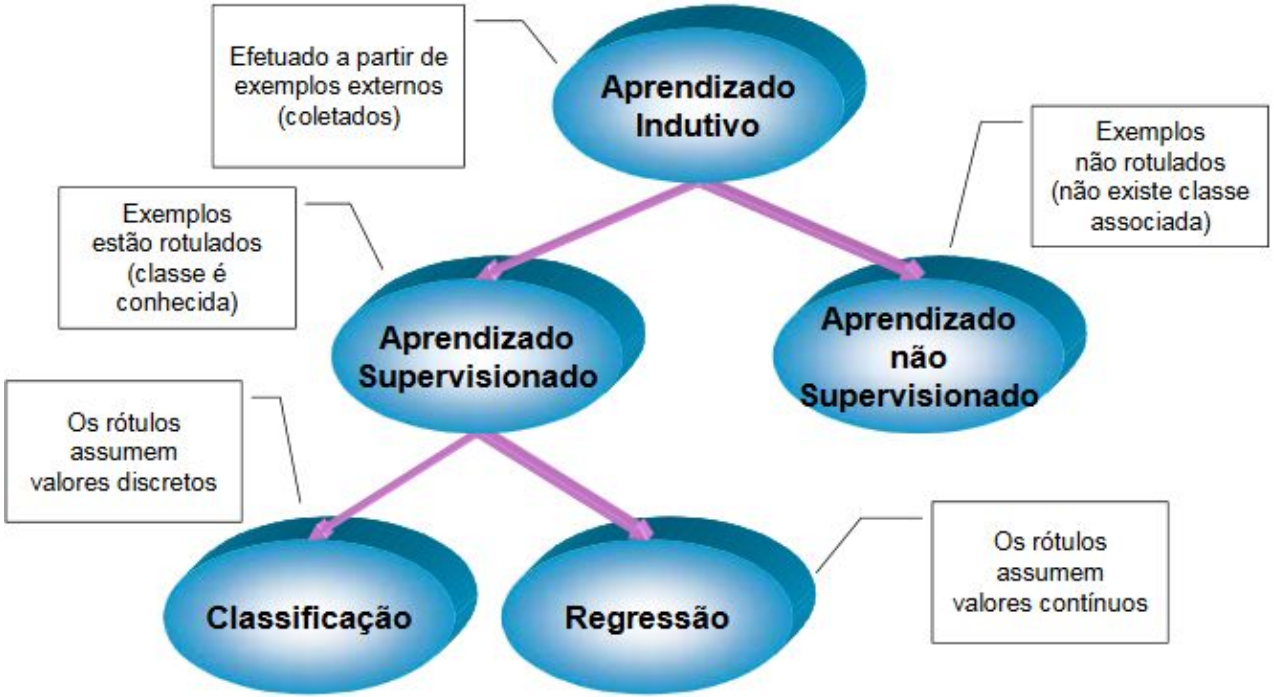
K-Nearest Neighbors

Diego Addan

Unibrasil 2019

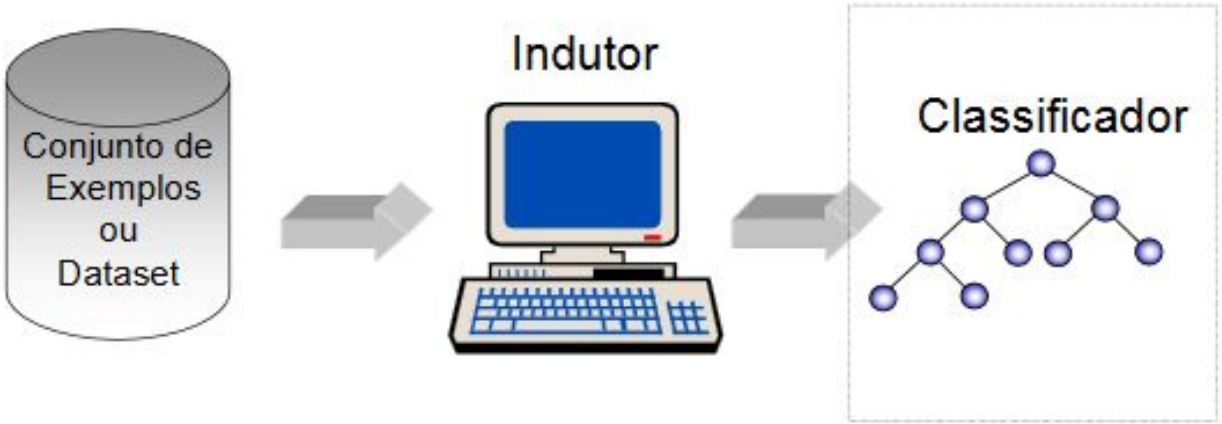
Aprendizado de máquina

Hierarquia do Aprendizado



Classificador

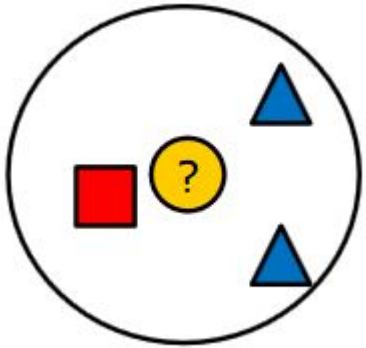
Dado um conjunto de exemplos, o classificador é a saída do indutor.



Algorítmo k-NN

Observa-se alguns pares de exemplos de entrada e saída, de forma a **aprender** uma **função** que mapeia a entrada para a saída.

Usado para classificar objetos com base em exemplos de treinamento que estão mais próximos no espaço de características.



Algorítmo k-NN

Para utilizar o KNN é necessário:

