

**CENTRO UNIVERSITÁRIO FARIAS BRITO**  
**CIÊNCIA DA COMPUTAÇÃO**

DISCIPLINA: INTELIGÊNCIA ARTIFICIAL  
PROFESSOR: JOSÉ BELO ARAGÃO JÚNIOR

**RELATÓRIO: ALGORITMO KNN**

**EQUIPE:**

Pedro Victor da Silva Ávila - Matrícula:1620237  
Renê Victor Lucas - Matrícula:1611181  
Samuel Benevides Linhares - Matrícula:1721168

## **INTRODUÇÃO:**

O K-nearest neighbors algorithm (KNN) é um método utilizado em reconhecimento de padrões para classificação e regressão. O KNN é um dos algoritmos de aprendizado de máquina mais simples existentes, no qual a função é somente aproximada localmente e toda a computação é adiada até a classificação.

## **DESCRIÇÃO DA ATIVIDADE:**

Este documento pretende estudar a implementação de um classificador baseado no algoritmo KNN, para  $K = 1$ , usando como base valores apresentados no arquivo fornecido chamado Iris Dataset. Iremos verificar o modo como as taxas de acerto variam em função da quantidade de dados usados para treinar o classificador, primeiro os separando na proporção 10/90 (10% para treino e 90% para teste) e, em seguida, de 10 em 10% até 90/10 (90% para treino e 10% para teste). Será feito também o gráfico do erro de classificação em função da porcentagem de vetores de treinamento usados na implementação.

Em cada proporção, serão executadas 30 rodadas de treinamento e teste, com os respectivos cálculos de taxas de acerto (i) média, (ii) mínima, (iii) máxima e (iv) média por classe.

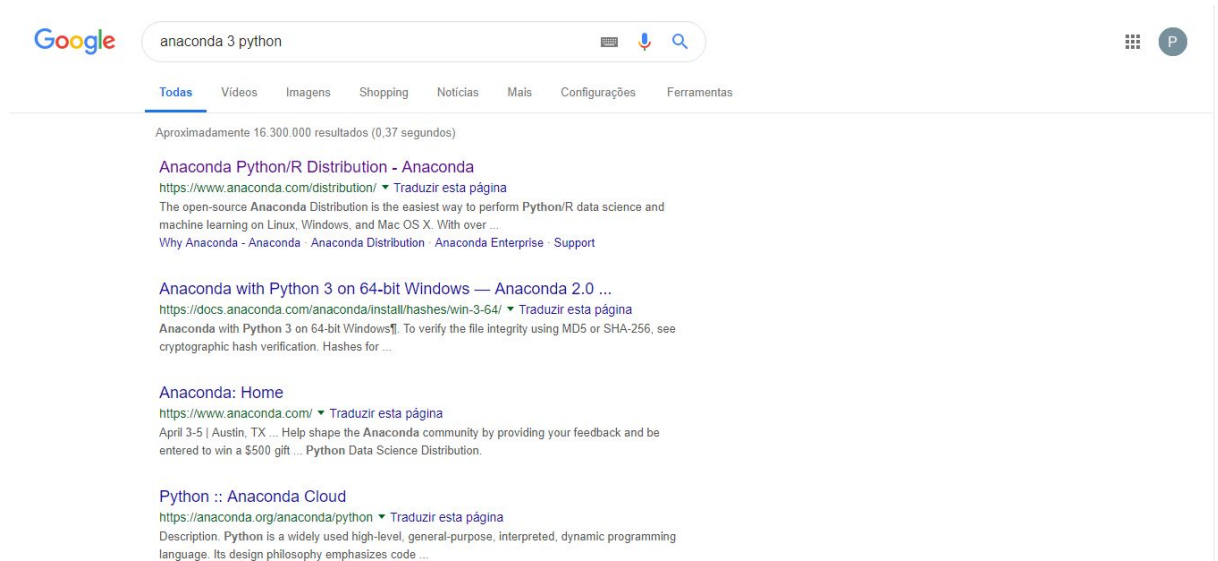
## INSTALAÇÃO:

Para essa etapa de instalação e execução será mostrada a linguagem para desenvolvimento do trabalho, nesse caso, o Python.





***Python*** é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por **Guido van Rossum** em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos **Python Software Foundation**.

Antes de irmos para a etapa de execução do trabalho e começar testar a aplicação desenvolvida, precisamos primeiro instalar o interpretador python. Para isso, abra seu navegador e digite anaconda python na barra de busca dele.



Em seguida clique no link e você será direcionado para a página.

ProductsWhy Anaconda?SolutionsResourcesCompany[Download](#)






# Anaconda Distribution






The World's Most Popular Python/R Data Science Platform






[Download](#)

The open-source [Anaconda Distribution](#) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas,







Clique na opção de “download”.

