

► **Abordando Class Imbalance com o algoritmo KNN**

Pedro Antonio Resende Gulart
Mariana Filipa Ribeiro Ferreira
Ekaterina Aksenova

Grupo G_3_4

► Sumario

Objetivos de aprendizagem

Compreender a acurácia na prática e reconhecer como ela pode ser enganosa, entender profundamente o funcionamento do algoritmo KNN, explorar algoritmos de aprendizado não supervisionado e não linear, e analisar o efeito do desequilíbrio de classes em modelos de machine learning.

Sumario dos resultados

Tivemos como objetivo alterar o funcionamento do KNN para conseguir lidar com class imbalance em datasets binários, algo que foi atingido com o nosso algoritmo através de atribuição de pesos aos exemplos.

► Esboço de abordagem

- Exploração de Dados: Uma análise preliminar dos dados fornecidos.
- Pré-processamento de Dados: Correção de problemas nos dados (ex: valores ausentes) e engenharia de características (ex: criação de novas características ou remoção de características redundantes).
- Modelagem de Dados (Aprendizado Supervisionado): Uso de algoritmos de classificação para criar modelos de previsão.
- Avaliação de Dados: Análise da precisão dos modelos de previsão.
- Interpretação dos Resultados: Análise aprofundada do resultado do modelo de dados para criar melhores modelos e extrair informações sobre os dados.

► KNN (K Vizinhos Mais Próximos)

K-Nearest Neighbors (KNN) é um algoritmo de aprendizado supervisionado utilizado tanto para classificação quanto para regressão. Ele é conhecido como um algoritmo "lazy" porque não realiza aprendizado explícito durante a fase de treinamento. Em vez disso, o KNN simplesmente armazena todas as instâncias dos dados de treinamento e realiza os cálculos necessários apenas durante a fase de previsão.

A escolha do valor de k é crucial para o desempenho do algoritmo. Um valor maior de k pode ajudar a suprimir os efeitos do ruído nos dados, mas pode também tornar as fronteiras de classificação menos distintas, prejudicando a precisão. Em resumo, o KNN é um método simples e eficaz, mas que depende fortemente da escolha adequada de k e do cálculo eficiente das distâncias entre as instâncias.

► KNN (K Vizinhos Mais Próximos)

Quando uma nova amostra precisa ser classificada, o KNN calcula a distância entre essa nova amostra e todas as instâncias armazenadas no conjunto de dados de treinamento. Ele então identifica os k vizinhos mais próximos (onde k é um número inteiro positivo previamente definido) e classifica a nova amostra com base na classe mais frequente entre esses vizinhos.

A escolha do valor de k é crucial para o desempenho do algoritmo. Um valor maior de k pode ajudar a suprimir os efeitos do ruído nos dados, mas pode também tornar as fronteiras de classificação menos distintas, prejudicando a precisão. Em resumo, o KNN é um método simples e eficaz, mas que depende fortemente da escolha adequada de k e do cálculo eficiente das distâncias entre as instâncias.

Entendendo Class Imbalance

O desequilíbrio de classes (ou class imbalance) ocorre quando uma classe é significativamente mais representada do que outra(s) em um conjunto de dados, resultando em um número desproporcional de instâncias para cada classe. Esse problema é comum em diversas áreas, como diagnóstico médico, detecção de fraudes e reconhecimento de padrões. Entre os desafios gerados pelo desequilíbrio de classes estão a predição tendente à classe majoritária, métricas de desempenho enganosas (como alta acurácia global) e dificuldades na aprendizagem de certos algoritmos. Consequentemente, os modelos podem apresentar alta precisão global, mas falham em detectar corretamente a classe minoritária, pois os algoritmos tendem a favorecer a classe majoritária. Isso pode resultar em modelos que não generalizam bem e que possuem um desempenho insatisfatório na identificação de casos importantes pertencentes à classe menos representada, evidenciando a baixa sensibilidade e a tendência para a classe majoritária, onde a maioria dos votos tende a ser da classe mais frequente, causando alta taxa de erro para a classe minoritária.

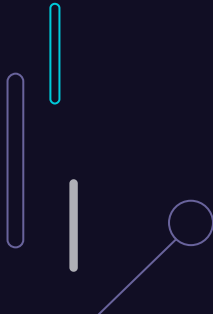
Exemplo

Em um conjunto de dados com 99 cachorros e apenas 1 gato, o algoritmo KNN pode classificar todas as instâncias como cachorros, alcançando uma precisão de 99%. No entanto, ele falharia em identificar o gato em todos os casos, resultando em uma taxa de erro de 100% para essa classe minoritária.



