Anotações sobre k-NN

Raphael Valdetaro

November 11, 2023

1 Introdução

O *k-NN* (k-Nearest Neighbors) é um método de aprendizado supervisionado versátil. Ele se destaca em tarefas de classificação, regressão e detecção de outliers. Este método é não paramétrico, o que significa que não faz suposições explícitas sobre a forma funcional dos dados, tornando-o especialmente útil em situações com limites de decisão complexos e não lineares. O k-NN opera na ideia de proximidade: pontos de dados semelhantes tendem a ter rótulos semelhantes. A escolha do valor de "k" influencia a suavidade da decisão e a sensibilidade ao ruído. Valores pequenos podem levar a modelos mais sensíveis a outliers, enquanto valores grandes podem suavizar demais as fronteiras de decisão.

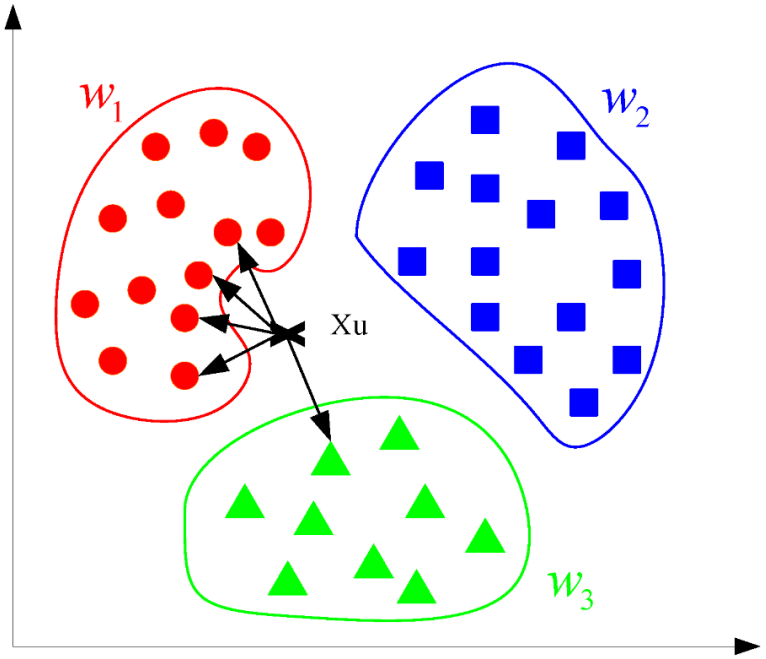

Figure 1: Exemplo do KNN.

Nesse contexto, a "proximidade" é geralmente medida pela distância euclidiana entre os pontos, que é uma métrica comum para avaliar a dissimilaridade entre instâncias. Apesar de sua simplicidade e interpretabilidade, a escolha do valor de k é crucial, impactando diretamente o desempenho do modelo. Um k muito pequeno pode levar a sobreajuste, enquanto um k muito grande pode suavizar demais as fronteiras entre as classes.

2 Preparação dos Dados

Na etapa de preparação dos dados, o algoritmo k-NN é exemplificado utilizando o conjunto de dados Iris, um conjunto clássico na área de aprendizado de máquina, transformando-o em um DataFrame do Pandas. Além da adição da coluna de classes para representar as diferentes espécies de íris, exploramos visualizações preliminares para compreender a distribuição e características dos dados. A análise exploratória nesse estágio inicial pode revelar insights valiosos que influenciam as decisões futuras no processo de modelagem. A divisão dos dados em atributos (x) e rótulos (y) ocorre, onde x representa as características das plantas e y as classes. Aqui, vale a pena destacar a importância da estratificação na divisão dos dados, especialmente em conjuntos desbalanceados, garantindo que as proporções das classes sejam mantidas tanto nos conjuntos de treino quanto nos de teste.

```
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd

iris = datasets.load_iris()
df_iris = pd.DataFrame(iris.data, columns=iris.feature_names)

# Adicionando a coluna de classes
df_iris['class'] = iris.target

# Dividindo os dados em atributos (x) e rótulos (y)
x = df_iris.iloc[:, :-1].values
y = df_iris.iloc[:, -1].values
```

3 Divisão dos Dados

A divisão do conjunto de dados em treino e teste é essencial para avaliar o desempenho do modelo em dados não vistos.Essa prática é vital para avaliar o desempenho do modelo em dados não vistos, proporcionando uma simulação realista das condições do mundo. A biblioteca scikit-learn é empregada para realizar essa divisão de maneira eficiente, reduzindo o risco de sobreajuste ao conjunto de treino.

