Alunos:
Felipe Zarattini Miranda / 1220557
Pedro de Andrade Marcial Gomes / 1221273

Introdução

Neste trabalho desenvolvemos dois métodos de aprendizado de máquinas, um supervisionado e um não supervisionado, para classificar elementos de uma base de dados escolhida por nós através do site: http://archive.ics.uci.edu/ml/index.php

Métodos Utilizados

Knn

O método supervisionado que implementamos foi o K-nearest neighbors, que classifica a amostra desconhecida através de um sistema de votos.

O método irá calcular a distância euclidiana entre os elementos já definidos e verificará qual as classes dos K elementos com valores mais próximos.

Então cada “vizinho” faz um voto, dizendo a que classe ele pertence. A classe que tiver mais votos será escolhida para classificar a amostra desconhecida.

Utilizamos a linguagem C++ para implementar o método, visto que ela nos trás facilidades para trabalhar com a criação de classes e a leitura de dados de um arquivo, assim como exportar a saída para arquivos separados (Redirecionamento do stdout é facilmente implementado).

- Para utilizar o programa:
- Abra a prompt de comando;
- Vá até o diretório onde o knn.exe e dataset.txt estão localizados;
- Digite: “knn.exe x y < database.txt” sem aspas, onde x é o número de vizinhos e y é o tamanho da amostra de teste.

O código foi feito da seguinte maneira:

Temos uma classe “Flor” que comporta os 4 parâmetros e o nome do tipo de Iris. Para pegarmos estes parâmetros e nome da flor, temos os seguinte getters: getArg1(), getArg2(), getArg3(), getArg4() e getClasse().

Temos uma função “distancia” que calcula a distância euclidiana entre cada flor. Para isso fazemos o cálculo:

```
double soma = pow((flor1.getArg1() - flor2.getArg1()), 2) + pow((flor1.getArg2() - flor2.getArg2()), 2) + pow((flor1.getArg3() - flor2.getArg3()), 2) + pow((flor1.getArg4() - flor2.getArg4()), 2);

return sqrt(soma);
```

Temos também a função “classifica” que classifica a nova amostra desconhecida.
Para isso, ela recebe um vetor de objetos Flor, uma amostra desconhecida da classe Flor e um valor inteiro k, que é a quantidade de vizinhos.

Então ela cria pares ordenados compostos da distância dos elementos do conjunto de treinamento até a amostra desconhecida e o índice desse desse elemento no vetor de flores.

Assim a amostra desconhecida é classificada ao percorrermos o vetor de distâncias de seus vizinhos, recebendo os votos dos K vizinhos mais próximo. A classe que possuir mais votos será atribuída a amostra desconhecida.

A main do programa verifica que o K digitado foi um número ímpar (impedindo a execução do programa caso contrário), cria o conjunto de treinamento e então começa seu processo de classificação, calculando o tempo de execução de cada estrutura de repetição.

Criando um objeto para a amostra desconhecida e chamando a função “classifica” para determinar qual sua classe.

No prompt de comando então é exibido qual classe foi obtida e qual era o resultado esperado. Assim como a porcentagem de classificações corretas.

Kmeans

O método não supervisionado que implementamos foi o K-means, cujo objetivo é determinar similaridade entre os pontos e organizar as classes da base de dados em clusters (grupos) a partir da distância de seus atributos em relação ao centro dos clusters.

O método consiste em determinar um centróide de forma aleatória para cada cluster, e então associar cada ponto da base de dados ao grupo com centróide mais próximo (utilizando distância euclidiana), e então por fim recalcular a posição destes centróides.

Utilizamos a linguagem C++ para implementar o método, visto que ela nos trás facilidades para trabalhar com a criação de classes e a leitura de dados de um arquivo, assim como exportar a saída para arquivos separados (Redirecionamento do stdout é facilmente implementado).

- Para utilizar o programa:
- Abra a prompt de comando;

- Vá até o diretório onde o kmeans.exe e dataset.txt estão localizados;
- Digite: “kmeans.exe x < database.txt” sem aspas, onde x é o número de clusters em que se deseja dividir a base de dados.