Desça e baixe a versão do anaconda python referente a seu sistema operacional.

## Anaconda 2018.12 for macOS Installer

### Python 3.7 version

Download

64-Bit Graphical Installer (652.7 MB)  
64-Bit Command Line Installer (557 MB)

### Python 2.7 version

Download

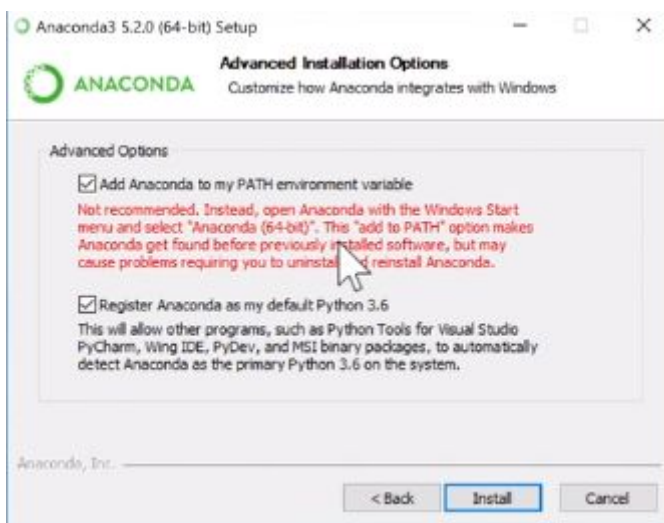
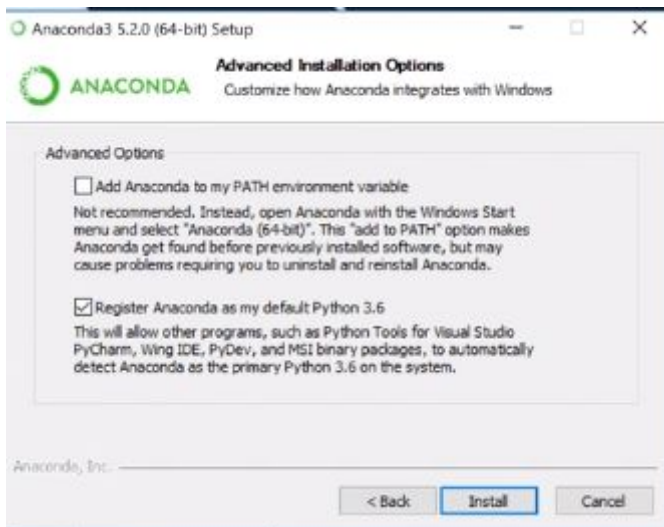
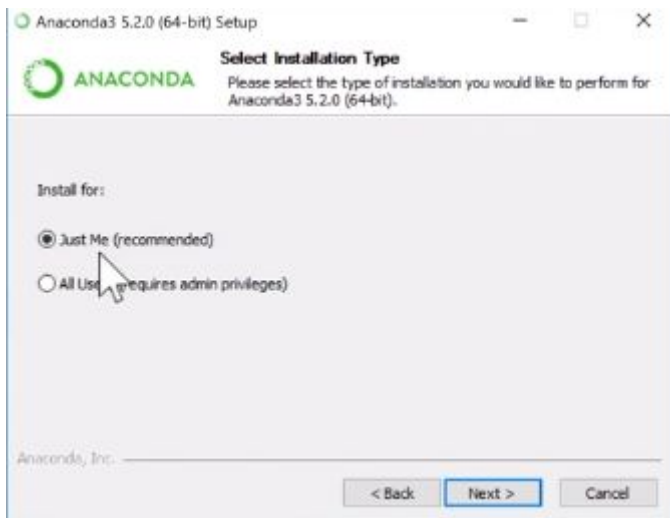
64-Bit Graphical Installer (640.7 MB)  
64-Bit Command Line Installer (547 MB)

