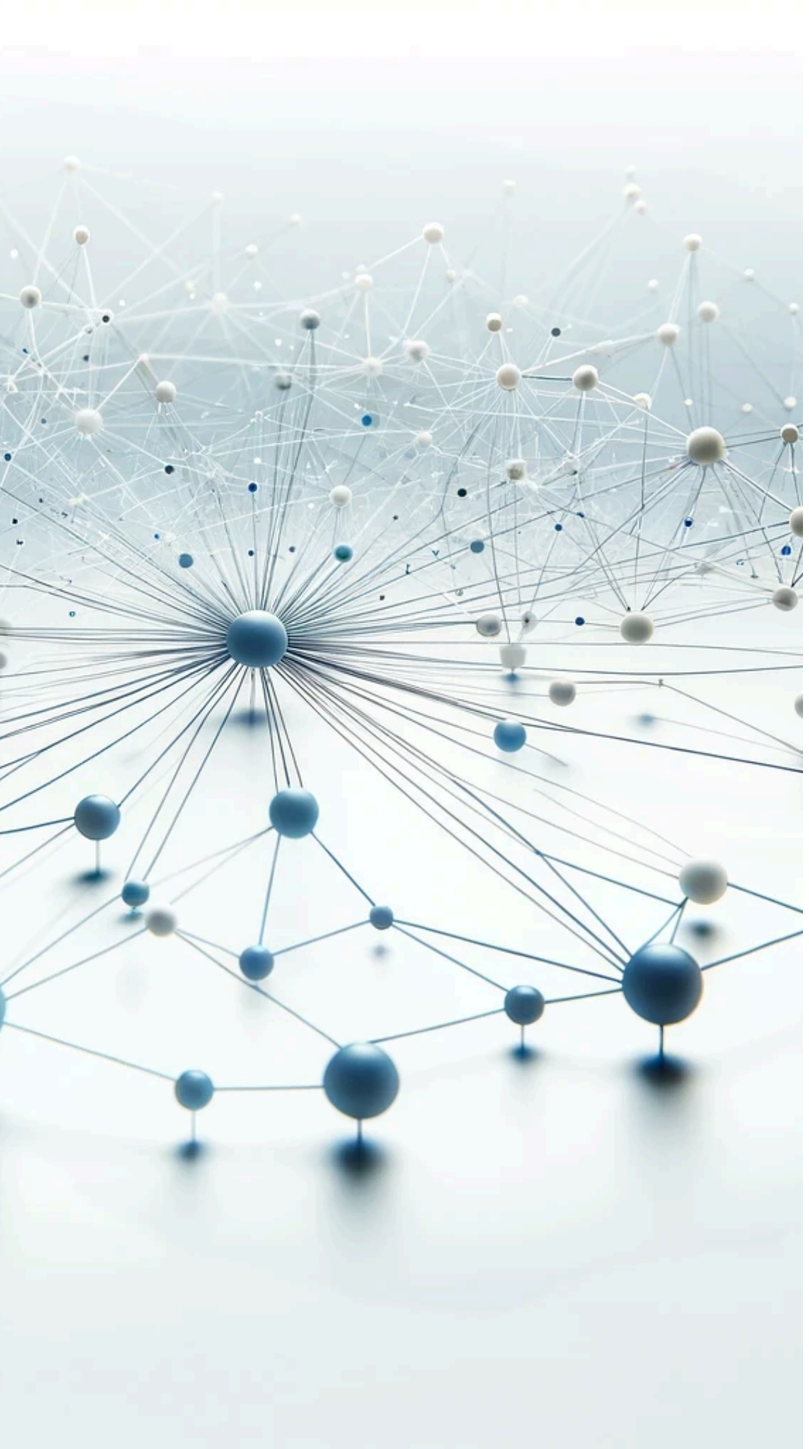


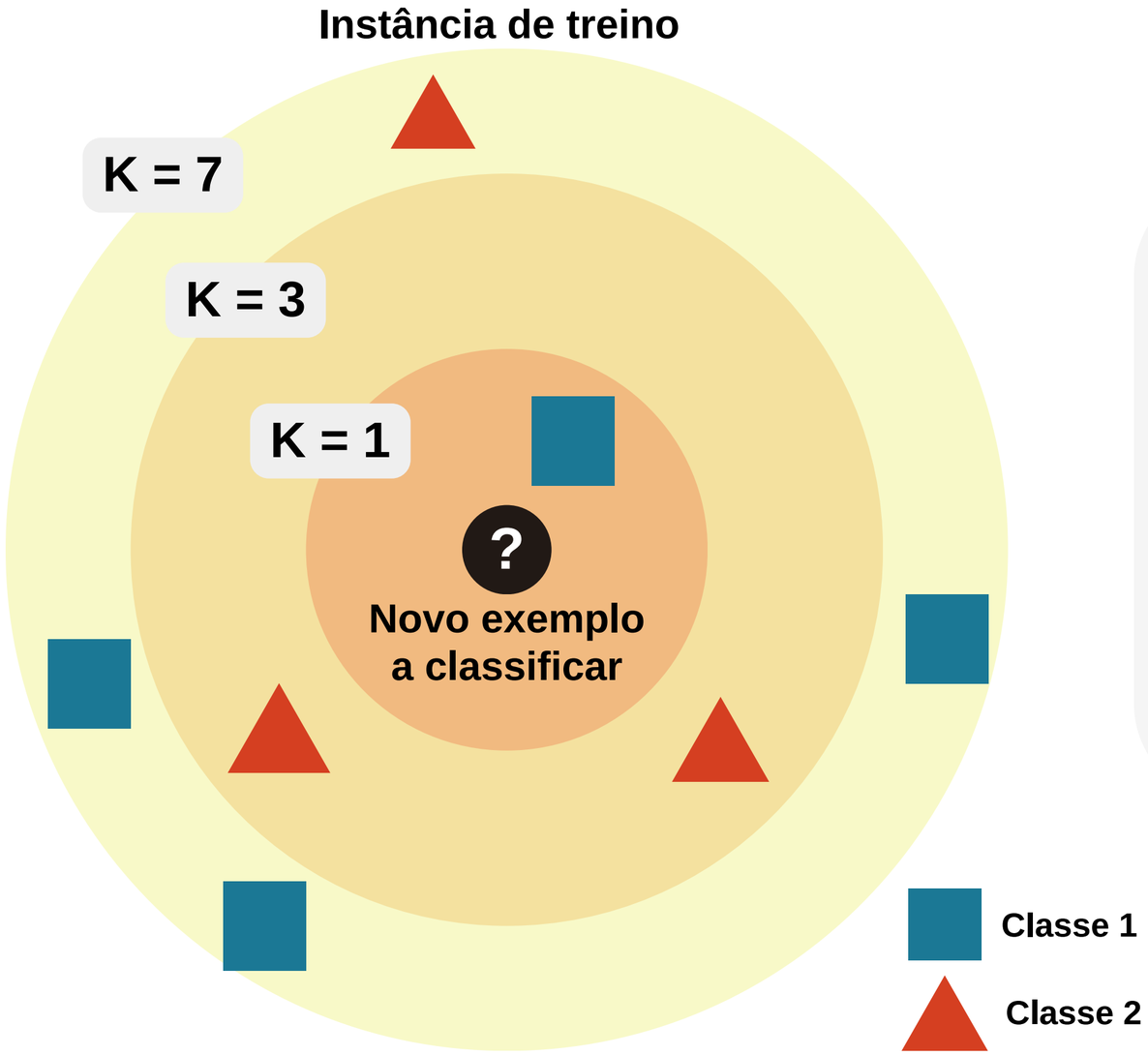
K-nearest neighbors algorithm

Aprendizagem Computacional I

Beatriz Sá up202105831
Francisco Ribeiro up202104797
Marta Pereira up202105713

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

KNN - K Nearest Neighbor



- Classifica novos pontos de dados com base na proximidade com os pontos existentes no espaço de atributos.
- Determina os K pontos de dados mais próximos ao novo ponto que está a ser classificado.