O código foi feito da seguinte maneira:

Possuímos inicialmente 3 classes:

Classe Ponto: Caracterizada por idPonto, idCluster, nome e um vetor de atributos que serão lidos do arquivo de entrada. Esta classe ainda conta com 5 getters para que estes parâmetros sejam recuperados.

- getID
- getNome
- getAtributo
- getIdCluster
- getQtdAtributos

e um setter para que possamos determinar o cluster ao qual um determinado ponto está atrelado.

- setIdCluster

Classe Cluster: É composta de um inteiro idCluster, um vetor de centróides e um vetor de pontos que representam os pontos que fazem parte deste determinado cluster. Esta classe possui 4 getters.

- getID
- getPonto
- getQtdPontos
- getCentroide

um setter.

- setCentroide

e dois métodos que utilizamos para inserir e remover um ponto do cluster

- inserePonto
- removePonto

Classe Kmeans: É construída utilizando valores lidos tanto da linha de comando (K) e da primeira linha do arquivo da base de dados (qtdPontos, qtdAtributos e maxIter). Possui também um getter para o qual é passado um ponto, e retorna o id do cluster cujo centróide é o mais próximo deste ponto.

- getIdMaisPerto

Esta classe também é composta pelos dois métodos principais do programa:

- processa: Este método busca pontos aleatórios na base de dados e utiliza seus atributos para gerar K centróides aleatórios. Após gerados, cada ponto da base é atrelado ao centróide mais próximo de si utilizando distância euclidiana, ou seja, o idCluster de cada ponto é determinado. Após todos os pontos terem seus clusters definidos, o centro de cada um dos clusters é recalculado utilizando os pontos que

estão presentes nele. Por fim, cada cluster com seus pontos é impresso em um arquivo de saída “clusters.txt”.

- verifica: Este método é responsável por verificar a porcentagem de pontos errados presentes em cada cluster. Ele funciona lendo o arquivo de saída onde estão presentes a relação entre os clusters e os pontos e através de contadores, verifica qual classe está predominante em cada um dos clusters. As demais classes presentes neste mesmo cluster são tratadas como erro e então a porcentagem é calculada com base na quantidade total de pontos naquele cluster e nesta quantidade de pontos incorretos presentes. O resultado é impresso no arquivo “verificacao.txt”.

Experimentos

Para realizarmos os experimentos e testes com os métodos escolhidos, utilizamos a database Íris, composta por 3 classes diferentes de flores.

O arquivo é organizado em 4 parâmetros, cada um deles significando respectivamente a distância da sépala em comprimento, distância sépala em largura, distância pétala em comprimento e distância pétala em largura. Por fim, na mesma linha há a qual classe o elemento pertence: Iris Setosa, Iris Versicolour ou Iris Virginica.

Descrição da modelagem dos exemplos de treinamento

Atributos selecionados para descrever os exemplos;

Justificativa para a escolha dos atributos;

Os atributos selecionados para descrever os exemplos foram os 4 atributos presentes no dataset utilizado. Decidimos permanecer com esse número para evitar que a pequena quantidade de atributos influenciasse na corretude dos resultados.

Descrição dos experimentos realizados:

Experimentos em KNN

Para descobrirmos quando o algoritmo possui mais eficiência, testamos diversos valores de vizinhos e tamanhos de conjunto de treinamento. Fomos gradativamente aumentando os valores e fazendo combinações entre muitos/poucos vizinhos com muito/poucos exemplos de treinamento.