Exemplo

Num conjunto de dados com 99 cachorros e apenas 1 gato, o algoritmo KNN pode classificar todas as instâncias como cachorros, alcançando uma precisão de 99%. No entanto, ele falharia em identificar o gato em todos os casos, resultando em uma taxa de erro de 100% para essa classe minoritária.



► Solução Proposta: Tratando o Desequilíbrio de Classes com KNN usando pesos

Decidimos utilizar a técnica de atribuir peso às classes no algoritmo KNN.

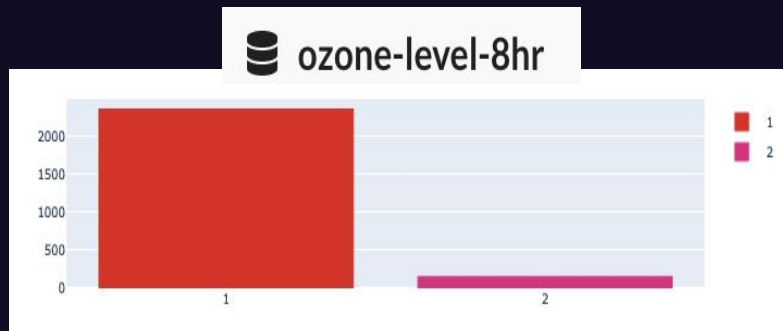
Essa escolha foi feita devido à sua simplicidade, eficácia e natureza intuitiva.

Ao atribuir pesos, o modelo é ajustado para levar em conta o desequilíbrio de classes, incentivando uma melhor identificação das instâncias da classe minoritária.

Esta abordagem complementa bem a flexibilidade do KNN.

► Estudo Empírico: Configuração Experimental

id	name
31	credit-g
38	sick
1067	kc1
1068	pc1
1461	bank-marketing
1464	blood-transfusion-service-center
1487	ozone-level-8hr
1590	adult
40701	churn
40983	wilt



Exemplo de um data set com 93,6/6,4 class imbalance

Data sets utilizados

- Os conjuntos de dados utilizados são binários, com um desequilíbrio de classes de pelo menos 30/70, havendo casos em que o desequilíbrio de classes é ainda maior.

► Métricas utilizadas

O F1-score, acurácia(accuracy), precisão(precision) e revocação(recall) são métricas essenciais para avaliar o desempenho de modelos de classificação.

- O F1-score combina precisão e revocação em um único valor, útil para classes desbalanceadas.
- A acurácia mede a proporção de previsões corretas, enquanto a precisão e a revocação destacam a precisão das previsões positivas e a capacidade do modelo em encontrar todos os exemplos positivos, respectivamente.
- Essas métricas oferecem uma visão abrangente da eficácia do modelo em sua tarefa de classificação.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$



Estudo empírico: Análise de resultados

1487: KNN -> Accuracy: 0.9349112426035503
F1 Score: 0.9180883022303141
Precision per class: [0.94578313 0.33333333]
Recall per class: [0.98742138 0.1]
F1 Score per class: [0.96615385 0.15384615]

KNN2 -> Accuracy: 0.9349112426035503
F1 Score: 0.920465202997901
Precision per class: [0.94758065 0.36363636]
Recall per class: [0.98532495 0.13333333]
F1 Score per class: [0.96608428 0.19512195]

1494: KNN -> Accuracy: 0.8293838862559242
F1 Score: 0.8310351738368302
Precision per class: [0.88888889 0.72368421]
Recall per class: [0.85106383 0.78571429]
F1 Score per class: [0.86956522 0.75342466]

KNN2 -> Accuracy: 0.8483412322274881
F1 Score: 0.8493565337303616
Precision per class: [0.89781022 0.75675676]
Recall per class: [0.87234043 0.8]
F1 Score per class: [0.88489209 0.77777778]

1068: KNN -> Accuracy: 0.9414414414414415
F1 Score: 0.9311365855019106
Precision per class: [0.9537037 0.5]
Recall per class: [0.98564593 0.23076923]
F1 Score per class: [0.96941176 0.31578947]

KNN2 -> Accuracy: 0.9369369369369369
F1 Score: 0.9279236783953765
Precision per class: [0.95348837 0.42857143]
Recall per class: [0.98086124 0.23076923]
F1 Score per class: [0.96698113 0.3]

40983: KKN -> Accuracy: 0.9762396694214877
F1 Score: 0.9730950356532211
Precision per class: [0.97547974 1.]
Recall per class: [1. 0.56603774]
F1 Score per class: [0.9875877 0.72289157]