- Classifica o novo ponto com base na classe mais comum entre os seus vizinhos mais próximos.

KNN - K Nearest Neighbor

Vantagens:

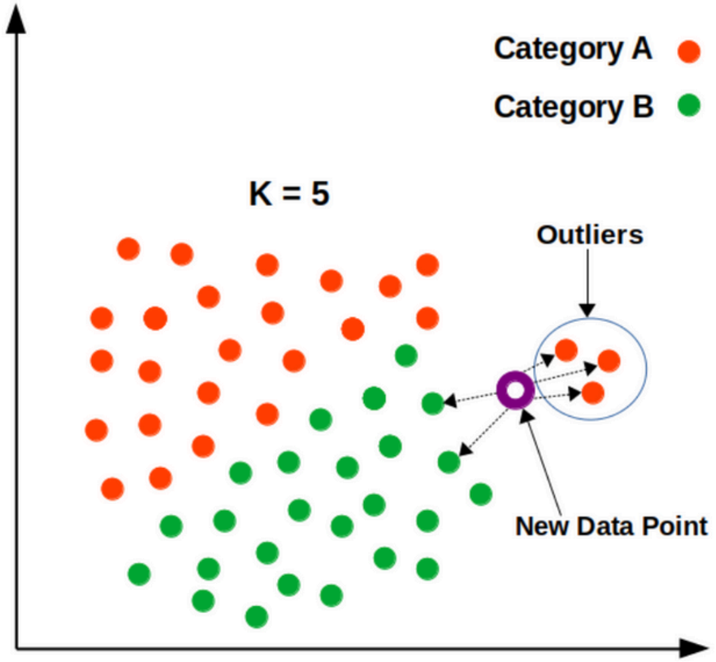
- Simples e intuitivo;
- Boa capacidade preditiva em vários problemas;
- Inerentemente incremental: se novos objetos são incorporados ao conjunto de treino, eles serão automaticamente considerados quando o algoritmo precisar prever a classe de um novo objeto.