(1) Um conjunto de exemplos de treinamento.

(2) Definir uma métrica para calcular a distância entre os exemplos de treinamento.

(3) Definir o valor de K (o número de vizinhos mais próximos que serão considerados pelo algoritmo).

Algorítmo k-NN

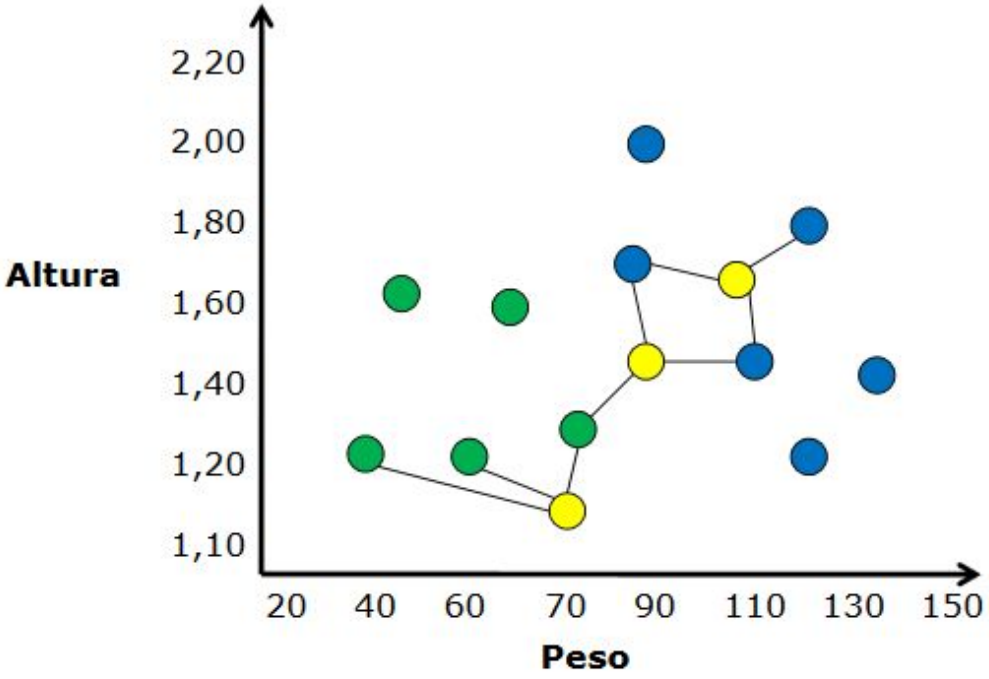
Classificar um exemplo desconhecido com o algoritmo KNN consiste em:

(1) Calcular a distância entre o exemplo desconhecido e o outros exemplos do conjunto de treinamento.

(2) Identificar os K vizinhos mais próximos.

(3) Utilizar o rótulo da classe dos vizinhos mais próximos para determinar o rótulo de classe do exemplo desconhecido (votação majoritária).

