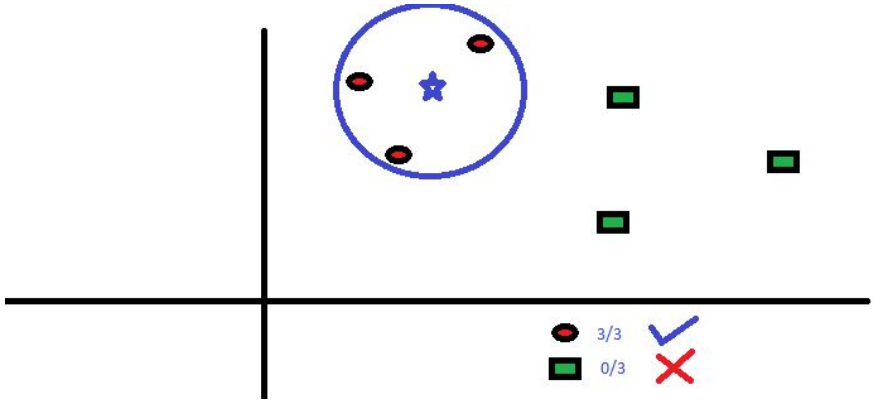
Premissas Algoritmos de ML

Classificadores

KNN

- Os dados estão no espaço de características, o que significa que os dados no espaço de características podem ser medidos por métricas de distância, como Manhattan, Euclidiana, etc.
- Cada um dos pontos de dados de treinamento consiste em um conjunto de vetores e um rótulo de classe associado a cada vetor.
- Desejado ter 'K' como um número ímpar no caso de classificação de 2 classes.



KNN

Prós:

- Fácil de entender, implementar e explicar.
- É um algoritmo não paramétrico, portanto, não possui suposições estritas.
- Nenhuma etapa de treinamento é necessária. Ele usa dados de treinamento em tempo de execução para fazer previsões, tornando-o mais rápido do que todos os algoritmos que precisam ser treinados.
- Como não precisa de treinamento, os pontos de dados podem ser facilmente adicionados.

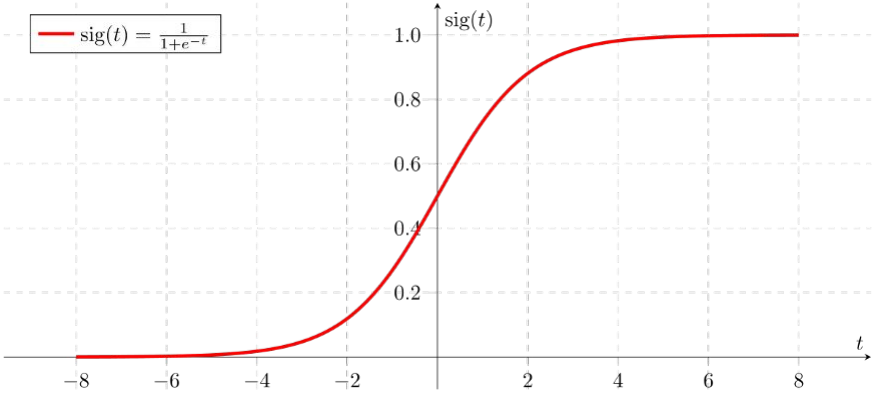
Contras:

- Ineficiente e lento quando o conjunto de dados é grande. Quanto ao custo do cálculo, a distância entre o novo ponto e os pontos do treinamento é alta.
- Não funciona bem com dados de alta dimensão porque fica mais difícil encontrar a distância em dimensões mais altas.
- Sensível a outliers, pois é facilmente afetado por outliers.
- Não pode funcionar quando faltam dados. Portanto, os dados precisam ser imputados manualmente para que funcionem.
- Precisa de dimensionamento/normalização de recursos.