Desvantagens:

- Tempo longo necessário para classificar novos objetos;
- Utiliza apenas informações locais para classificar novos objetos;
- Sensível à presença de atributos irrelevantes;
- Sensível à presença de outliers;
- Sem modelo, portanto, sem interpretação possível.

Comportamento do KNN perante outliers

- Outliers podem distorcer a medida de distância, levando a uma classificação incorreta ou imprecisa.
- Podem afetar a escolha dos vizinhos mais próximos e influenciar o resultado final do modelo.



Mudanças efetuados no KNN

Deteção de ouliers

Atribuição de um peso, dado pelo Local Outlier Factor ou pelo Isolation Forest, durante a fase de treino;

A seguinte expressão visa normalizar, de 0 a 1 em que 0 é mais provável de ser outlier, os dados de acordo com um intervalo que varia desde o mínimo e o máximo valor da aplicação do LOF ou do Isolation Forest;

$$\text{self.weight} = \frac{\text{weight} - \min(\text{weight})}{\max(\text{weight}) - \min(\text{weight})}$$

***weight*:** peso atribuído pelo LOF ou pelo Isolation Forest

Proximidade dos pontos vizinhos

De acordo com a proximidade dos dados ao novo ponto, é lhes atribuído um valor de maior ou menor importância;

$$weight[t] = \sum_{i=0}^{size-1} \left(\frac{size - i}{size} \right) w$$

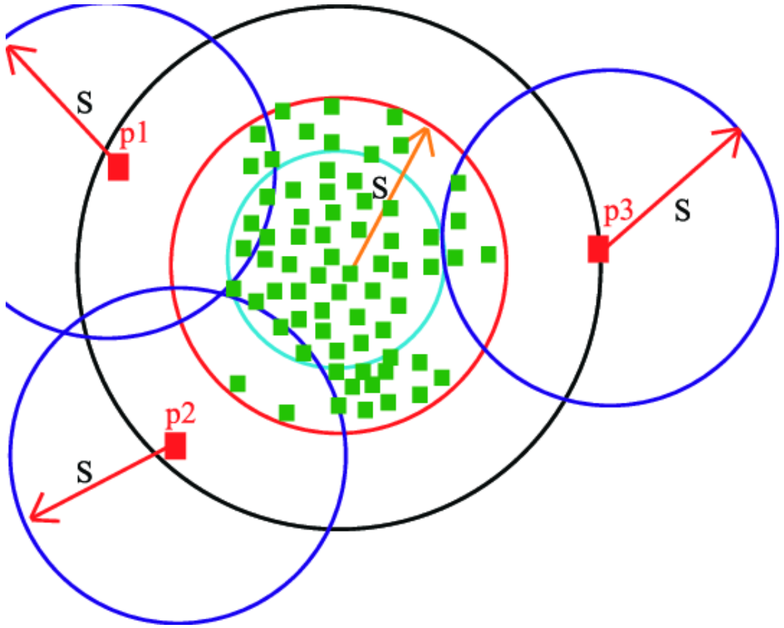
$$(t, w) \in N$$

***size*:** inicialmente igual ao tamanho do conjunto.

***weight*:** dicionário que armazena pesos acumulados para cada alvo.

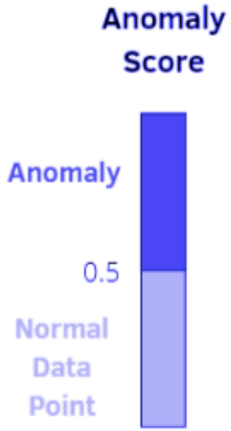
N: lista de pares (target,weight).