Algorítmo k-NN



Algorítmo k-NN

Calculando a distância entre dois pontos: Existem várias formas diferentes de calcular essa distância. A mais simples é a distância euclidiana:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

É importante normalizar os dados.

Outras formas de mediar a distancia: Distância de Mahalanobis (correlação). Distância de Minkowsky (Euclideana e Manhattan).

Algorítmo k-NN

Determinando a classe do exemplo desconhecido a partir da de lista de vizinhos mais próximos:

Considera-se o voto majoritário entre os rótulos de classe dos K vizinhos mais próximos.

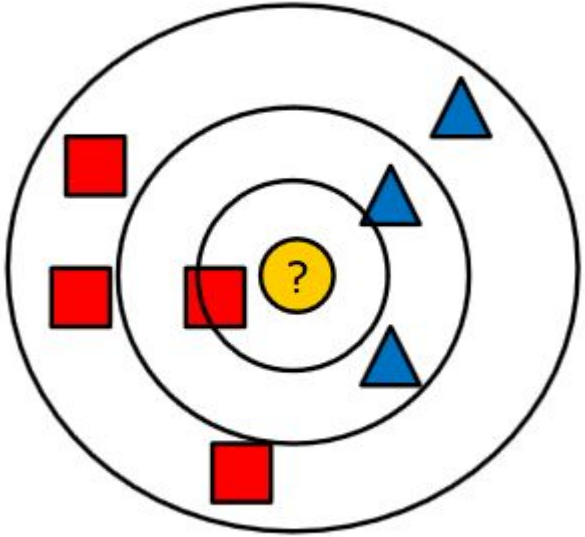
Como escolher o valor de K?

Algorítmo k-NN

K = 1 Pertence a classe de quadrados.

K = 3 Pertence a classe de triângulos.

K = 7 Pertence a classe de quadrados



Algorítmo k-NN

Como escolher o valor de K?

Se K for muito pequeno, a classificação fica sensível a pontos de ruído.

Se k é muito grande, a vizinhança pode incluir elementos de outras classes.

Além disso, é necessário sempre escolher um valor ímpar para K, assim se evita empates na votação.

Algorítmo k-NN

A precisão da classificação utilizando o algoritmo KNN depende fortemente do modelo de dados.

Na maioria das vezes os atributos precisam ser **normalizados** para evitar que as medidas de distância sejam dominado por um único atributo.

Exemplos:

- Altura de uma pessoa pode variar de 1,20 a 2,10.
- Peso de uma pessoa pode variar de 40 kg a 150 kg.
- O salário de uma pessoa podem variar de R$ 800 a R$ 20.000.

Algorítmo k-NN

Vantagens:

Técnica simples e facilmente implementada.

Bastante flexível.

Em alguns casos apresenta ótimos resultados.

Algorítmo k-NN

Desvantagens: Classificar um exemplo desconhecido pode ser um processo computacionalmente complexo. Requer um cálculo de distância para cada exemplo de treinamento.

Pode consumir muito tempo quando o conjunto de treinamento é muito grande.

A precisão da classificação pode ser severamente degradada pela presença de ruído ou características irrelevantes.