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

4 Normalização dos Dados

A normalização dos dados é uma etapa crucial, especialmente para o k-NN, que depende das distâncias entre os pontos. Nesse contexto, aplicamos o `StandardScaler` para padronizar as características, assegurando que todas contribuam igualmente para as distâncias calculadas pelo algoritmo. Essa normalização é essencial para garantir que características em diferentes escalas não influenciem de forma desproporcional o resultado do modelo.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

5 Treinamento e Avaliação do Modelo

Com os dados normalizados, partimos para o treinamento do modelo k-NN. Avaliamos seu desempenho por meio de métricas fundamentais, como a matriz de confusão e o relatório de classificação. Essas métricas oferecem uma compreensão detalhada da precisão, recall e F1-score do modelo para cada classe, proporcionando uma visão abrangente de sua capacidade preditiva.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report

k = 8  # Escolha inicial de k
classifier = KNeighborsClassifier(n_neighbors=k)
classifier.fit(x_train, y_train)

y_pred = classifier.predict(x_test)

# Avaliando o modelo
cm = confusion_matrix(y_test, y_pred)
cr = classification_report(y_test, y_pred)

print("Matriz de Confusão:")
print(cm)
print("\nRelatório de Classificação:")
print(cr)
```

6 Ajustando Hiperparâmetros

A escolha criteriosa do hiperparâmetro "k" é de suma importância. Utilizamos técnicas avançadas, como a validação cruzada e a busca em grade, para explorar diferentes valores de "k" e identificar aquele que melhor se ajusta ao conjunto de dados. Esse processo é crucial para evitar sub ou sobreajuste, garantindo uma generalização eficaz do modelo para novos dados.

```
from sklearn.model_selection import GridSearchCV

# Definindo os parâmetros para ajustar
parametros = {'n_neighbors': [3, 5, 7, 9, 11]}

# Criando o classificador k-NN
knn = KNeighborsClassifier()

# Realizando a validação cruzada para encontrar o melhor valor de "k"
grid_search = GridSearchCV(knn, parametros, cv=5)
grid_search.fit(x_train, y_train)

# Obtendo o melhor valor de "k"
melhor_k = grid_search.best_params_['n_neighbors']

# Utilizando o melhor modelo
melhor_modelo = grid_search.best_estimator_
```

7 Avaliação de Desempenho Adicional

Além das métricas fundamentais, expandimos nossa análise incluindo métricas adicionais como acurácia, precisão, recall e F1-score. Essas métricas proporcionam uma visão mais holística do desempenho do modelo em diferentes aspectos, enriquecendo nossa avaliação.

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Calculando métricas adicionais
acuracia = accuracy_score(y_test, y_pred)
precisao = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f'Acurácia: {acuracia}')
print(f'Precisão: {precisao}')
print(f'Recall: {recall}')
print(f'F1 Score: {f1}')
```

8 Visualização dos Resultados

A etapa final consiste na visualização dos resultados. Utilizamos a biblioteca *seaborn* para criar uma matriz de confusão visualmente informativa. Essa visualização facilita a interpretação das previsões do modelo em relação às classes reais, contribuindo significativamente para uma compreensão mais profunda de seu comportamento em cenários diversos.

```
import seaborn as sns

# Visualizando a matriz de confusão
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.title("Matriz de Confusão")
plt.xlabel("Rótulos Preditos")
plt.ylabel("Rótulos Reais")
plt.show()
```

9 Conclusão

Em resumo, a implementação do algoritmo k-NN envolve uma série de passos essenciais para garantir um desempenho robusto e confiável. A preparação dos dados, a divisão adequada entre conjuntos de treino e teste, e a normalização são fundamentais para criar um modelo que generalize bem para dados não vistos. O treinamento do modelo e sua avaliação por meio de métricas como matriz de confusão e relatório de classificação oferecem insights valiosos sobre seu desempenho em diferentes classes.