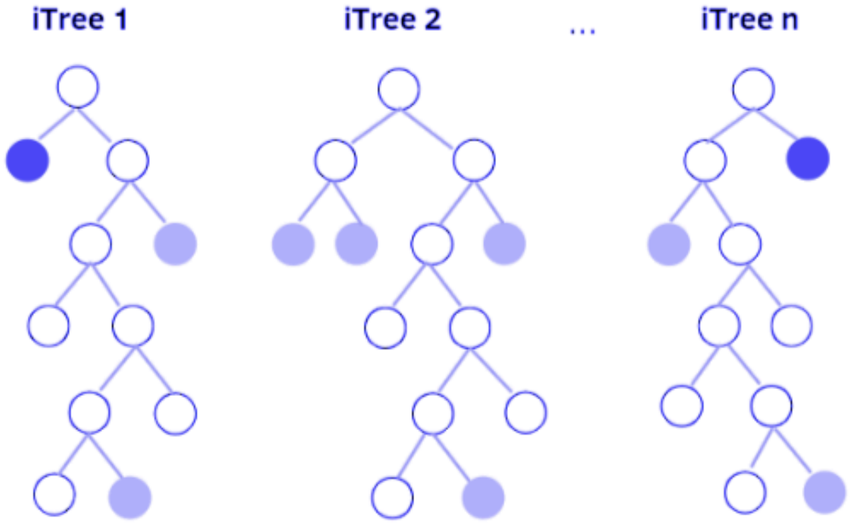
Métodos não supervisionados para deteção de outliers



LOF

Local Outlier Factor





Isolation Forest

Local Outlier Factor (LOF)

Conceito

É um algoritmo de deteção de outliers que mede a densidade local de cada objeto em relação aos seus vizinhos.

Funcionamento

O LOF calcula um valor de anomalia para cada objeto, com base na densidade relativa da sua vizinhança.

Objetos com baixo LOF são considerados outliers.

Vantagens

O LOF pode ajudar o KNN a identificar e lidar melhor com outliers, melhorando a sua precisão e robustez.

Isolation Forest

1

Conceito

Baseia-se no princípio de que anomalias são pontos que são muito poucos e diferentes em relação à maioria dos outros pontos nos dados.