Regressão Logística

Suposições:

- Assume-se que há mínima ou nenhuma multicolinearidade entre as variáveis independentes.
- Geralmente requer um grande tamanho de amostra para prever corretamente.
- Ele assume que as observações são independentes umas das outras.
- Sofre com outliers



Regressão Logística

Prós:

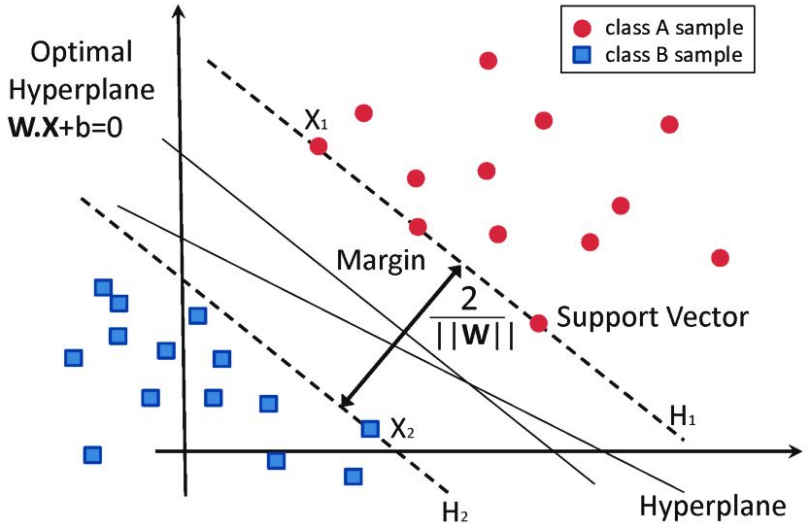
- Fácil de interpretar, implementar e treinar. Não requer muito poder computacional.
- Não faz nenhuma suposição da distribuição de classe.
- Rápido na classificação de registros desconhecidos.
- Pode acomodar facilmente novos pontos de dados.
- É muito eficiente quando os recursos são linearmente separáveis.

Contras:

- Tenta prever resultados probabilísticos precisos, o que leva ao overfitting em altas dimensões.
- Como possui uma superfície de decisão linear, não pode resolver problemas não lineares.
- Difícil obter relações complexas que não sejam relações lineares.
- Requer muito pouca ou nenhuma multicolinearidade.
- Precisa de um grande conjunto de dados e exemplos de treinamento suficientes para todas as categorias para fazer previsões corretas.

SVM

Ele assume que os dados são independentes e distribuídos de forma idêntica.



SVM

Prós:

- Funciona muito bem em dados de alta dimensão.
- Memória eficiente.
- Eficaz nos casos em que o número de dimensões é maior que o número de amostras.

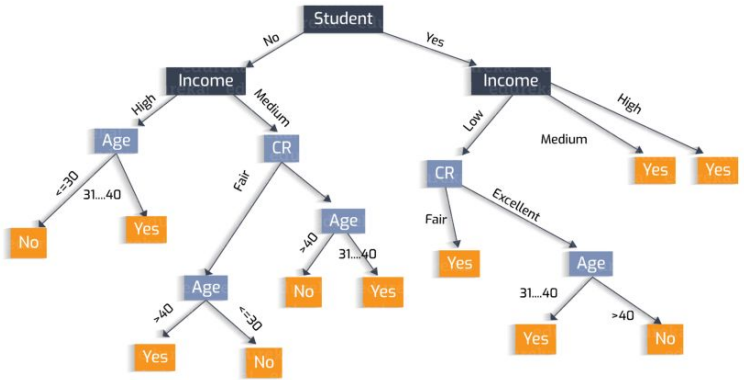
Contras:

- Não é adequado para grandes conjuntos de dados.
- Não funciona bem quando o conjunto de dados tem ruído, ou seja, as classes de destino estão sobrepostas.
- Demora para treinar.
- Nenhuma explicação probabilística para a classificação.

Árvores de Decisão

Inicialmente, todos os dados de treinamento são considerados como raiz.

Os registros são distribuídos recursivamente com base no valor do atributo.