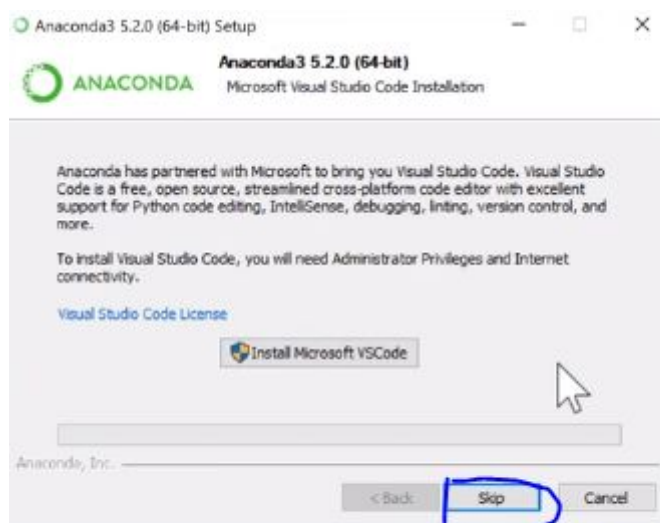
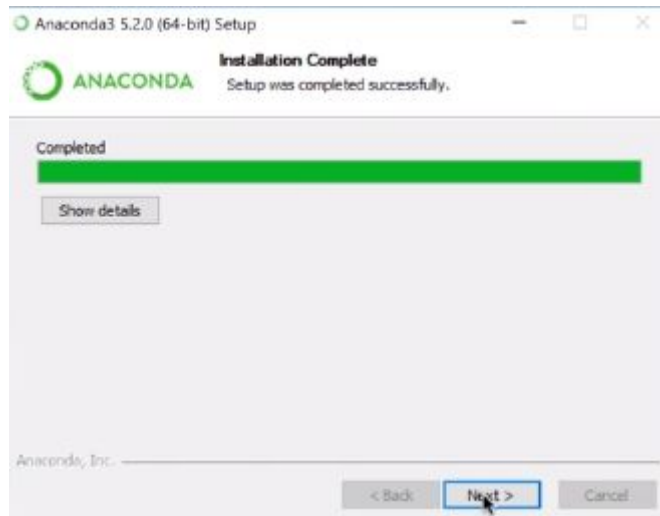
## Get Started with Anaconda Distribution

Name	Type	Size
○ Anaconda3-5.2.0-Windows-x86_64.exe	Application	646,472 KB

Type: Application  
Size: 631 MB  
Date modified: 2019-03-05 2:05 PM







CLIQUE



## ABORDAGEM UTILIZADA:

Em um primeiro momento, faremos a implementação do algoritmo sem a identificação de outliers.

```
import math;
import pandas as pd;
import matplotlib.pyplot as plt;
import random;

dataset=[a.split("\t") for a in open("Iris dataset.dat","r",encoding="utf-8").readlines()]
dataset=[[float(x) for x in a] for a in dataset]
tamanho=len(dataset)
tamMud=tamanho*0.1
tamTeste=tamanho*0.9;
tamTreinamento=tamanho*0.1
AcertoClassif={"Treinamento/Teste":[],"TxMin":[],"TxMedia":[],"TxMax":[],"TxClasse1":[],"TxClasse2":[],"TxClasse3":[]};
#dicionário para armazenar os valores,taxas que serao usados em grafico e etc
dist={}

for a in range(0,tamanho):
    for b in range(a+1,tamanho):

dist[a,b]=math.sqrt(pow((dataset[b][0]-dataset[a][0]),2)+pow((dataset[b][1]-dataset[a][1]),2)+pow((dataset[b][2]-dataset[a][2]),2)+pow((dataset[b][3]-dataset[a][3]),2))

def pegarDist(id1,id2):
    ids=[id1,id2];
    ids.sort();

    return dist[ids[0],ids[1]]

def classificar(teste, treinamento):
    distancias=[pegarDist(treinamento,a) for a in teste]
    indice=distancias.index(min(distancias));
    return dataset[(teste[indice])][4];

"""for a in range(1,50):
    print(dist[0,a]);

print(classificar(list(range(1,50)),0))
print(dist[0,17])"""

#loop que continuará enquanto as proporcoes nao forem 135/15(treinamento/teste)
while(tamTreinamento<=0.9*tamanho and tamTeste>=0.1*tamanho):
```



```

print("Calculando para proporcao %r /
%r"%((tamTreinamento*100/tamanho),(tamTeste*100/tamanho)));

acertosclasse1=0;#variavel aux para contar os acertos da classe 1
acertosclasse2=0;#variavel aux para contar os acertos da classe 2
acertosclasse3=0;#variavel aux para contar os acertos da classe 3
acertototal=0;#variavel aux para contar acerto total
minAcerto=10000;#variavel que guardara o valor min de acerto para cada proporcao. E
atualizada apos cada rodada
maxAcerto=0;#variavel para armazenar a quantidade maxima de acerto para cada
proporcao. E atualizada apos cada rodada

for x in range(0,30):
    somaAcerto=0;#variavel temp que somara o numero de acertos de cada rodada, e ao
final de cada rodada, e utilizada
    #pelo minAcerto, maxAcerto e acertototal
    #variavel para armazenar ids do dataset embaralhado
    datEmb=random.sample(list(range(0,tamanho)),tamanho);
    #dividindo dataset para treino e teste
    datTest=datEmb[:int(tamTeste)];
    datTrein=datEmb[int(tamTeste):];

    #print(classificar(dataset[0:948],dataset.values[948]));
    #for para iterar pelo vetor de treinamento e utilizar o classificador
    #[tamTeste:] cria uma lista que comeca a partir do fim do vetor de teste. depois do
ultimo elemento
    for a in datTrein:
        if(classificar(datTest,a)==dataset[a][4]):
            somaAcerto+=1;#se o resultado da classificacao for igual ao do vetor de
treinamento
            #incrementa a variavel somaAcerto
            if(dataset[a][4]==1):
                acertosclasse1+=1;#incrementa a variavel caso o resultado seja da classe 1
            elif(dataset[a][4]==2):
                acertosclasse2+=1;#incrementa a variavel caso o resultado seja da classe 2
            elif(dataset[a][4]==3):
                acertosclasse3+=1;#incrementa a variavel caso o resultado seja da classe 3

    #verificando se o valor de somaAcerto e maior do que o maior valor atual
    if(somaAcerto>maxAcerto):
        maxAcerto=somaAcerto;#caso seja, o valor de somaAcerto e atribuido a maxAcerto

    #verificando se o valor de somaAcerto e menor do que o menor valor atual
    if(somaAcerto<minAcerto):
        minAcerto=somaAcerto;#caso seja, o valor de somaAcerto e atribuido a minAcerto

