Relatório

Trabalho de Inteligência Artificial (INF1771)

Guilherme Dantas - 1713155
Nagib Suaid - 1710839

Introdução

Selecionamos a base de dados **Flags**[1] do repositório de *Machine Learning* da UCI (University of California, Irvine), que consiste em dados a respeito de todos os países reconhecidos pela ONU (Organização das Nações Unidas), 194 no total. Não há um rótulo pré-estabelecido, contudo optamos pela sugestão de reconhecer, a partir dos outros atributos, o atributo “religion”, que diz respeito da religião oficial ou predominante de um país.

Não esperamos que obteremos uma taxa de reconhecimento alta pois não existe uma relação causal entre esses atributos e a religião do país, mas sim alguns padrões notáveis, como a presença de uma lua crescente é um forte indicador de que no país é praticado o islamismo. Além disso, a base de dados é relativamente pequena, então religiões como o Hinduísmo ocorrem apenas duas vezes na base de dados e dificilmente serão reconhecidos.

Seleção dos atributos

Primeiramente, foi descartado o atributo “**name**”, que é o nome do país. Isto porque facilmente pode-se ver que não existe uma correlação entre o nome de um país com sua religião. Por exemplo, “Índia” e “Guiana” são países predominantemente hindus, contudo a “Guiana Francesa” é predominantemente católica. Mesmo assim, a palavra “Guiana” é mais parecido com “Guiana Francesa” do que com “Índia”.

Todos os atributos *a priori* são passíveis de serem usados. Isto porque algoritmos e métricas de proximidade determinam conjuntos de atributos “ótimos” diferentes. Contudo, para cada algoritmo e configuração de parâmetros, foi encontrado iterativamente um conjunto de atributos que aumentaria (localmente) a acurácia para aquele algoritmo e configuração de parâmetros. O algoritmo por trás desta seleção é bastante simples:

1. **É medida a acurácia usando todos os atributos**
2. **É medida a acurácia sem um atributo que estava sendo usado**
3. **Realiza o passo 2 para todos os atributos usados até o instante e seleciona o atributo que, ao ser ignorado, melhora mais significativamente a acurácia.**
4. **Caso haja tal atributo, é realizado o passo 2 agora sem levar em conta o atributo selecionado. Caso contrário, o algoritmo**

[1] https://archive.ics.uci.edu/ml/datasets/Flags

```
cessa.
```

Variação dos conjuntos de treinamento e de teste

Foi usado o algoritmo de *cross-validation* **k-fold** para validar os algoritmos implementados. Assim, os conjuntos de treinamento e de teste são escolhidos dentre k subconjuntos (de tamanho similar) do conjunto de exemplos. Na literatura, valores para este parâmetro variam entre 5 e 10, então foi optado pelo maior valor neste intervalo, de modo a ter uma boa diversidade de dados de treinamento. Por isso, foi estabelecido como o valor de k sendo **10**. Não alteramos artificialmente esse valor, pois alteraria a acurácia do algoritmo.

Variação dos parâmetro dos algoritmos

Foram implementados dois algoritmos: **KNN** e **Árvores de Decisão**.

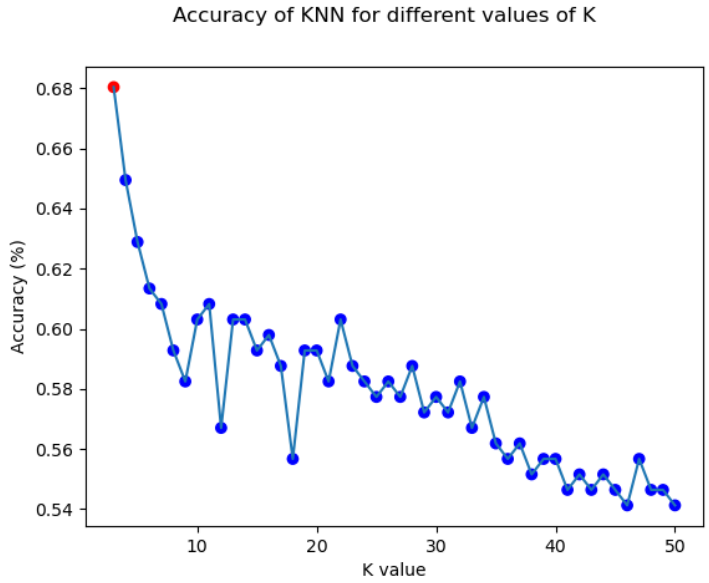
KNN (K-Nearest Neighbour)

Os parâmetros ajustáveis deste algoritmo são (1) k, o tamanho da vizinhança e (2) a métrica de proximidade.