A etapa de ajuste de hiperparâmetros destaca a importância de encontrar o valor ótimo para "k", evitando sub ou sobreajuste do modelo. A avaliação de desempenho adicional, com métricas como acurácia, precisão, recall e F1-score, proporciona uma compreensão mais completa da eficácia do modelo em diversas situações.

A visualização dos resultados, especialmente por meio de uma matriz de confusão, torna-se crucial para interpretar as previsões do modelo de forma intuitiva. A escolha do Seaborn para essa visualização adiciona uma camada de clareza à compreensão do comportamento do modelo.

Em última análise, a implementação cuidadosa desses passos, juntamente com a análise aprofundada das métricas de desempenho, contribui para a criação de modelos k-NN mais precisos e generalizáveis. Essa abordagem sistemática é essencial para aplicar o k-NN de maneira eficaz em uma variedade de problemas de classificação.

Dicionário de Termos

Estratificação: No contexto da divisão de dados, significa manter a proporção das classes nos conjuntos de treino e teste, garantindo uma representação equitativa.

Validação Cruzada (Leave-One-Out): Técnica que envolve dividir o conjunto de dados em partes, treinando o modelo em todas as partes, exceto uma, que é usada para teste. O Leave-One-Out é uma forma extrema de validação cruzada, onde cada observação é usada como conjunto de teste em uma iteração.

Matriz de Confusão: Uma tabela que mostra as previsões de um modelo em relação às classes reais, fornecendo uma visão detalhada do desempenho em termos de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos.

Sobreajuste (Overfitting): Ocorre quando um modelo se adapta demais aos dados de treino, perdendo a capacidade de generalização para novos dados.

Busca em Grade (Grid Search): Técnica que envolve testar diversas combinações de hiperparâmetros para encontrar a configuração mais eficaz para o modelo.

Hiperparâmetros: Parâmetros ajustáveis que não são aprendidos durante o treinamento do modelo e precisam ser definidos antes do treinamento, como o valor de "k" no algoritmo k-NN.

Acurácia: Uma métrica que mede a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões.

Curvas ROC e AUC: As Curvas ROC (Receiver Operating Characteristic) e AUC (Area Under the Curve) são ferramentas gráficas que ajudam a avaliar o desempenho de um modelo de classificação em diferentes pontos de operação.

Precision-Recall: Um par de métricas que fornecem informações sobre a precisão e o recall do modelo, especialmente úteis em conjuntos de dados desbalanceados.

Visualização Exploratória: Uma etapa inicial que envolve a exploração gráfica dos dados para ganhar insights sobre distribuição, padrões e características que podem influenciar as decisões de modelagem.

Normalização Robusta: Técnica de normalização que é menos sensível a outliers, garantindo que valores extremos não distorçam a escala dos dados.

Complexidade do Conjunto de Dados: Refere-se à intricidade e diversidade dos padrões presentes nos dados, influenciando a escolha adequada de hiperparâmetros e estratégias de modelagem.

Versatilidade: A capacidade do modelo de se adaptar e performar bem em diferentes cenários ou tipos de dados, destacando sua aplicabilidade em diversas situações.

Interpretabilidade: A capacidade de entender e explicar o funcionamento e as decisões do modelo, uma característica valiosa para modelos utilizados em aplicações práticas.

Enriquecimento da Análise: Adição de camadas mais profundas ou complexas à análise de dados, muitas vezes envolvendo métricas e visualizações além das básicas para uma compreensão mais refinada do desempenho do modelo.

Leave-One-Out (LOO): Uma técnica extrema de validação cruzada onde, em cada iteração, apenas um ponto de dados é usado como conjunto de teste, sendo uma forma intensiva de avaliação do modelo.