```

```
acertototal+=somaAcerto;#incrementando acertotal com valor de somaAcerto
```

```
#adicionando os valores de cada proporcao ao campo correspondente no dicionario  
AcertoClassif
```

```
#ao mesmo tempo que adiciona, calcula os valores das taxas, transforma em  
porcentagem
```

```
AcertoClassif["Treinamento/Teste"].insert(0,str(tamTreinamento*100/tamanho)+" /  
"+str(tamTeste*100/tamanho));
```

```
AcertoClassif["TxMin"].insert(0,minAcerto/tamTreinamento);
```

```
AcertoClassif["TxMedia"].insert(0,acertototal/(tamTreinamento*30));
```

```
AcertoClassif["TxMax"].insert(0,maxAcerto/tamTreinamento);
```

```
AcertoClassif["TxClasse1"].insert(0,acertosclasse1/(acertototal));
```

```
AcertoClassif["TxClasse2"].insert(0,acertosclasse2/(acertototal));
```

```
AcertoClassif["TxClasse3"].insert(0,acertosclasse3/(acertototal));
```

```
tamTeste-=tamMud;#reduzindo em 100 tamanho do vetor de teste
```

```
tamTreinamento+=tamMud;#incrementando em 100 o tamanho do vetor de treinamento
```

```
#criando um dataframe com os valores do dicionário. Isso é só para ficar mais organizado
```

```
#na hora de mostrar os resultados de cada proporcao
```

```
tabelaTx = pd.DataFrame(AcertoClassif,  
columns=["Treinamento/Teste","TxMin","TxMedia","TxMax","TxClasse1","TxClasse2","TxClasse3"]);
```

```
#printando o resultado da classificacao
```

```
print(tabelaTx)
```

```
#plotando gráfico com as taxas de acerto por proporcao
```

```
plt.figure();
```

```
plt.title("Taxas de acerto por proporcao");
```

```
plt.grid();
```

```
plt.plot(AcertoClassif["Treinamento/Teste"][:-1],AcertoClassif["TxMedia"][:-1];
```

```
plt.xlabel("Proporção Treinamento / Teste");
```

```
plt.ylabel("Taxa de acerto");
```

```
plt.show();
```

```
# In[ ]:
```

