclicos fiquem com a distância correta

```
df = pd.read_csv("data/forestfires.csv")

semana_aux = df.day.unique()

day = []

day.append(semana_aux[3])
day.append(semana_aux[4])
day.append(semana_aux[1])
day.append(semana_aux[5])
day.append(semana_aux[6])
day.append(semana_aux[0])
day.append(semana_aux[2])

hash_conversao_dias = {}

day_num = []

for x in range(len(day)):
    day_num.append(x+1)


for x in range(df.day.size):
    df.day[x] = day.index(df.day[x])

tamanho = []

for y in range(df.day.size):
    tamanho.append(0)
```

Criação das novas colunas

```
df['dia_sen'] = tamanho

df['dia_cos'] = tamanho
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho

UFAPE

Cálculo para os valores cíclicos

```
dois_pi = (2 * math.pi)
for x in range(df.day.size):
    df['dia_sen'][x] = math.sin((dois_pi * df.day[x]/7))
    df['dia_cos'][x] = math.cos((dois_pi * df.day[x]/7))
```

```
meses_aux = df.month.unique()

meses_ordenados = []
meses_ordenados.append(meses_aux[8])
meses_ordenados.append(meses_aux[7])
meses_ordenados.append(meses_aux[0])
meses_ordenados.append(meses_aux[4])
meses_ordenados.append(meses_aux[10])
meses_ordenados.append(meses_aux[5])
meses_ordenados.append(meses_aux[6])
meses_ordenados.append(meses_aux[2])
meses_ordenados.append(meses_aux[3])
meses_ordenados.append(meses_aux[1])
meses_ordenados.append(meses_aux[11])
meses_ordenados.append(meses_aux[9])

for x in range(df.month.size):
    df.month[x] = meses.index(df.month[x])

df['mes_sen'] = tamanho

df['mes_cos'] = tamanho
```

```
for x in range(df.month.size):
    df['mes_sen'][x] = math.sin((dois_pi * df.month[x]/12))
    df['mes_cos'][x] = math.cos((dois_pi * df.month[x]/12))

df = df.drop(columns=['month', 'day'])

df
```

```
df = df.drop(columns=['month', 'day'])

df
```

✓ 0.8s

	X	Y	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area	dia_sen	dia_cos	mes_sen	mes_cos
0	7	5	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.00	-0.974928	-0.222521	0.866025	5.000000e-01
1	7	4	90.6	35.4	669.1	6.7	18.0	33	0.9	0.0	0.00	0.974928	-0.222521	-1.000000	-1.836970e-16
2	7	4	90.6	43.7	686.9	6.7	14.6	33	1.3	0.0	0.00	-0.781831	0.623490	-1.000000	-1.836970e-16
3	8	6	91.7	33.3	77.5	9.0	8.3	97	4.0	0.2	0.00	-0.974928	-0.222521	0.866025	5.000000e-01
4	8	6	89.3	51.3	102.2	9.6	11.4	99	1.8	0.0	0.00	0.000000	1.000000	0.866025	5.000000e-01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
512	4	3	81.6	56.7	665.6	1.9	27.8	32	2.7	0.0	6.44	0.000000	1.000000	-0.500000	-8.660254e-01
513	2	4	81.6	56.7	665.6	1.9	21.9	71	5.8	0.0	54.29	0.000000	1.000000	-0.500000	-8.660254e-01
514	7	4	81.6	56.7	665.6	1.9	21.2	70	6.7	0.0	11.16	0.000000	1.000000	-0.500000	-8.660254e-01
515	1	4	94.4	146.0	614.7	11.3	25.6	42	4.0	0.0	0.00	-0.781831	0.623490	-0.500000	-8.660254e-01
516	6	3	79.5	3.0	106.7	1.1	11.8	31	4.5	0.0	0.00	0.974928	-0.222521	-0.866025	5.000000e-01

517 rows × 15 columns

3. Utilizando a base Car Evaluation. archive.ics.uci.edu/ml/datasets/Car+Evaluation

(a) (5 pontos) Indique a forma mais adequada de converter para numéricos cada um dos atributos da base.

R:

para os atributos (buying, maint, lug_boot e safety) por terem uma ordem clara (low < med < high < v-high) para os dois primeiros e (small/low < med < high) para os dois últimos transformei em ordinal não-binário na própria coluna cada um deles, para os atributos (persons e doors) por terem valores claros e não necessariamente consecutivos como por exemplo não há carros com 3 portas, utilizei “não-binário nominal”) criando um novo atributo para cada valor dos atributos antigos.

(b) (10 pontos) Realize a conversão da base conforme a resposta indicada.