Alguns resultados marcantes foram:

k = 1
Exemplos de treinamento = 1
Porcentagem de classificações corretas = 32%

k = 1
Exemplos de treinamento = 50
Porcentagem de classificações corretas = 95%

k = 1
Exemplos de treinamento = 140
Porcentagem de classificações corretas = 80%

k = 3
Exemplos de treinamento = 50
Porcentagem de classificações corretas = 95%

k = 3
Exemplos de treinamento = 100
Porcentagem de classificações corretas = 98%

k = 7
Exemplos de treinamento = 50
Porcentagem de classificações corretas = 98%

k = 7
Exemplos de treinamento = 140
Porcentagem de classificações corretas = 80%

k = 7
Exemplos de treinamento = 50
Porcentagem de classificações corretas = 97%

k = 9
Exemplos de treinamento = 140
Porcentagem de classificações corretas = 90%

k = 11
Exemplos de treinamento = 50
Porcentagem de classificações corretas = 98%

k = 11
Exemplos de treinamento = 100
Porcentagem de classificações corretas = 100%

k = 51
Exemplos de treinamento = 60
Porcentagem de classificações corretas = 63%

k = 51
Exemplos de treinamento = 100
Porcentagem de classificações corretas = 94%

k = 91
Exemplos de treinamento = 100
Porcentagem de classificações corretas = 64%

k = 101
Exemplos de treinamento = 110
Porcentagem de classificações corretas = 62%

Por estes testes podemos perceber que valores muito pequenos ou muito grandes de k diminuem a porcentagem de classificações corretas.

Quando temos um k muito pequeno, permitimos que qualquer ruído que possa ter ocorrido durante a classificação dos exemplos de treinamento, como um elemento com parâmetros muito destoantes para sua classe, afete a classificação da amostra desconhecida.

Já com k muito grande permitimos que a classificação da amostra desconhecida seja muito influenciada pela classe predominante dentro dos exemplos de treinamento. Como neste banco de dados em específico a distribuição de cada classe de flor é proporcional, isso não se mostrou um problema tão grande.

Pelos testes podemos concluir que para esse dataset o valor K = 11 possui alta eficiência quando possuímos 50 ou mais Exemplos de Treinamento.

Com relação a quantidade dos Exemplos de Treinamento, valores abaixo de 50 mostraram uma queda na porcentagem de classificações corretas.

Uma quantidade pequena faz com que haja grande disparidade nos valores de cada parâmetro dentro de uma mesma classe, o que aumenta a incidência de erros na hora da classificação da amostra desconhecida.

Por outro lado uma grande quantidade de exemplos de treinamento faz com que o algoritmo seja capaz de reconhecer melhor as características do elemento desconhecido, aumentando assim a chance de acertar a classificação de uma nova amostra.

Mas em alguns testes feitos com muitos exemplos de treinamento, como 140, podemos notar que a porcentagem de acertos caiu. Isso se deve ao fato de que em testes como esses haviam poucas amostras desconhecidas, 10 desconhecidas no caso de haver 140 exemplos de teste, fazendo com que um erro altere demasiadamente na porcentagem de acertos.

Experimentos em Kmeans

O algoritmo Kmeans implementado recebe um valor de K diretamente da linha de comando junto com o arquivo de base de dados que deve ler. Este arquivo, porém, possui na primeira linha valores que indicam a quantidade de pontos na base, quantos atributos cada um dos pontos possui, a quantidade máxima de iterações em que estes pontos devem ser processados e um valor booleano (1 ou 0) que indica se o ponto já possui uma classificação.

Para o primeiro teste colocaremos a imagem da saída dos pontos agrupados em clusters. Como estas saídas, porém, são muito extensas, nos demais testes iremos colocar apenas a porcentagem de pontos classificados erroneamente em cada cluster.