Em um segundo momento, faremos a implementação do algoritmo com a identificação de outliers:

```
#!/usr/bin/env python
# coding: utf-8
```

```
# In[43]:
```

```
import math;
import pandas as pd;
import matplotlib.pyplot as plt;
import random;
import numpy as np;
```

```
dataset1=[a.split("\t") for a in open("Iris dataset.dat","r",encoding="utf-8").readlines()]
dataset1=[[float(x) for x in a] for a in dataset1]
```

```
def RetOutliers():
```

```
    outliers=[0 for a in range(0,len(dataset1))];#lista para marcar as posições dos dados
    com outliers. 0 =false; 1 true
```

```
    for a in range(0,4):
        dado=[x[a] for x in dataset1]#dado/coluna que será verificada para ver se encontra-se
        outliers
        #print(dado)
        media=np.mean(dado);#media da coluna
        desP=np.std(dado);#desvio padrao da coluna que será usado para identificar outliers
        q1,q3=np.percentile(sorted(dado),[25,75]);#valores para se usar no iqr
        iqr=q3-q1;#para identificar outliers por iqr
```

```
    for b in range(0, len(dataset1)):
        #verificando utilizando o metodo do desvio
        if(dado[b]>media+(2*desP) or dado[b]<media-(2*desP)):
            outliers[b]=1;
        #if(dado[b]>q3+(1.5*iqr) or dado[b]<q1-(1.5*iqr)):
            #outliers[b]=1;
```

```
    novoDataset=[];
    for a in range(0, len(dataset1)):
        if(outliers[a]==0):
            novoDataset.append(dataset1[a]);
```

```
    return novoDataset;
```

```

#novo dataset limpo
dataset=RetOutliers();
#tamanho do novo dataset limpo
tamanho=len(dataset)
#l1=novdat[:int(len(novdat)*0.1)]
#l2=novdat[int(len(novdat)*0.1):]

tamTeste=90;
tamTreinamento=10
AcertoClassif={"Treinamento/Teste":[], "TxMin":[], "TxMedia":[], "TxMax":[], "TxClasse1":[], "TxClasse2":[], "TxClasse3":[]};
#dicionário para armazenar os valores,taxas que serão usados em gráfico e etc
dist={}
#calcular distancias
for a in range(0,tamanho):
    for b in range(a+1,tamanho):

dist[a,b]=math.sqrt(pow((dataset[b][0]-dataset[a][0]),2)+pow((dataset[b][1]-dataset[a][1]),2)+pow((dataset[b][2]-dataset[a][2]),2)+pow((dataset[b][3]-dataset[a][3]),2))

#função para buscar distancia no dicionario
def pegarDist(id1,id2):
    ids=[id1,id2];
    ids.sort();

    return dist[ids[0],ids[1]]

#função para classificar
def classificar(teste, treinamento):
    distancias=[pegarDist(treinamento,a) for a in teste]
    indice=distancias.index(min(distancias));
    return dataset[(teste[indice])][4];

"""for a in range(1,50):
    print(dist[0,a]);

print(classificar(list(range(1,50)),0))
print(dist[0,17])"""

#loop que continuar enquanto as proporções não forem 135/15(treinamento/teste)
while(tamTreinamento<=90 and tamTeste>=10):
    print("Calculando para proporção %r / %r"%(int(tamTreinamento),int(tamTeste)));

```

```

acertosclasse1=0;#variavel aux para contar os acertos da classe 1
acertosclasse2=0;#variavel aux para contar os acertos da classe 2
acertosclasse3=0;#variavel aux para contar os acertos da classe 3
acertototal=0;#variavel aux para contar acerto total
minAcerto=10000;#variavel que guardarÃ o valor min de acerto para cada proporÃ£o.
Ã© atualizada apÃs cada rodada
maxAcerto=0;#variavel para armazenar a quantidade mÃxima de acerto para cada
proporÃ£o. Ã© atualizada apÃs cada rodada

```

```

for x in range(0,30):
    somaAcerto=0;#variavel temp que somarÃ o nÃmero de acertos de cada rodada, e
    ao final de cada rodada, Ã utilizada
    #pelo minAcerto, maxAcerto e acertototal
    #variavel para armazenar ids do dataset embaralhado
    datEmb=random.sample(list(range(0,tamanho)),tamanho);
    #dividindo dataset para treino e teste
    datTest=datEmb[int((tamTeste*tamanho)/100)];
    datTrein=datEmb[int((tamTeste*tamanho)/100):];

    tTeste=len(datTest);
    tTrein=len(datTrein);

```

```

#print(classificar(dataset[0:948],dataset.values[948]));
#for para iterar pelo vetor de treinamento e utilizar o classificador
#[tamTeste:] cria uma lista que comeÃa a partir do fim do vetor de teste. depois do
ultimo elemento

```

```

for a in datTrein:
    if(classificar(datTest,a)==dataset[a][4]):
        somaAcerto+=1;#se o resultado da classificaÃo for igual ao do vetor de
        treinamento
        #incrementa a variavel somaAcerto
        if(dataset[a][4]==1):
            acertosclasse1+=1;#incrementa a variavel caso o resultado seja da classe 1
        elif(dataset[a][4]==2):
            acertosclasse2+=1;#incrementa a variavel caso o resultado seja da classe 2
        elif(dataset[a][4]==3):
            acertosclasse3+=1;#incrementa a variavel caso o resultado seja da classe 3

```

```

#verificando se o valor de somaAcerto Ã maior do que o maior valor atual
if(somaAcerto>maxAcerto):
    maxAcerto=somaAcerto;#caso seja, o valor de somaAcerto Ã atribuido a
    maxAcerto

```

```

#verificando se o valor de somaAcerto Ã menor do que o menor valor atual
if(somaAcerto<minAcerto):