2

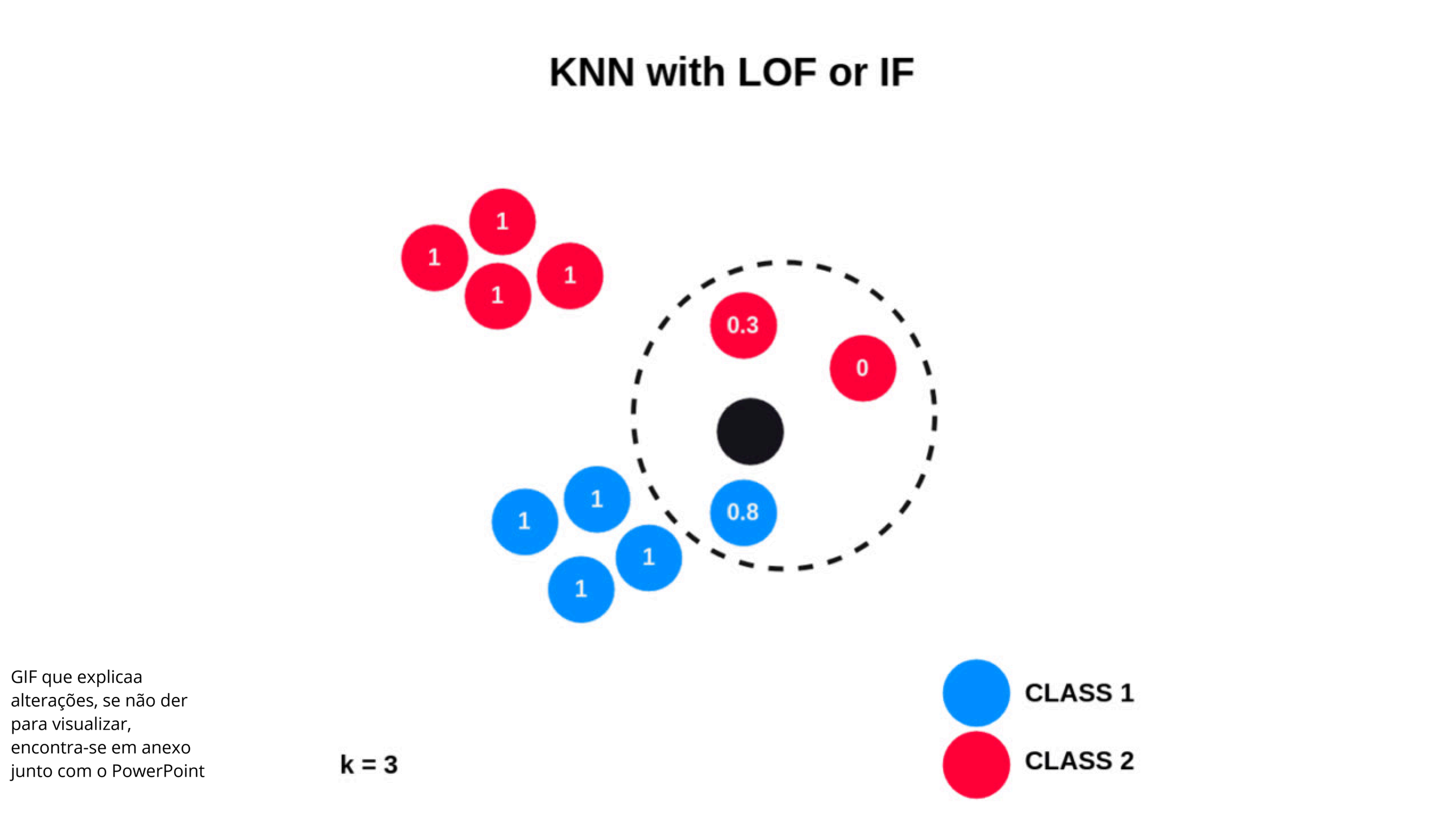
Funcionamento

O algoritmo constrói árvores de isolamento aleatórias e mede a facilidade com que cada objeto pode ser isolado, identificando outliers com base nessa métrica.

3

Vantagens

Assim como o LOF, o Isolation Forest pode ser usado em conjunto com o KNN para melhorar a sua capacidade de lidar com dados atípicos, outliers.



GIF que explicaa alterações, se não der para visualizar, encontra-se em anexo junto com o PowerPoint

Como o LOF e o Isolation Forest podem melhorar o KNN

Ponderação de vizinhos

As pontuações de anomalia geradas pelo LOF e Isolation Forest são usadas para ponderar a contribuição de cada vizinho no processo de classificação do KNN.

Código

```
anomaly_detector = self.outlier_detector # pode ser LOF ou Isolation Forest
anomaly_detector.fit(X)

if hasattr(anomaly_detector, 'negative_outlier_factor_'): # verificar se é LOF
    weight = anomaly_detector.negative_outlier_factor_
else: # Se for Isolation Forest
    weight = anomaly_detector.score_samples(X)
```

Setup experimental:

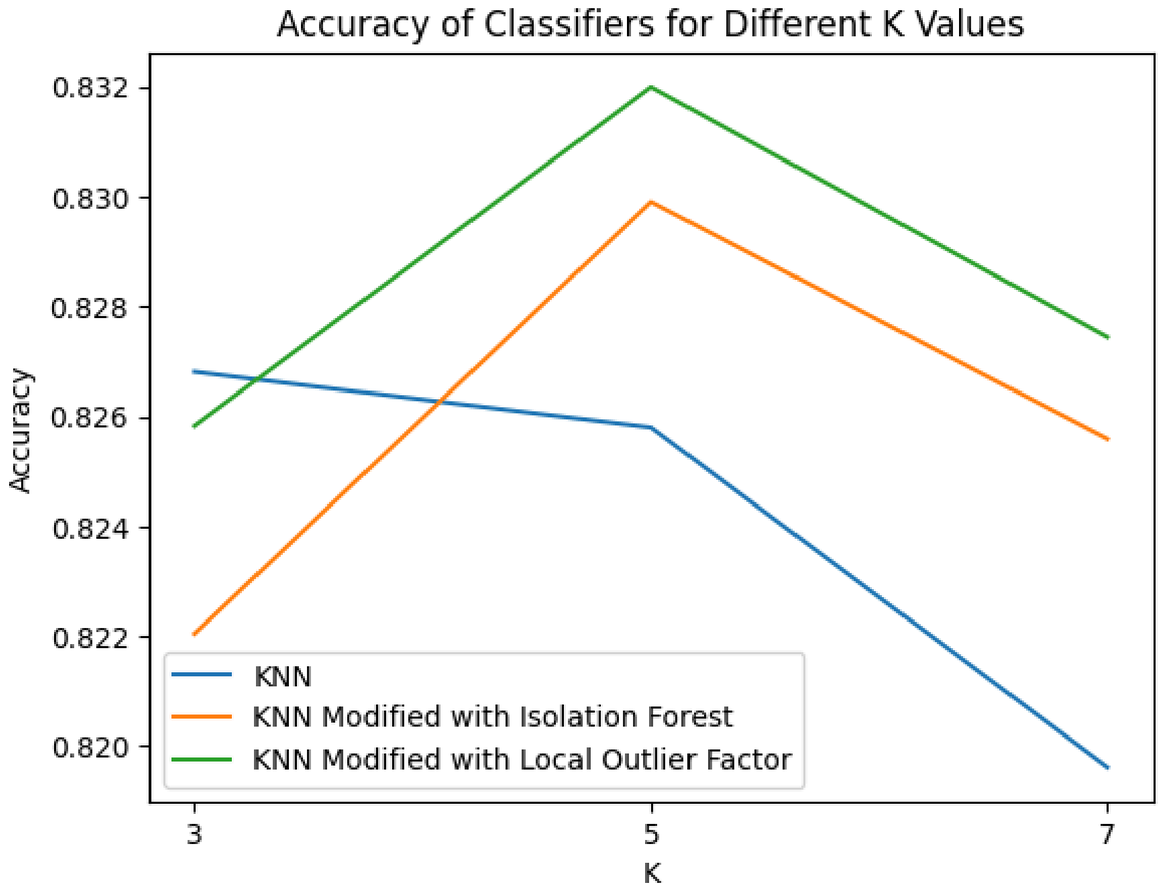
Dataset e as suas características

Dataset	Tipo de dados dos targets	Accuracy, KNN original
Diabetes	categóricos	K= 7, 72.13 %
Iris	categóricos	K = 5, 96.00 %
Wilt	numéricos	K = 5, 98.00%
Spambase	numéricos	K = 5, 80.92%
One-hundred-plants-margin	numéricos	K = 7, 76.75 %

Dataset	Tipo de dados dos targets	Accuracy, KNN original
analcatdata_authorship	categóricos	K = 7, 99.17 %
Wine Quality	numéricos	K = 5, 72.42 %
Vehicle	categóricos	K = 5, 64.31 %
Blood-transfusion-service-center	numéricos	K = 5, 76.20 %
Ionosphere	categóricos	K = 5, 84.34%

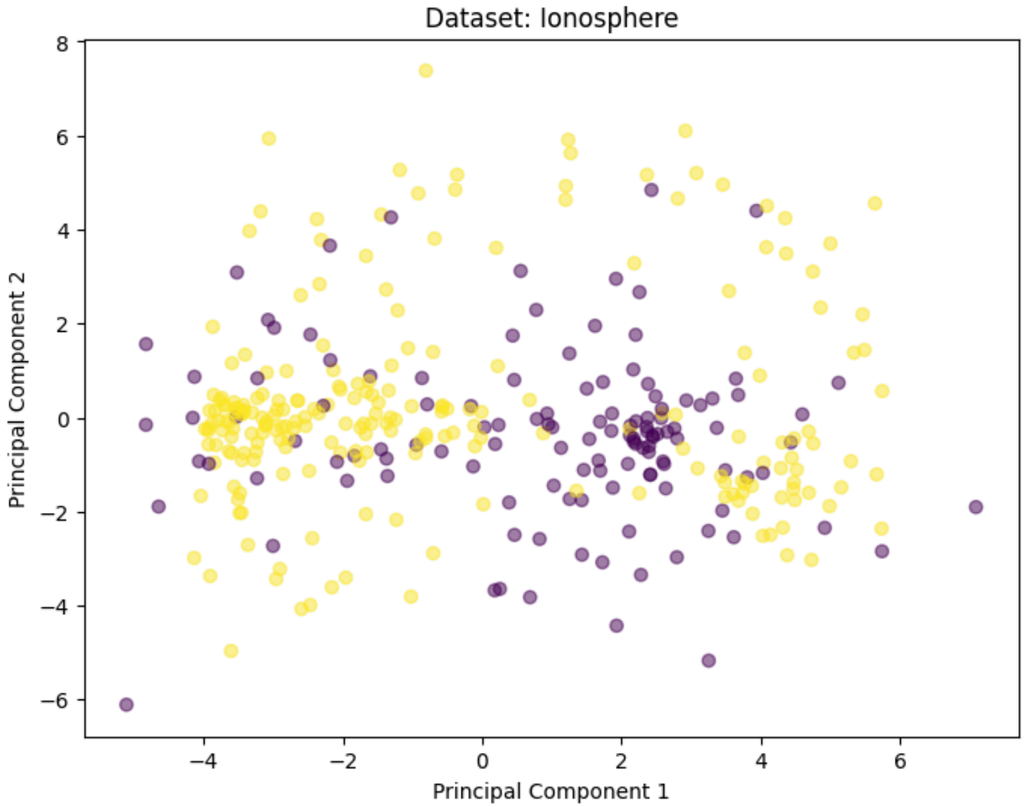
Setup experimental:

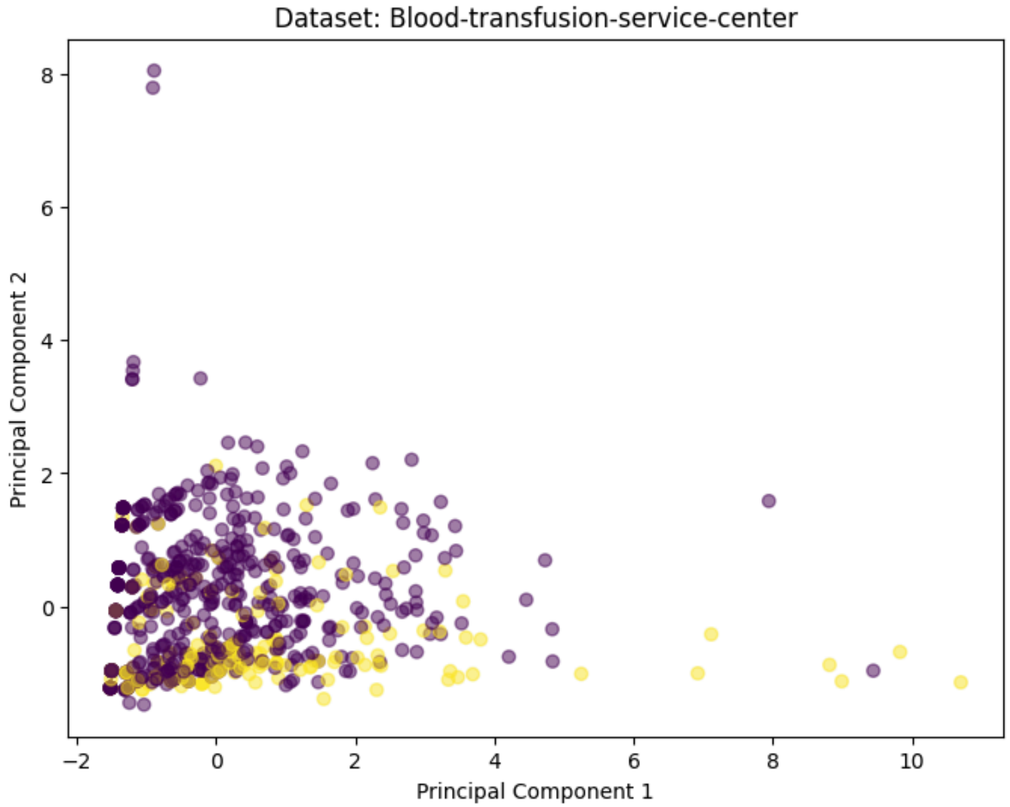
Hiperparâmetros



Setup experimental:

Análise de componentes principais (PCA)





Setup experimental:

Pré-processamento de dados

Dataset: Diabets

Antes do pré-processamento dos dados:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive

Depois do pré-processamento dos dados:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```python
def preprocess_data(X, y):
    # Convert categorical variables to dummy variables
    X = pd.get_dummies(X, drop_first=True)

    # Convert bool columns to int
    X = X.astype({col: 'int' for col in X.select_dtypes(['bool']).columns})

    # Check if y is of type 'numeric'
    if pd.api.types.is_numeric_dtype(y):
        if is_continuous(y):
            y = y.astype('category')
            label_map = {label: idx for idx, label in enumerate(y.cat.categories)}
            y_numeric = y.map(label_map)
        else:
            y_numeric = y

    # Type categorical
    else:
        try:
            y_numeric =pd.to_numeric(y)
        except ValueError:
            label_map = {label: idx for idx, label in enumerate(y.cat.categories)}
            y_numeric = y.map(label_map)