Resultados encontrados variando o valor de K e com quantidade máxima de iterações 100:

K = 3

```
Cluster 1
Ponto 1: 5.1 3.5 1.4 0.2 - Iris-setosa
Ponto 2: 4.9 3 1.4 0.2 - Iris-setosa
Ponto 3: 4.7 3.2 1.3 0.2 - Iris-setosa
Ponto 4: 4.6 3.1 1.5 0.2 - Iris-setosa
Ponto 5: 5 3.6 1.4 0.2 - Iris-setosa
Ponto 6: 5.4 3.9 1.7 0.4 - Iris-setosa
Ponto 7: 4.6 3.4 1.4 0.3 - Iris-setosa
Ponto 8: 5 3.4 1.5 0.2 - Iris-setosa
Ponto 9: 4.4 2.9 1.4 0.2 - Iris-setosa
Ponto 10: 4.9 3.1 1.5 0.1 - Iris-setosa
Ponto 11: 5.4 3.7 1.5 0.2 - Iris-setosa
Ponto 12: 4.8 3.4 1.6 0.2 - Iris-setosa
Ponto 13: 4.8 3 1.4 0.1 - Iris-setosa
Ponto 14: 4.3 3 1.1 0.1 - Iris-setosa
Ponto 15: 5.8 4 1.2 0.2 - Iris-setosa
Ponto 16: 5.7 4.4 1.5 0.4 - Iris-setosa
Ponto 17: 5.4 3.9 1.3 0.4 - Iris-setosa
Ponto 18: 5.1 3.5 1.4 0.3 - Iris-setosa
Ponto 19: 5.7 3.8 1.7 0.3 - Iris-setosa
Ponto 20: 5.1 3.8 1.5 0.3 - Iris-setosa
Ponto 21: 5.4 3.4 1.7 0.2 - Iris-setosa
Ponto 22: 5.1 3.7 1.5 0.4 - Iris-setosa
Ponto 23: 4.6 3.6 1 0.2 - Iris-setosa
Ponto 24: 5.1 3.3 1.7 0.5 - Iris-setosa
Ponto 25: 4.8 3.4 1.9 0.2 - Iris-setosa
Ponto 26: 5 3 1.6 0.2 - Iris-setosa
Ponto 27: 5 3.4 1.6 0.4 - Iris-setosa
Ponto 28: 5.2 3.5 1.5 0.2 - Iris-setosa
Ponto 29: 5.2 3.4 1.4 0.2 - Iris-setosa
Ponto 30: 4.7 3.2 1.6 0.2 - Iris-setosa
Ponto 31: 4.8 3.1 1.6 0.2 - Iris-setosa
Ponto 32: 5.4 3.4 1.5 0.4 - Iris-setosa
Ponto 33: 5.2 4.1 1.5 0.1 - Iris-setosa
Ponto 34: 5.5 4.2 1.4 0.2 - Iris-setosa
Ponto 35: 4.9 3.1 1.5 0.1 - Iris-setosa
Ponto 36: 5 3.2 1.2 0.2 - Iris-setosa
Ponto 37: 5.5 3.5 1.3 0.2 - Iris-setosa
Ponto 38: 4.9 3.1 1.5 0.1 - Iris-setosa
Ponto 39: 4.4 3 1.3 0.2 - Iris-setosa
Ponto 40: 5.1 3.4 1.5 0.2 - Iris-setosa
Ponto 41: 5 3.5 1.3 0.3 - Iris-setosa
Ponto 42: 4.5 2.3 1.3 0.3 - Iris-setosa
Ponto 43: 4.4 3.2 1.3 0.2 - Iris-setosa
Ponto 44: 5 3.5 1.6 0.6 - Iris-setosa
Ponto 45: 5.1 3.8 1.9 0.4 - Iris-setosa
Ponto 46: 4.8 3 1.4 0.3 - Iris-setosa
Ponto 47: 5.1 3.8 1.6 0.2 - Iris-setosa
Ponto 48: 4.6 3.2 1.4 0.2 - Iris-setosa
Ponto 49: 5.3 3.7 1.5 0.2 - Iris-setosa
Ponto 50: 5 3.3 1.4 0.2 - Iris-setosa
Atributos do Cluster: 5.006 3.418 1.464 0.244
```

```
Cluster 2
Ponto 88: 6.3 2.3 4.4 1.3 - Iris-versicolor
Ponto 52: 6.4 3.2 4.5 1.5 - Iris-versicolor