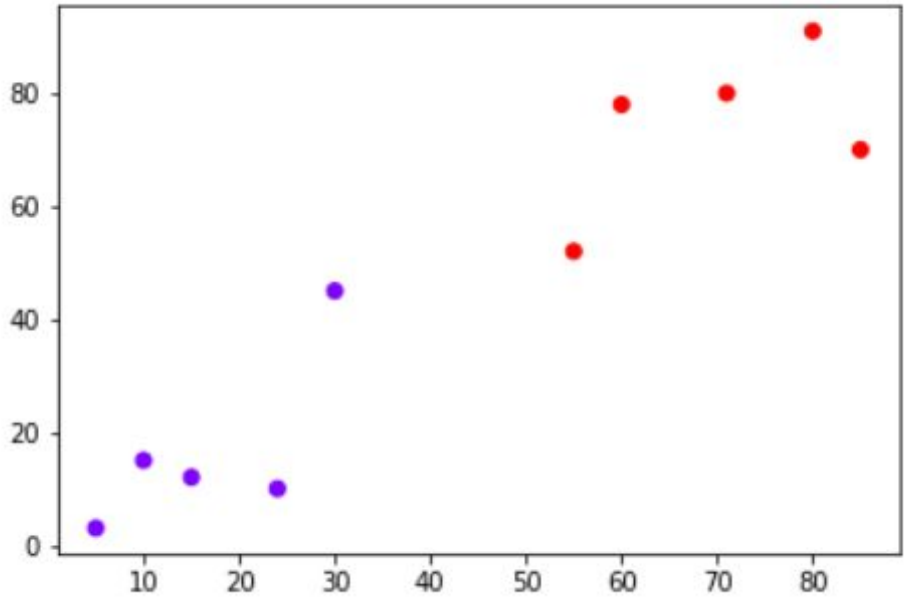
Algorítmo k-NN - Implementação

Implementação (utilizando Python e Scikit-learn)

- Calcula a distância de um novo ponto de dados a todos os outros pontos de dados de treinamento.
- A distância pode ser de qualquer tipo, por exemplo, Euclidean ou Manhattan etc.
- Em seguida, selecciona os pontos de dados K mais próximos, em que K pode ser qualquer número inteiro. Finalmente, ele atribui o ponto de dados à classe à qual a maioria dos pontos de dados K pertence.

Algorítmo k-NN - Implementação

Suponha que você tenha um conjunto de dados com duas variáveis, que, quando plotadas, se parece com o da figura a seguir.

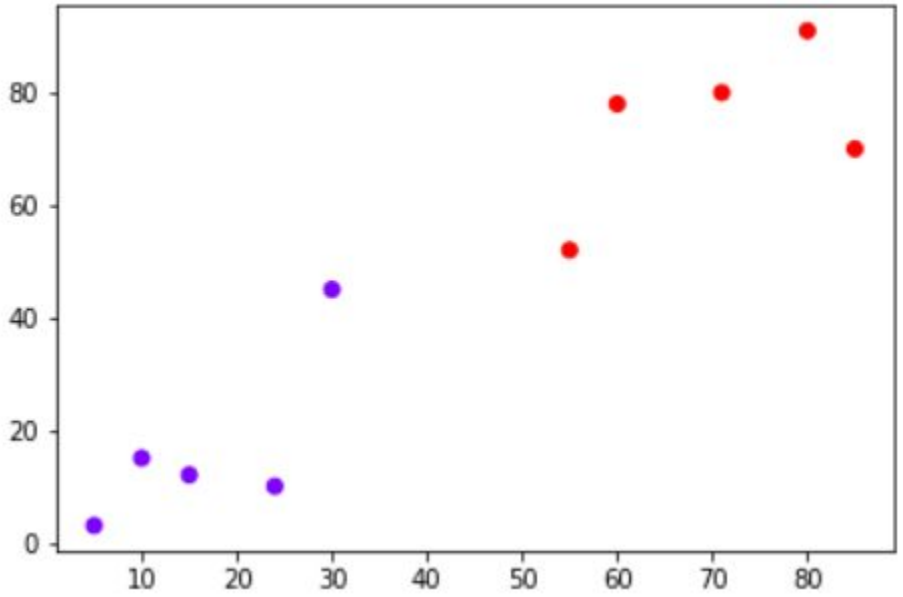


Algorítmo k-NN - Implementação

Sua tarefa é classificar um novo ponto de dados com 'X' em classe "Blue" ou classe "Red".

Os valores de coordenadas do ponto

de dados são x = 45 e y = 50.