Árvores de Decisão

Prós:

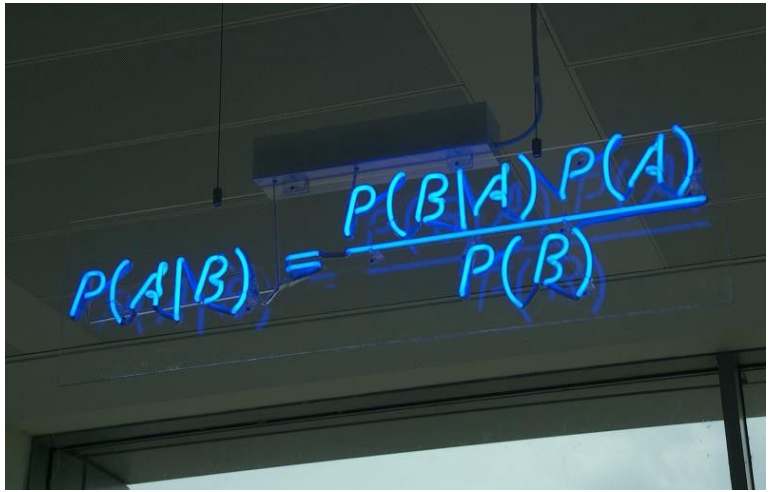
- Em comparação com outros algoritmos, a preparação de dados requer menos tempo.
- Não requer que os dados sejam normalizados.
- Valores ausentes, até certo ponto, não afetam muito seu desempenho.
- É muito intuitivo, como pode ser explicado como condições if-else.

Contras:

- Precisa de muito tempo para treinar o modelo.
- Uma pequena mudança nos dados pode causar uma mudança consideravelmente grande na estrutura da Árvore de Decisão.
- Comparativamente caro para treinar.
- Não é bom para tarefas de regressão.

Naive Bayes

A maior e única suposição é a suposição de independência condicional.



Naive Bayes

Prós:

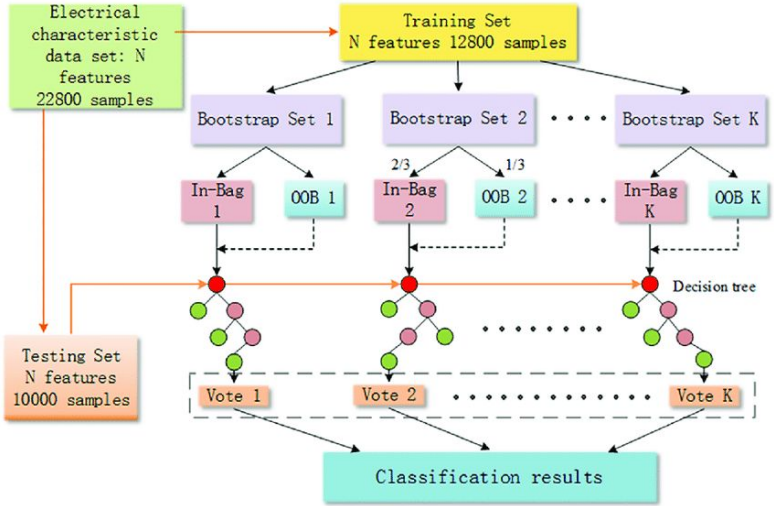
- Fornece alto desempenho quando a suposição de independência condicional é satisfeita.
- Fácil de implementar porque apenas probabilidades precisam ser calculadas.
- Funciona bem com dados de alta dimensão, como texto.
- Rápido para previsões em tempo real.

Contras:

- Se a independência condicional não se mantiver, o desempenho será ruim.
- Tem o problema de Estabilidade Numérica ou Underflow Numérico por causa da multiplicação de vários dígitos pequenos.

Random Forest

Assunção de não distribuições formais. Sendo um modelo não paramétrico, ele pode lidar com dados distorcidos e multimodais.



Random Forest

Prós:

- Robusto para outliers.
- Funciona bem para dados não lineares.
- Baixo risco de overfitting.
- Funciona com eficiência em grandes conjuntos de dados.

Contras:

- Treinamento lento.
- Tendência ao lidar com variáveis categóricas.