Ponto 55: 6.5 2.8 4.6 1.5 - Iris-versicolor
Ponto 59: 6.6 2.9 4.6 1.3 - Iris-versicolor
Ponto 63: 6 2.2 4 1 - Iris-versicolor
Ponto 64: 6.1 2.9 4.7 1.4 - Iris-versicolor
Ponto 66: 6.7 3.1 4.4 1.4 - Iris-versicolor
Ponto 68: 5.8 2.7 4.1 1 - Iris-versicolor
Ponto 69: 6.2 2.2 4.5 1.5 - Iris-versicolor
Ponto 72: 6.1 2.8 4 1.3 - Iris-versicolor
Ponto 73: 6.3 2.5 4.9 1.5 - Iris-versicolor
Ponto 74: 6.1 2.8 4.7 1.2 - Iris-versicolor
Ponto 75: 6.4 2.9 4.3 1.3 - Iris-versicolor
Ponto 76: 6.6 3 4.4 1.4 - Iris-versicolor
Ponto 77: 6.8 2.8 4.8 1.4 - Iris-versicolor
Ponto 82: 5.5 2.4 3.7 1 - Iris-versicolor
Ponto 83: 5.8 2.7 3.9 1.2 - Iris-versicolor
Ponto 93: 5.8 2.6 4 1.2 - Iris-versicolor
Ponto 98: 6.2 2.9 4.3 1.3 - Iris-versicolor
Ponto 120: 6 2.2 5 1.5 - Iris-virginica
Ponto 124: 6.3 2.7 4.9 1.8 - Iris-virginica
Ponto 127: 6.2 2.8 4.8 1.8 - Iris-virginica
Ponto 134: 6.3 2.8 5.1 1.5 - Iris-virginica
Ponto 147: 6.3 2.5 5 1.9 - Iris-virginica
Ponto 51: 7 3.2 4.7 1.4 - Iris-versicolor
Ponto 54: 5.5 2.3 4 1.3 - Iris-versicolor
Ponto 56: 5.7 2.8 4.5 1.3 - Iris-versicolor
Ponto 57: 6.3 3.3 4.7 1.6 - Iris-versicolor
Ponto 60: 5.2 2.7 3.9 1.4 - Iris-versicolor
Ponto 62: 5.9 3 4.2 1.5 - Iris-versicolor
Ponto 65: 5.6 2.9 3.6 1.3 - Iris-versicolor
Ponto 67: 5.6 3 4.5 1.5 - Iris-versicolor
Ponto 70: 5.6 2.5 3.9 1.1 - Iris-versicolor
Ponto 71: 5.9 3.2 4.8 1.8 - Iris-versicolor
Ponto 79: 6 2.9 4.5 1.5 - Iris-versicolor
Ponto 81: 5.5 2.4 3.8 1.1 - Iris-versicolor
Ponto 84: 6 2.7 5.1 1.6 - Iris-versicolor
Ponto 85: 5.4 3 4.5 1.5 - Iris-versicolor
Ponto 86: 6 3.4 4.5 1.6 - Iris-versicolor
Ponto 87: 6.7 3.1 4.7 1.5 - Iris-versicolor
Ponto 89: 5.6 3 4.1 1.3 - Iris-versicolor
Ponto 90: 5.5 2.5 4 1.3 - Iris-versicolor
Ponto 91: 5.5 2.6 4.4 1.2 - Iris-versicolor
Ponto 92: 6.1 3 4.6 1.4 - Iris-versicolor
Ponto 95: 5.6 2.7 4.2 1.3 - Iris-versicolor
Ponto 96: 5.7 3 4.2 1.2 - Iris-versicolor
Ponto 97: 5.7 2.9 4.2 1.3 - Iris-versicolor
Ponto 100: 5.7 2.8 4.1 1.3 - Iris-versicolor
Ponto 102: 5.8 2.7 5.1 1.9 - Iris-virginica
Ponto 107: 4.9 2.5 4.5 1.7 - Iris-virginica
Ponto 114: 5.7 2.5 5 2 - Iris-virginica
Ponto 122: 5.6 2.8 4.9 2 - Iris-virginica
Ponto 128: 6.1 3 4.9 1.8 - Iris-virginica