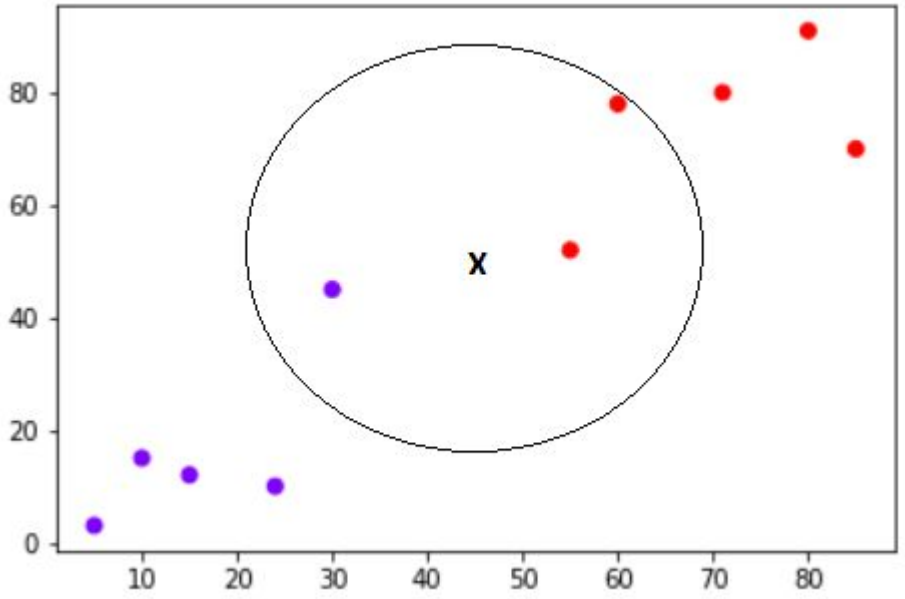


Algorítmo k-NN - Implementação

Sua tarefa é classificar um novo ponto de dados com 'X' em classe "Blue" ou classe "Red".

Suponha que o valor de K seja 3.
O algoritmo KNN começa calculando a distância do ponto X de todos os pontos.

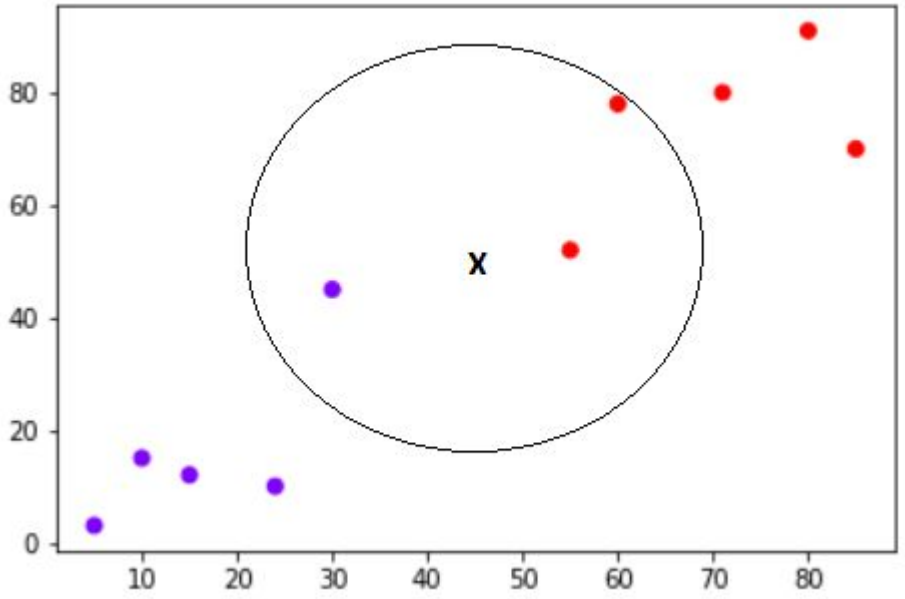
Em seguida, ele encontra os 3 pontos mais próximos com a menor distância até o ponto X.



Algorítmo k-NN - Implementação

A etapa final do algoritmo KNN é atribuir um novo ponto à classe à qual a maioria dos três pontos mais próximos pertence.

Dois dos três pontos mais próximos pertencem à classe "Red" enquanto um pertence à classe "Blue". Portanto, o novo ponto de dados será classificado como "Red".



Algorítmo k-NN - Implementação

Considerações de implementação:

- Como dito anteriormente, é um algoritmo de aprendizagem preguiçoso e, portanto, não requer treinamento antes de fazer previsões em tempo real.

- Isso torna o algoritmo KNN muito mais rápido do que outros algoritmos que exigem treinamento, por exemplo, SVM, regressão linear, etc.

- Como o algoritmo não requer treinamento antes de fazer previsões, novos dados podem ser adicionados sem problemas.

- Existem apenas dois parâmetros necessários para implementar KNN, isto é, o valor de K e a função de distância (por exemplo, Euclidean ou Manhattan, etc.).

Algorítmo k-NN - Implementação

Considerações de implementação:

- O algoritmo KNN não funciona bem com dados dimensionais elevados porque, com grande número de dimensões, torna-se difícil para o algoritmo calcular a distância em cada dimensão.