```

```
minAcerto=somaAcerto;#caso seja, o valor de somaAcerto Ã© atribuido a minAcerto
```

```
acertototal+=somaAcerto;#incrementando acertotal com valor de somaAcerto
```

```
#adicionando os valores de cada proporÃ§Ã£o ao campo correspondente no dicionÃ¡rio  
AcertoClassif
```

```
#ao mesmo tempo que adiciona, calcula os valores das taxas, transforma em  
porcentagem
```

```
AcertoClassif["Treinamento/Teste"].insert(0,str(int(tamTreinamento))+ " /  
"+str(int(tamTeste)));
```

```
AcertoClassif["TxMin"].insert(0,minAcerto/tTrein);
```

```
AcertoClassif["TxMedia"].insert(0,acertototal/(tTrein*30));
```

```
AcertoClassif["TxMax"].insert(0,maxAcerto/tTrein);
```

```
AcertoClassif["TxClasse1"].insert(0,acertosclasse1/(acertototal));
```

```
AcertoClassif["TxClasse2"].insert(0,acertosclasse2/(acertototal));
```

```
AcertoClassif["TxClasse3"].insert(0,acertosclasse3/(acertototal));
```

```
tamTeste-=10;#reduzindo em 100 tamanho do vetor de teste
```

```
tamTreinamento+=10;#incrementando em 100 o tamanho do vetor de treinamento
```

```
#criando um dataframe com os valores do dicionÃ¡rio. Isso Ã© sÃ³ para ficar mais  
organizado
```

```
#na hora de mostrar os resultados de cada proporÃ§Ã£o
```

```
tabelaTx = pd.DataFrame(AcertoClassif,  
columns=["Treinamento/Teste","TxMin","TxMedia","TxMax","TxClasse1","TxClasse2","TxClasse3"]);
```

```
#printando o resultado da classificaÃ§Ã£o
```

```
print(tabelaTx)
```

```
#plotando grÃ¡fico com as taxas de acerto por proporÃ§Ã£o
```

```
plt.figure();
```

```
plt.title("Taxas de acerto por proporÃ§Ã£o");
```

```
plt.grid();
```

```
plt.plot(AcertoClassif["Treinamento/Teste"][:-1],AcertoClassif["TxMedia"][:-1];
```

```
plt.xlabel("ProporÃ§Ã£o Treinamento / Teste");
```

```
plt.ylabel("Taxa de acerto");
```

```
plt.show();
```

```
# In[ ]:
```

```
# In[ ]:
```

## RESULTADOS:

Para os testes, foi utilizado no Jupyter Notebook:

The image shows a Jupyter Notebook interface. The top part displays a file browser with the following files:

Name	Last Modified	File size
KNN-CopyComRetOut.ipynb	uma hora atrás	32.5 kB
KNN.ipynb	2 dias atrás	30.3 kB
Iris dataset.dat	11 dias atrás	2.85 kB
KNN-CopyComRetOut.py	uma hora atrás	6.72 kB
KNN.py	2 dias atrás	5.39 kB