Ponto 139: 6 3 4.8 1.8 - Iris-virginica
Ponto 143: 5.8 2.7 5.1 1.9 - Iris-virginica
Ponto 150: 5.9 3 5.1 1.8 - Iris-virginica
Ponto 58: 4.9 2.4 3.3 1 - Iris-versicolor
Ponto 61: 5 2 3.5 1 - Iris-versicolor
Ponto 80: 5.7 2.6 3.5 1 - Iris-versicolor
Ponto 94: 5 2.3 3.3 1 - Iris-versicolor
Ponto 115: 5.8 2.8 5.1 2.4 - Iris-virginica
Ponto 99: 5.1 2.5 3 1.1 - Iris-versicolor
Atributos do Cluster: 5.90161 2.74839 4.39355 1.43387
```

```
Cluster 3
Ponto 140: 6.9 3.1 5.4 2.1 - Iris-virginica
Ponto 78: 6.7 3 5 1.7 - Iris-versicolor
Ponto 101: 6.3 3.3 6 2.5 - Iris-virginica
Ponto 103: 7.1 3 5.9 2.1 - Iris-virginica
Ponto 104: 6.3 2.9 5.6 1.8 - Iris-virginica
Ponto 105: 6.5 3 5.8 2.2 - Iris-virginica
Ponto 106: 7.6 3 6.6 2.1 - Iris-virginica
Ponto 108: 7.3 2.9 6.3 1.8 - Iris-virginica
Ponto 109: 6.7 2.5 5.8 1.8 - Iris-virginica
Ponto 110: 7.2 3.6 6.1 2.5 - Iris-virginica
Ponto 111: 6.5 3.2 5.1 2 - Iris-virginica
Ponto 112: 6.4 2.7 5.3 1.9 - Iris-virginica
Ponto 113: 6.8 3 5.5 2.1 - Iris-virginica
Ponto 116: 6.4 3.2 5.3 2.3 - Iris-virginica
Ponto 117: 6.5 3 5.5 1.8 - Iris-virginica
Ponto 118: 7.7 3.8 6.7 2.2 - Iris-virginica
Ponto 119: 7.7 2.6 6.9 2.3 - Iris-virginica
Ponto 121: 6.9 3.2 5.7 2.3 - Iris-virginica
Ponto 123: 7.7 2.8 6.7 2 - Iris-virginica
Ponto 125: 6.7 3.3 5.7 2.1 - Iris-virginica
Ponto 126: 7.2 3.2 6 1.8 - Iris-virginica
Ponto 129: 6.4 2.8 5.6 2.1 - Iris-virginica
Ponto 130: 7.2 3 5.8 1.6 - Iris-virginica
Ponto 131: 7.4 2.8 6.1 1.9 - Iris-virginica
Ponto 132: 7.9 3.8 6.4 2 - Iris-virginica
Ponto 133: 6.4 2.8 5.6 2.2 - Iris-virginica
Ponto 135: 6.1 2.6 5.6 1.4 - Iris-virginica
Ponto 136: 7.7 3 6.1 2.3 - Iris-virginica
Ponto 137: 6.3 3.4 5.6 2.4 - Iris-virginica
Ponto 138: 6.4 3.1 5.5 1.8 - Iris-virginica
Ponto 141: 6.7 3.1 5.6 2.4 - Iris-virginica
Ponto 142: 6.9 3.1 5.1 2.3 - Iris-virginica
Ponto 144: 6.8 3.2 5.9 2.3 - Iris-virginica
Ponto 145: 6.7 3.3 5.7 2.5 - Iris-virginica
Ponto 146: 6.7 3 5.2 2.3 - Iris-virginica
Ponto 148: 6.5 3 5.2 2 - Iris-virginica
Ponto 149: 6.2 3.4 5.4 2.3 - Iris-virginica
Ponto 53: 6.9 3.1 4.9 1.5 - Iris-versicolor
Atributos do Cluster: 6.85 3.07368 5.74211 2.07105

Tempo passado para o processo de agrupamento: 0.004
```