    return X, y_numeric
```

Setup experimental:

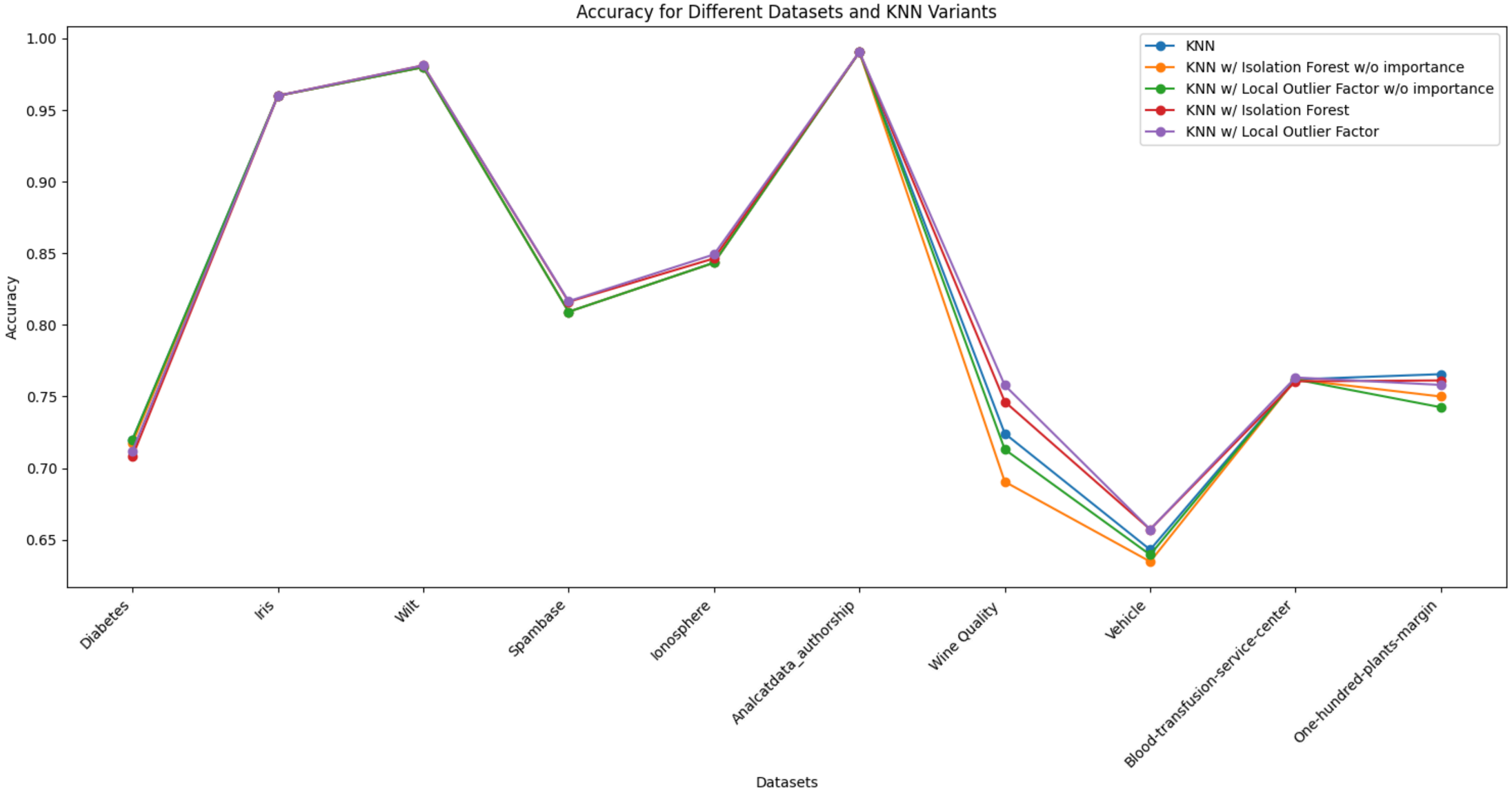
Método de avaliação dos resultados

K folds, Cross-validation:

- Todos os dados são usados tanto para treino quanto para validação, mas em diferentes iterações.
- Ajuda a garantir que o modelo generaliza bem para dados não vistos.
- Optamos por, perante as precisões retornadas, calcular a média entre elas e basear-nos nesse valor para avaliar os classificadores.

```
kf = KFold(n_splits=10, shuffle=True, random_state=1)
def perform_cross_validation(classifier, X, y):
    scores = cross_val_score(classifier, X, y, cv=kf, scoring='accuracy')
    return np.mean(scores)
```

Resultados


Accuracy for Different Datasets and KNN Variants
KNN
KNN w/ Isolation Forest w/o importance
KNN w/ Local Outlier Factor w/o importance
KNN w/ Isolation Forest
KNN w/ Local Outlier Factor
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
Diabetes
Iris
Wilt
Spambase
Ionosphere
Analcatdata_authorship
Wine Quality
Vehicle
Blood-transfusion-service-center
One-hundred-plants-margin
Datasets

Discussão final

- Usar o LOF e o Isolation Forest para atribuir pesos permite que pontos mais confiáveis tenham maior influência na decisão final;
- Ao reduzir a influência dos outliers, o KNN pode focar nos pontos mais representativos e evitar ser influenciado por dados anômalos;

Conclusão

As mudanças propostas ao algoritmo KNN visam melhorar a sua eficácia, especificamente em datasets com presença de outliers. Utilizando métodos como o LOF e o Isolation Forest, conseguimos atribuir pesos que resolvem problemas relacionados à sensibilidade a atributos irrelevantes e a outliers.

Assim, concluímos que perante datasets com essas especificações, optar pela nossa alternativa apresentada do KNN, seria uma mais valia para obter melhor precisão na classificação de novos pontos.