```
import numpy as np
import pandas as pd
from scipy import stats
pd.options.mode.chained_assignment = None
```

UFAPE

```
nomes_col = ['comprar', 'manutencao', 'portas', 'pessoas',
'porta_malas', 'seguranca', 'class']

df = pd.read_csv("data/car.data", names=nomes_col)

comprar = df.comprar.unique()

comprar = sorted(comprar)

array_ordenado_comp = [comprar[1], comprar[2], comprar[0], comprar[3]]

manutencao = df.manutencao.unique()

array_ordenado_manu = list(manutencao[::-1])

portas = sorted(list(df.portas.unique()))

tamanho_malas = list(df.porta_malas.unique())

pessoas = sorted(list(df.pessoas.unique()))

seguranca = list(df.seguranca.unique())

print(comprar)

print(manutencao)

print(portas)

print(pessoas)

print(tamanho_malas)

print(seguranca)

for x in range(df.comprar.size):
    df.comprar[x] = array_ordenado_comp.index(df.comprar[x])

for x in range(df.manutencao.size):
```


```
    df.manutencao[x] = array_ordenado_manu.index(df.manutencao[x])

for x in range(df.portas.size):
    if df.portas[x] != portas[len(portas)-1]:
        df.portas[x] = df.portas[x] + "_portas"
    else:
        df.portas[x] = "mais_de_" + portas[len(portas)-2] + "_portas"

for x in range(df.porta_malas.size):
    df.porta_malas[x] = tamanho_malas.index(df.porta_malas[x])

for x in range(df.pessoas.size):
    if df.pessoas[x] != pessoas[len(pessoas)-1]:
        df.pessoas[x] = df.pessoas[x] + "_pessoas"
    else:
        df.pessoas[x] = "mais_de_" + pessoas[len(pessoas)-2] +
"_pessoas"

for x in range(df.seguranca.size):
    df.seguranca[x] = seguranca.index(df.seguranca[x])
```

```
novas_colunas = []

for x in range(df.comprar.size):
    novas_colunas.append(0)

for x in range(df.pessoas.unique().size):
    df[df.pessoas.unique()[x]] = novas_colunas

for x in range(df.portas.unique().size):
    df[df.portas.unique()[x]] = novas_colunas
```

```
for index, row in df.iterrows():
    df[row['portas']][index] = 1
    df[row['pessoas']][index] = 1

df = df.drop(columns=['portas', 'pessoas'])

classe = df.pop('class')
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho

UFAPE

```
df['class'] = classe

df
```

```
df
```

✓ 0.4s

	comprar	manutencao	porta_malas	seguranca	2_pessoas	4_pessoas	mais_de_4_pessoas	2_portas	3_portas	4_portas	mais_de_4_portas	class
0	3	3	0	0	1	0	0	1	0	0	0	unacc
1	3	3	0	1	1	0	0	1	0	0	0	unacc
2	3	3	0	2	1	0	0	1	0	0	0	unacc
3	3	3	1	0	1	0	0	1	0	0	0	unacc
4	3	3	1	1	1	0	0	1	0	0	0	unacc
...	...	...	...	...	...	...	...	...	...	...	...	...
1723	0	0	1	1	0	0	1	0	0	0	1	good
1724	0	0	1	2	0	0	1	0	0	0	1	vgood
1725	0	0	2	0	0	0	1	0	0	0	1	unacc
1726	0	0	2	1	0	0	1	0	0	0	1	good
1727	0	0	2	2	0	0	1	0	0	0	1	vgood

1728 rows × 12 columns

4. A base Heart Disease (hungarian) possui alguns valores de atributos omissos. Realize o experimento descrito abaixo utilizando o classificador 1-NN. Divida a base em treino (90%) e teste (10%) de forma estratificada. Calcule o intervalo de confiança para a taxa de acerto do classificador utilizando 100 repetições deste experimento. https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.hungarian.data
https://archive.ics.uci.edu/ml/datasets/Heart+Disease

(a) (10 pontos) Preencha os valores omissos no conjunto de treino.

(b) (10 pontos) Preencha os valores omissos no conjunto de teste utilizando o método e os valores definidos para o conjunto de treino.

5. Utilizando a base de dados Wine https://archive.ics.uci.edu/ml/datasets/wine, para cada um dos casos abaixo, realize 100 repetições de Holdout 50/50 e calcule o intervalo de confiança da acurácia utilizando o classificador 1-NN com distância Euclidiana. Realize testes de hipótese por sobreposição dos intervalos de confiança comparando os pré-processamentos de cada um dos casos abaixo com a base de dados original:

(a) (10 pontos) Com todas as características ajustadas para o intervalo [0,1].