Após verificação dos clusters calculamos:

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
23%
Porcentagem de pontos incorretos no cluster: 3
5.3%
Porcentagem total de pontos agrupados incorretamente: 11%
Tempo passado para o processo de verificacao: 0.009

K = 3

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
23%
Porcentagem de pontos incorretos no cluster: 3

5.3%
Porcentagem total de pontos agrupados incorretamente: 11%
Tempo passado para o processo de verificacao: 0.007

K = 5

Porcentagem de pontos incorretos no cluster: 1
5.3%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
23%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem total de pontos agrupados incorretamente: 11%
Tempo passado para o processo de verificacao: 0.01

K = 5

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
41%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
3.7%
Porcentagem total de pontos agrupados incorretamente: 12%
Tempo passado para o processo de verificacao: 0.007

K = 7

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
6.9%
Porcentagem de pontos incorretos no cluster: 5

0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
3.7%
Porcentagem total de pontos agrupados incorretamente: 2%
Tempo passado para o processo de verificacao: 0.008

K = 7

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
3.6%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
36%
Porcentagem de pontos incorretos no cluster: 7
0%
Porcentagem total de pontos agrupados incorretamente: 9.3%
Tempo passado para o processo de verificacao: 0.008

K = 7

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
3.7%
Porcentagem de pontos incorretos no cluster: 3
41%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
0%
Porcentagem total de pontos agrupados incorretamente: 12%
Tempo passado para o processo de verificacao: 0.008

K = 9

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
9.7%
Porcentagem de pontos incorretos no cluster: 8
0%
Porcentagem de pontos incorretos no cluster: 9
4.2%
Porcentagem total de pontos agrupados incorretamente: 2.7%
Tempo passado para o processo de verificacao: 0.008

K = 9

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
5%
Porcentagem de pontos incorretos no cluster: 8
17%
Porcentagem de pontos incorretos no cluster: 9
0%
Porcentagem total de pontos agrupados incorretamente: 2.7%
Tempo passado para o processo de verificacao: 0.009

K = 9

Porcentagem de pontos incorretos no cluster: 1
0%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
42%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
0%
Porcentagem de pontos incorretos no cluster: 8
7.7%
Porcentagem de pontos incorretos no cluster: 9
0%
Porcentagem total de pontos agrupados incorretamente: 9.3%
Tempo passado para o processo de verificacao: 0.009

K = 11

Porcentagem de pontos incorretos no cluster: 1
25%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
6.2%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
0%
Porcentagem de pontos incorretos no cluster: 8
0%
Porcentagem de pontos incorretos no cluster: 9
20%
Porcentagem de pontos incorretos no cluster: 10
0%
Porcentagem de pontos incorretos no cluster: 11

0%
Porcentagem total de pontos agrupados incorretamente: 3.3%
Tempo passado para o processo de verificacao: 0.008

K = 11

Porcentagem de pontos incorretos no cluster: 1
17%
Porcentagem de pontos incorretos no cluster: 2
0%
Porcentagem de pontos incorretos no cluster: 3
6.2%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7
0%
Porcentagem de pontos incorretos no cluster: 8
0%
Porcentagem de pontos incorretos no cluster: 9
0%
Porcentagem de pontos incorretos no cluster: 10
0%
Porcentagem de pontos incorretos no cluster: 11
0%
Porcentagem total de pontos agrupados incorretamente: 2.7%
Tempo passado para o processo de verificacao: 0.008

K = 11

Porcentagem de pontos incorretos no cluster: 1
40%
Porcentagem de pontos incorretos no cluster: 2
33%
Porcentagem de pontos incorretos no cluster: 3
0%
Porcentagem de pontos incorretos no cluster: 4
0%
Porcentagem de pontos incorretos no cluster: 5
0%
Porcentagem de pontos incorretos no cluster: 6
0%
Porcentagem de pontos incorretos no cluster: 7