- O algoritmo KNN tem um alto custo de previsão para grandes conjuntos de dados. Isso ocorre porque, em grandes conjuntos de dados, o custo de calcular a distância entre o novo ponto e cada ponto existente torna-se maior.

- Por fim, o algoritmo KNN não funciona bem com recursos categóricos, pois é difícil encontrar a distância entre dimensões com recursos categóricos.

Algorítmo k-NN - Implementação

O conjunto de dados utilizado no exemplo implementado consiste em quatro atributos: largura da sépala, comprimento da sépala, largura da pétala e comprimento da pétala (sepal-width, sepal-length, petal-width and petal-length).

Estes são os atributos de tipos específicos de caracteristicas de íris.

A tarefa é prever a classe à qual esses exemplos pertencem. Existem três classes no conjunto de dados: Iris-setosa, Iris-versicolor e Iris-virginica.

Algorítmo k-NN - Implementação

Importando o Dataset

Para importar o conjunto de dados e carregá-lo em memória foi utilizado dataframe do pandas:

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

# Assign colum names to the dataset
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']

# Read dataset to pandas dataframe
dataset = pd.read_csv(url, names=names)
```

Algorítmo k-NN - Implementação

Importando o Dataset

dataset.head()

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Algorítmo k-NN - Implementação

Pré-processamento:

O próximo passo é dividir nosso conjunto de dados em seus atributos e rótulos. Para fazer isso, use o seguinte código:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 4].values
```

A variável X contém as quatro primeiras colunas do conjunto de dados (isto é, atributos), enquanto y contém os rótulos.

Algorítmo k-NN - Implementação

Antes de fazer qualquer previsão real, é sempre uma boa prática escalar os recursos para que todos possam ser uniformemente avaliados.

O algoritmo de descida de gradiente (que é usado em treinamento de redes neurais e outros algoritmos de aprendizado de máquina) também converge mais rápido com recursos normalizados.
O script a seguir executa o dimensionamento de recursos:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Algorítmo k-NN - Implementação

Treinamento e Previsões

É extremamente simples treinar o algoritmo KNN e fazer previsões com ele, especialmente ao usar o Scikit-Learn.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
```

Algorítmo k-NN - Implementação

Treinamento e Previsões

A primeira etapa é importar a classe KNeighborsClassifier da biblioteca sklearn.neighbors. Na segunda linha, essa classe é inicializada com um parâmetro, ou seja, n_neigbours. Este é basicamente o valor para o K.

Não existe um valor ideal para K e ele é selecionado após o teste e a avaliação; entretanto, para começar, 5 parece ser o valor mais comumente usado para o algoritmo KNN.

O passo final é fazer previsões sobre nossos dados de teste. Para fazer isso, execute o seguinte comando: **y_pred = classifier.predict(X_test)**

Algorítmo k-NN - Implementação

Avaliando o Algoritmo

Para avaliar um algoritmo, a matriz de confusão, a precisão e a pontuação f1 são as métricas mais utilizadas. Os métodos **confusion_matrix** e **classification_report** do sklearn.metrics podem ser usados para calcular essas métricas.

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Algorítmo k-NN - Implementação

Avaliando o Algoritmo

A saída do script anterior será:

```
[[11  0  0]
 [ 0 13  0]
 [ 0  1  6]]
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        11
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00         6

    avg / total       1.00      1.00      1.00        30
```

Algorítmo k-NN - Implementação

Avaliando o Algoritmo

Os resultados mostram que o nosso algoritmo KNN foi capaz de classificar todos os 30 registros no conjunto de testes com 100% de precisão, o que é excelente.

Embora o algoritmo tenha funcionado muito bem com esse conjunto de dados, não espere os mesmos resultados com todos os aplicativos. Como observado anteriormente, o KNN nem sempre funciona tão bem com recursos de alta dimensionalidade ou categóricos.

Algorítmo k-NN - Implementação

Comparando a taxa de erros com o valor K

Na seção de treinamento e previsão, dissemos que não há como saber de antemão qual valor de K produz os melhores resultados na primeira tentativa.

Nós escolhemos aleatoriamente 5 como o valor K e isso simplesmente resulta em 100% de precisão.

Uma maneira de encontrar o melhor valor de K é plotar o gráfico do valor K e a taxa de erro correspondente para o conjunto de dados. Para isso, vamos plotar o erro médio para os valores previstos do conjunto de testes para todos os valores de K entre 1 e 40.

Algorítmo k-NN - Implementação

Comparando a taxa de erros com o valor K

Para fazer isso, vamos primeiro calcular a média de erro para todos os valores previstos, onde K varia de 1 e 40. Execute o seguinte script:

```
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error.append(np.mean(pred_i != y_test))
```

Em cada iteração, o erro médio dos valores previstos do conjunto de testes é calculado e o resultado é anexado à lista de erros.

Algorítmo k-NN - Implementação

Comparando a taxa de erros com o valor K

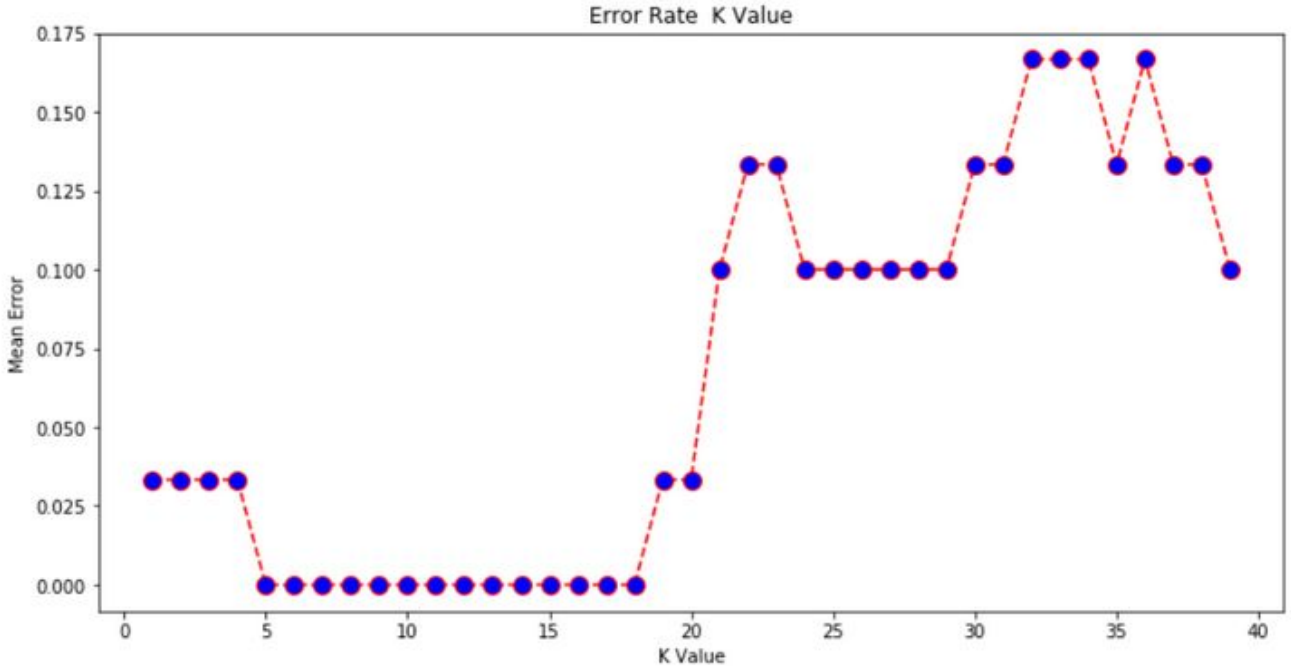
O próximo passo é plotar os valores de erro em relação aos valores de K. Execute o seguinte script para criar o enredo:

```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Algorítmo k-NN - Implementação

Comparando a taxa de erros com o valor K

O erro médio é zero quando o valor do K está entre 5 e 18.



Algorítmo k-NN - Conclusão

KNN é um algoritmo de classificação simples, mas poderoso. Não requer treinamento para fazer previsões, que é tipicamente uma das partes mais difíceis de um algoritmo de aprendizado de máquina.

O algoritmo KNN tem sido amplamente utilizado para encontrar similaridade de documentos e reconhecimento de padrões.

Também tem sido empregado para desenvolver sistemas de recomendação e para redução de dimensionalidade e etapas de pré-processamento para visão computacional, particularmente tarefas de reconhecimento facial.