KNN2 -> Accuracy: 0.9793388429752066
F1 Score: 0.9770480647859083
Precision per class: [0.97860963 1.]
Recall per class: [1. 0.62264151]
F1 Score per class: [0.98918919 0.76744186]

1050: KNN -> Accuracy: 0.8913738019169329
F1 Score: 0.8658742621974334
Precision per class: [0.91447368 0.11111111]
Recall per class: [0.97202797 0.03703704]
F1 Score per class: [0.94237288 0.05555556]

KNN2 -> Accuracy: 0.8722044728434505
F1 Score: 0.861973166771247
Precision per class: [0.91836735 0.15789474]
Recall per class: [0.94405594 0.11111111]
F1 Score per class: [0.93103448 0.13043478]

1464: KNN -> Accuracy: 0.7866666666666666
F1 Score: 0.765079365079365
Precision per class: [0.83333333 0.44444444]
Recall per class: [0.91666667 0.26666667]
F1 Score per class: [0.87301587 0.33333333]

KNN2 -> Accuracy: 0.7533333333333333
F1 Score: 0.7446382189239332
Precision per class: [0.832 0.36]
Recall per class: [0.86666667 0.3]
F1 Score per class: [0.84897959 0.32727273]

1049: KNN -> Accuracy: 0.863013698630137
F1 Score: 0.8292356709602671
Precision per class: [0.88256228 0.36363636]
Recall per class: [0.97254902 0.10810811]
F1 Score per class: [0.92537313 0.16666667]

KNN2 -> Accuracy: 0.8595890410958904
F1 Score: 0.8309572511828757
Precision per class: [0.88489209 0.35714286]
Recall per class: [0.96470588 0.13513514]
F1 Score per class: [0.92307692 0.19607843]

1489: KNN -> Accuracy: 0.8649398704902868
F1 Score: 0.8626940198962983
Precision per class: [0.88861076 0.79787234]
Recall per class: [0.92568449 0.71656051]
F1 Score per class: [0.90676884 0.75503356]

KNN2 -> Accuracy: 0.8899167437557817
F1 Score: 0.8886373268973798
Precision per class: [0.91012658 0.83505155]
Recall per class: [0.93741851 0.77388535]
F1 Score per class: [0.92357097 0.80330579]

1067: KNN -> Accuracy: 0.8815165876777251
F1 Score: 0.8656050718344458
Precision per class: [0.90609137 0.53571429]
Recall per class: [0.96486486 0.28846154]
F1 Score per class: [0.93455497 0.375]