0%
Porcentagem de pontos incorretos no cluster: 8
0%
Porcentagem de pontos incorretos no cluster: 9
0%
Porcentagem de pontos incorretos no cluster: 10
0%
Porcentagem de pontos incorretos no cluster: 11
0%
Porcentagem total de pontos agrupados incorretamente: 5.3%
Tempo passado para o processo de verificacao: 0.008

Ao tentarmos dividir a amostra de pontos em 3 clusters, podemos ver uma porcentagem de classificação errada de 23% em todas as tentativas. O motivo é o fato de que os atributos entre alguns pontos das classes Iris-virginica e Iris-versicolor serem muito próximos, o que acaba por fazer com que essas duas classes acabem se misturando em testes com valores de K menores. Já a classe Iris-setosa possui valores de atributos mais distintos, e acaba na grande maioria dos testes tendo todos seus pontos agrupados separados.

O impacto da similaridade dos valores de atributos diminui ao tentarmos separar a amostra em mais classes, já que haverá uma maior quantidade de centróides, e então a chance de pontos diferentes estarem próximos ao mesmo centróide diminui, como podemos ver nos testes pela porcentagem total de pontos agrupados incorretamente, que diminui ao aumentarmos o valor de K.

Utilizamos o valor 100 para quantidade máximas de iterações no processamento dos pontos, pois verificamos em diversos teste que existe uma faixa de aproximadamente 100 - 150 em que os resultados são otimizados, enquanto com maxIter < 100 e maxIter > 150, a porcentagem de erros se torna exorbitante na maioria dos testes.

Comparação dos algoritmos analisados

Enquanto o algoritmo KNN visa utilizar uma amostra de testes para ensinar o programa como classificar uma amostra desconhecida, o algoritmo Kmeans tem como objetivo agrupar os pontos da amostra em diferentes grupos de mesma classe a partir dos atributos de cada ponto.

Podemos ver pelos resultados obtidos acima que apesar do algoritmo KNN ter uma consistência de sucesso maior, o Kmeans diminui constantemente sua porcentagem de erro total conforme o valor de K aumenta. Ao mesmo tempo, porém, o quanto devemos incrementar o valor de K varia com a necessidade de agrupamento do usuário e com o tamanho e escala da amostra de dados.

Divisão do Trabalho

Knn feito por Pedro
Kmeans feito por Felipe

Conclusão

Pelos resultados obtidos podemos concluir que o algoritmo KNN possui uma consistência maior nos resultados, visto que é um algoritmo de aprendizado de máquina supervisionado, enquanto KMeans é não supervisionado.

Vemos que o algoritmo KNN tem como principal foco a classificação de amostras com classes desconhecidas a partir da classificação de amostras conhecidas, ou seja, seus vizinhos mais próximos, a partir da análise da distância entre seus atributos e os atributos da amostra conhecida, enquanto o algoritmo Kmeans tem como objetivo a classificação de amostras a partir de atributos escolhidos aleatoriamente de outras amostras, que se tornam centros de aglomerados de amostras que em teoria devem ser de mesma classe.

Enquanto KNN necessita que encontremos um valor de K (vizinhos) que nos proporcione a maior quantidade de acertos, é possível que ao aumentarmos demais este valor, a proporção de acertos diminua, enquanto no Kmeans ao aumentarmos o valor de K (clusters ou grupos) podemos garantir sempre que a porcentagem de acertos aumentará.

Há então, no Kmeans, uma grande discrepância entre K = 1 e K = quantidade de pontos da amostra, já que no primeiro todos os pontos estarão aglomerados no mesmo grupo e no segundo haverá um grupo por ponto da amostra. O valor de K em Kmeans então deve ser escolhido não apenas pela minimização da porcentagem de erro como no KNN, mas também pelo tamanho da amostra e pelas necessidades de agrupamento do próprio usuário.