The bottom part shows a code cell with the following Python code:

```
return dist[ids[0],ids[1]]

#função para classificar
def classificar(teste, treinamento):
    distancias=[pegarDist(treinamento,a) for a in teste]
    indice=distancias.index(min(distancias));
    return dataset[(teste[indice])][4];

"""for a in range(1,50):
    print(dist[0,a]);

print(classificar(list(range(1,50)),0))
print(dist[0,17])"""

#Loop que continuará enquanto as proporções não forem 135/15(treinamento/teste)
while(tamTreinamento<=90 and tamTeste>=10):
    print("Calculando para proporção %r / %r"%(int(tamTreinamento),int(tamTeste)));

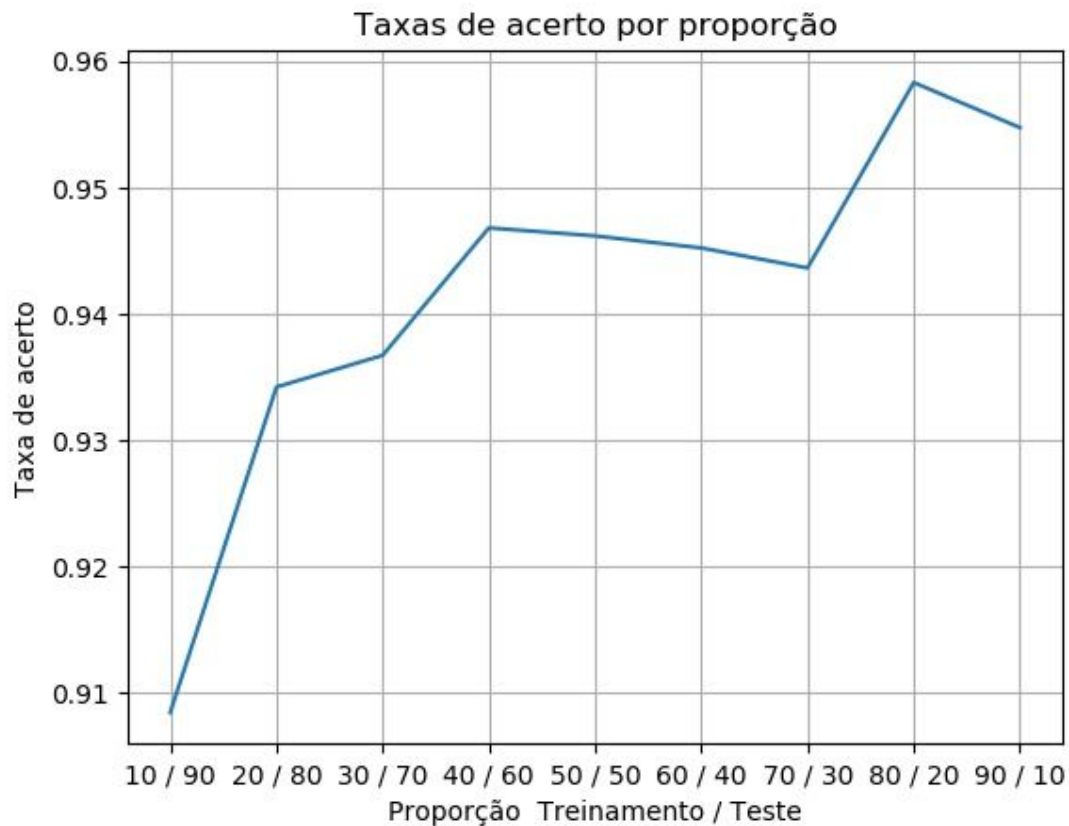
    acertosclasse0=0;#variavel aux para contar os acertos da classe 0
    acertosclasse1=0;#variavel aux para contar os acertos da classe 1
    acertosclasse2=0;#variavel aux para contar os acertos da classe 2
    acertosclasse3=0;#variavel aux para contar os acertos da classe 3
    acertototal=0;#variavel aux para contar acerto total
    minAcerto=10000;#variavel que guardará o valor min de acerto para cada proporção. é atualizada após cada rodada
    maxAcerto=0;#variavel para armazenar a quantidade máxma de acerto para cada proporção. é atualizada após cada rodada
```

A large blue watermark "SHIFT+ENTER" is overlaid on the code.

Para o algoritmo sem identificação de outliers:

```
C:\Users\pedru\Downloads\Inteligência artificial\Trabalho KNN>python KNN.py
Calculando para proporção 10.0 / 90.0
Calculando para proporção 20.0 / 80.0
Calculando para proporção 30.0 / 70.0
Calculando para proporção 40.0 / 60.0
Calculando para proporção 50.0 / 50.0
Calculando para proporção 60.0 / 40.0
Calculando para proporção 70.0 / 30.0
Calculando para proporção 80.0 / 20.0
Calculando para proporção 90.0 / 10.0
```

Treinamento/Teste	TxMin	TxMedia	TxMax	TxClasse1	TxClasse2	TxClasse3
90.0 / 10.0	0.629630	0.906420	0.977778	0.367475	0.321983	0.310542
80.0 / 20.0	0.883333	0.944167	0.983333	0.347749	0.321565	0.330685
70.0 / 30.0	0.923810	0.942222	0.971429	0.353774	0.323787	0.322439
60.0 / 40.0	0.933333	0.950741	0.977778	0.347098	0.327620	0.325282
50.0 / 50.0	0.906667	0.951111	0.986667	0.347196	0.337383	0.315421
40.0 / 60.0	0.916667	0.956667	1.000000	0.333333	0.344948	0.321719
30.0 / 70.0	0.888889	0.952593	1.000000	0.349145	0.351477	0.299378
20.0 / 80.0	0.900000	0.961111	1.000000	0.350289	0.329480	0.320231
10.0 / 90.0	0.800000	0.962222	1.000000	0.330254	0.337182	0.332564