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho


```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.neighbors import KNeighborsClassifier
pd.options.mode.chained_assignment = None
```

```
nomes_col = ['Class',
             'Alcohol',
             'Malic_acid',
             'Ash',
             'Alcalinity_of_ash',
             'Magnesium',
             'Total_phenols',
             'Flavanoids',
             'Nonflavanoid_phenols',
             'Proanthocyanins',
             'Color_intensity',
             'Hue',
             'AOD280_OD315_of_diluted_wines',
             'Proline',
            ]
```

```
df = pd.read_csv("data/wine.data", names=nomes_col)


df_zero_um = df.copy()
```

```
def divide_por_dezmil(coluna):
    for x in range(coluna.size):
        coluna[x] = coluna[x]/10000
    return coluna

def divide_por_mil(coluna):
    for x in range(coluna.size):
        coluna[x] = coluna[x]/1000
    return coluna
```

BCC | Reconhecimento de Padrões
Quinta Atividade
Professor Tiago B.A de Carvalho


```
def divide_por_cem(coluna):
    for x in range(coluna.size):
        coluna[x] = coluna[x]/100
    return coluna

def divide_por_dez(coluna):
    for x in range(coluna.size):
        coluna[x] = coluna[x]/10
    return coluna
```

```
df_zero_um.Proline = divide_por_dezmil(df_zero_um.Proline)
df_zero_um.Magnesium = divide_por_mil(df_zero_um.Magnesium)
df_zero_um.Alcalinity_of_ash =
divide_por_cem(df_zero_um.Alcalinity_of_ash)
df_zero_um.Alcohol = divide_por_cem(df_zero_um.Alcohol)
df_zero_um.Malic_acid = divide_por_dez(df_zero_um.Malic_acid)
df_zero_um.Ash = divide_por_dez(df_zero_um.Ash)
df_zero_um.Total_phenols = divide_por_dez(df_zero_um.Total_phenols)
df_zero_um.Flavanoids = divide_por_dez(df_zero_um.Flavanoids)
df_zero_um.Proanthocyanins = divide_por_dez(df_zero_um.Proanthocyanins)
df_zero_um.Color_intensity = divide_por_dez(df_zero_um.Color_intensity)
df_zero_um.Hue = divide_por_dez(df_zero_um.Hue)
df_zero_um.AOD280_OD315_of_diluted_wines=
divide_por_dez(df_zero_um.AOD280_OD315_of_diluted_wines)
```

```
y_df = df.pop("Class")

y_zero_um = df_zero_um.pop("Class")
```

```
def calcular_taxas_para_knn(k, qtd_taxas, porcent_test, atributos,
classe):
    array_taxas = []

    for n in range(k):
        arrays = []
        for i in range(qtd_taxas):
            x_treino, x_teste, y_treino, y_teste =
train_test_split(atributos, classe,test_size=porcent_test,
random_state=i)
```


```
            classificador = KNeighborsClassifier(n_neighbors=n+1)
            classificador.fit(x_treino, y_treino)
            taxas = classificador.score(x_teste, y_teste)
            arrays.append(taxas)

        array_taxas.append(arrays)

    return array_taxas

array_taxas_df = calcular_taxas_para_knn(1, 100, 0.5, df, y_df)

array_taxas_zero_um = calcular_taxas_para_knn(1, 100, 0.5, df_zero_um,
y_zero_um)
```

```
def calcular_intervalo_confianca(hash_medias, hash_diferencas_total):
    for i in range(len(hash_medias)):
        for x in range(len(hash_medias[i+1])):

            desvio_padrao = np.std(hash_diferencas_total[i+1][x])

            erro_padrao =
desvio_padrao/np.sqrt(len(hash_diferencas_total[i+1][x]))

            multiplicador = abs(stats.distributions.norm.ppf(0.025))

            print(i+1, (hash_medias[i+1][x] -
multiplicador*erro_padrao, hash_medias[i+1][x] +
multiplicador*erro_padrao)
            )
```

```
def calcular_medias_separadas(array):

    hash_saida = {}
    array_aux = []
    for x in range(len(array)):
        array_aux.append(np.mean(array[x]))
        hash_saida[x+1] = array_aux
        array_aux = []
    return hash_saida
```

UFAPE

```
medias_separadas = calcular_medias_separadas(array_taxas_df)

medias_separadas_zero_um =
calcular_medias_separadas(array_taxas_zero_um)
```

```
hash_taxas = {}
array_aux = []
for x in range(len(array_taxas_df)):
    array_aux.append(array_taxas_df[x])
    hash_taxas[x+1] = array_aux
    array_aux = []

hash_taxas2 = {}
array_aux2 = []
for x in range(len(array_taxas_zero_um)):
    array_aux2.append(array_taxas_zero_um[x])
    hash_taxas2[x+1] = array_aux2
    array_aux2 = []

calcular_intervalo_confianca(medias_separadas, hash_taxas)
calcular_intervalo_confianca(medias_separadas_zero_um, hash_taxas2)
```

```
✓ 0.9s
1 (0.7083788816487575, 0.7266772981265237)
1 (0.917809882487535, 0.9278080950405551)
```

Com a comparação acima podemos ver claramente que não há sobreposição, rejeita-se H0,
e concluímos com 95% de confiança que os atributos no intervalo [0,1] tem maior acurácia.

(b) (10 pontos) Com todas as características ajustadas para ter média zero e desvio padrão igual a um.

*** fazer apenas uma transformação e comparar com os dados originais**