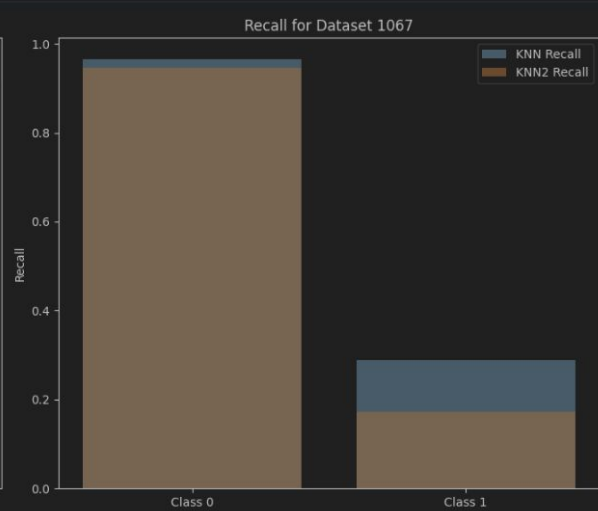
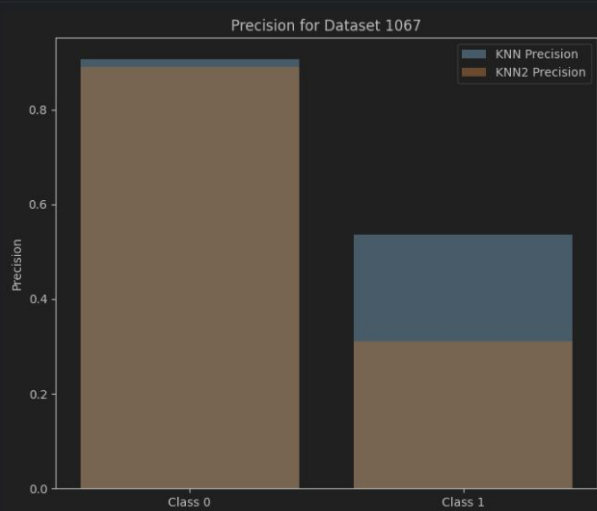
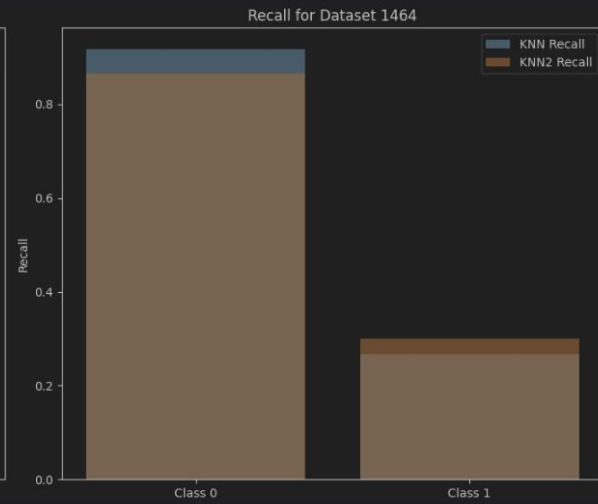
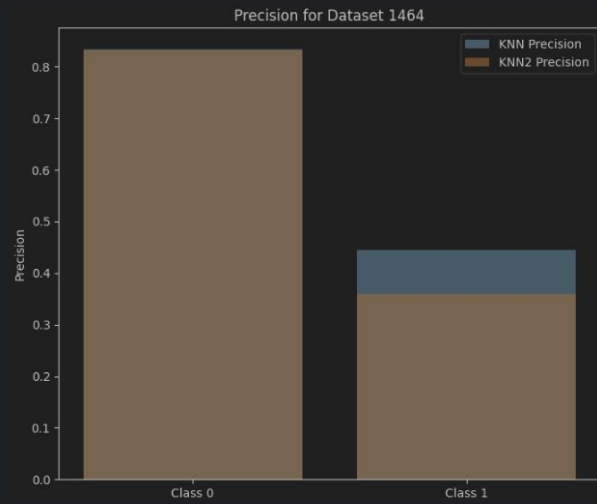
KNN2 -> Accuracy: 0.8507109004739337
F1 Score: 0.8317656323221783
Precision per class: [0.89058524 0.31034483]
Recall per class: [0.94594595 0.17307692]
F1 Score per class: [0.91743119 0.22222222]

1063: KNN -> Accuracy: 0.8
F1 Score: 0.7851273238556475
Precision per class: [0.84444444 0.53333333]
Recall per class: [0.91566265 0.36363636]
F1 Score per class: [0.87861272 0.43243243]

KNN2 -> Accuracy: 0.780952380952381
F1 Score: 0.7584217687074829
Precision per class: [0.82608696 0.46153846]
Recall per class: [0.91566265 0.27272727]
F1 Score per class: [0.86857143 0.34285714]

F1 Scores of KNN and KNN2 across Datasets





► Resultados Experimentais

Ao ajustarmos o algoritmo KNN para tratar o desequilíbrio de classes, notamos diferentes padrões nos resultados ao analisar vários conjuntos de dados.

Na maioria deles, observamos melhorias em todos os valores de métricas como acurácia, F1 score, precisão e recall.

No entanto, em alguns conjuntos de dados, vimos um cenário misto, onde houve melhoria em algumas métricas e diminuição em outras. Nesses casos, torna-se crucial avaliar cuidadosamente o desempenho do KNN2 em comparação com o KNN não ajustado, considerando as necessidades específicas do problema em questão e verificação de possibilidade de overfitting.

► Trabalho e Pesquisa Futuros

No estudo realizado, o algoritmo desenvolvido demonstrou eficácia exclusivamente em conjuntos de dados binários, limitando sua aplicabilidade em contextos com múltiplas classes.

Considerando esse cenário, sugere-se como trabalho futuro a adaptação do algoritmo para suportar conjuntos de dados não binários, explorando métodos como a codificação one-hot ou técnicas de divisão e conquista para lidar com a expansão de classes.

Além disso, outras estratégias para resolver o desbalanceamento de classes em KNN, como subamostragem ou superamostragem, merecem investigação para avaliar sua eficácia em comparação com a abordagem atualmente implementada, oferecendo assim perspectivas promissoras para melhorar a robustez e a aplicabilidade do modelo desenvolvido.

► Bibliografia

- <https://www.openml.org/>
- <https://elitedatascience.com/imbalanced-classes>
- <https://graphite-note.com/understanding-the-importance-of-f1-score-in-machine-learning-3/>