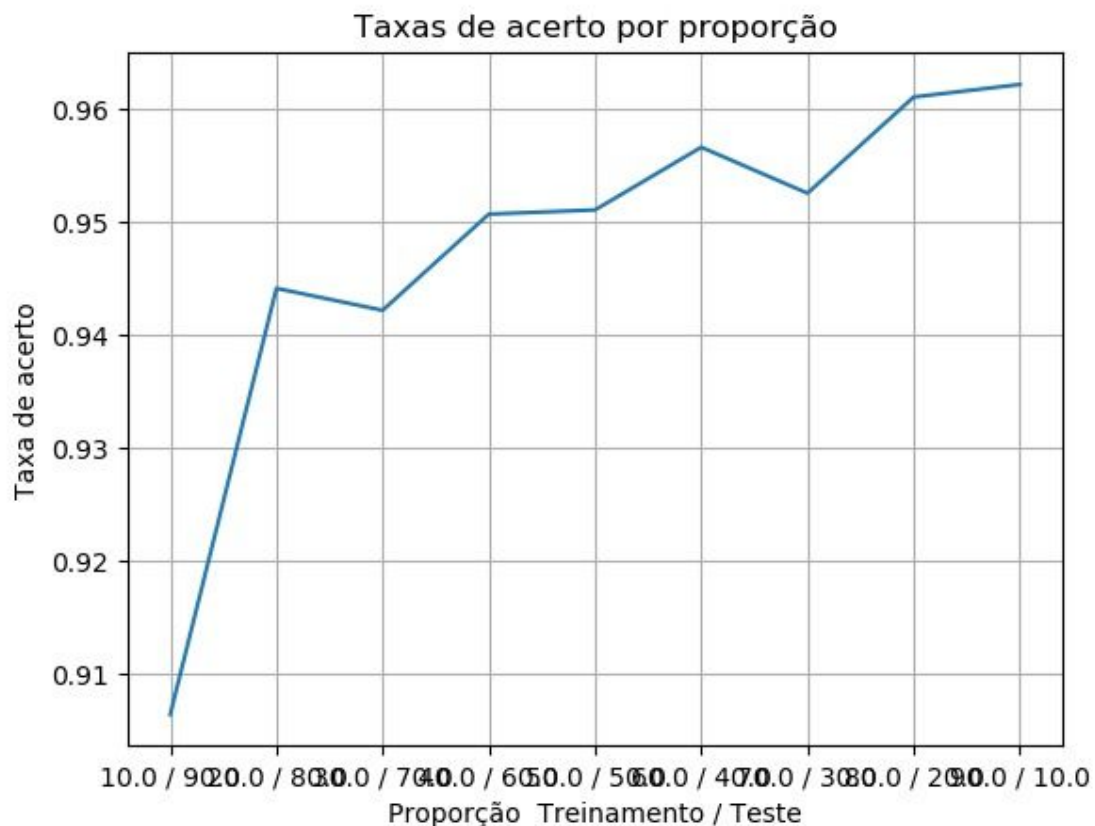


E para o algoritmo com identificação de outliers:

```
Microsoft Windows [versão 10.0.17763.475]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.

C:\Users\pedru\Downloads\Inteligência artificial\Trabalho KNN>python "KNN-CopyComRetOut.py"
Calculando para proporção 10 / 90
Calculando para proporção 20 / 80
Calculando para proporção 30 / 70
Calculando para proporção 40 / 60
Calculando para proporção 50 / 50
Calculando para proporção 60 / 40
Calculando para proporção 70 / 30
Calculando para proporção 80 / 20
Calculando para proporção 90 / 10
Treinamento/Teste TxMin TxMedia TxMax TxClasse1 TxClasse2 TxClasse3
90 / 10 0.587302 0.908466 0.976190 0.351776 0.343914 0.304310
80 / 20 0.875000 0.934226 0.982143 0.358076 0.342784 0.299140
70 / 30 0.897959 0.936735 0.959184 0.350036 0.348947 0.301017
60 / 40 0.916667 0.946825 0.964286 0.350796 0.337804 0.311400
50 / 50 0.885714 0.946190 0.985714 0.348264 0.347760 0.303976
40 / 60 0.910714 0.945238 0.982143 0.345088 0.334383 0.320529
30 / 70 0.880952 0.943651 1.000000 0.344828 0.359966 0.295206
20 / 80 0.857143 0.958333 1.000000 0.361491 0.366460 0.272050
10 / 90 0.857143 0.954762 1.000000 0.356608 0.324190 0.319202

C:\Users\pedru\Downloads\Inteligência artificial\Trabalho KNN>python KNN.py
```



Deve ser observado que a presença de outliers nos resultados finais pode influenciar a precisão do algoritmo e, portanto, problemas com inclusão de outliers devem receber atenção especial em sua interpretação.

## **CONCLUSÃO:**

Através deste trabalho foi possível estudar a forma como o algoritmo KNN é implementado e elucidar seu funcionamento de forma prática. Também foi possível estudá-lo na presença e na ausência de outliers nos dados do problema abordado; dessa forma, fica visível a influência de dados discrepantes nos resultados finais de uma implementação.