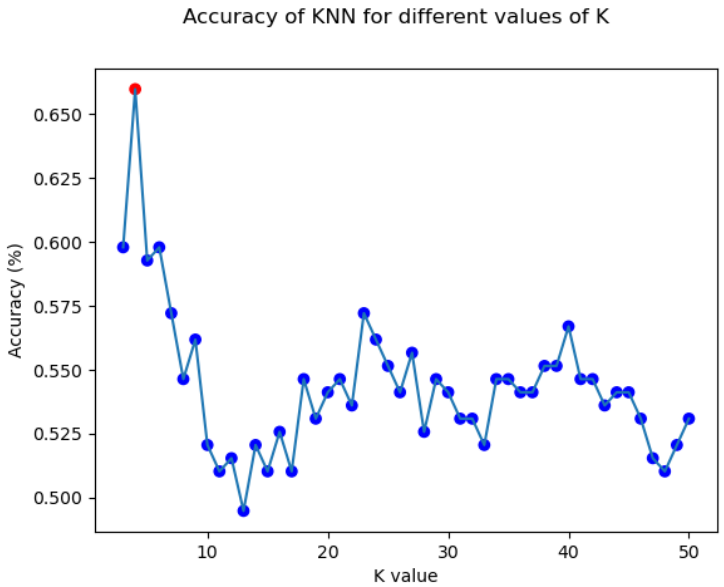
O tamanho da vizinhança, k, foi obtido de forma a maximizar a acurácia do algoritmo. O valor mínimo escolhido foi de 3, pois é o menor ímpar diferente de 1 (que seria muito suscetível a *outliers*). E o valor máximo escolhido foi de 50, pois como existem religiões que não possuem sequer 2 países, e uma vizinhança muito grande implicaria em nunca reconhecer estas religiões muito “raras” no banco de exemplos.

A métrica de proximidade também foi selecionada por resultar numa acurácia maior. As métricas implementadas foram a distância euclideana e a distância de Hamming. Percebeu-se que a distância de Hamming obteve uma acurácia, 2,0% maior, para o melhor valor de k no intervalo de 3 a 50.

Usando a distância de Hamming como métrica de proximidade, obtemos o seguinte gráfico que relaciona a acurácia do modelo com o valor de K. O ponto em vermelho indica o valor máximo de acurácia de 68,0% para k = 3.



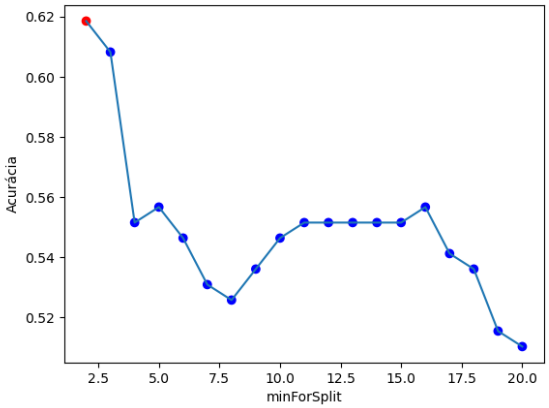
Usando a distância euclideana como métrica de proximidade, obtemos o seguinte gráfico que relaciona a acurácia do modelo com o valor de K. O ponto em vermelho indica o valor máximo de acurácia de 66,0% para k = 4.

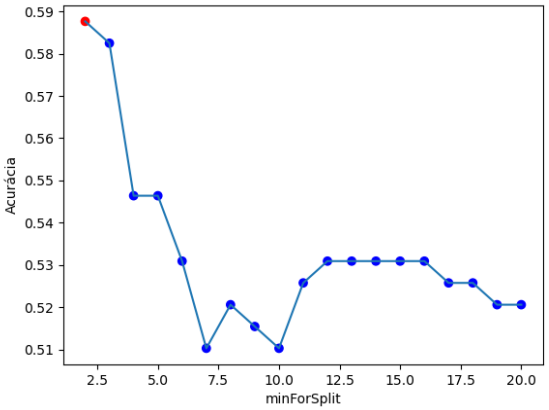


Árvores de Decisão

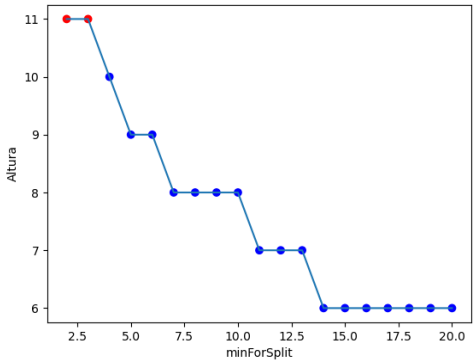
Os parâmetros ajustáveis deste algoritmo são (1) a altura máxima da árvore, (2) a métrica de impureza dos nós e (3) o número mínimo de instâncias necessárias para dividir um nó.

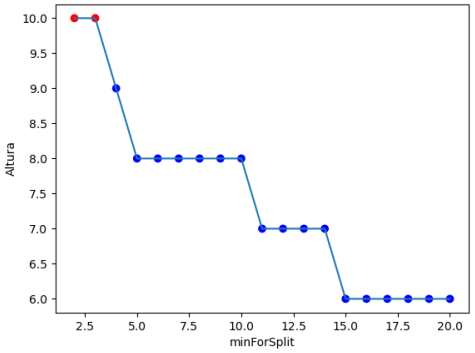
O parâmetro 3 foi escolhido fazendo uma análise do seu efeito na acurácia quando varia entre 2 e 20, teste que foi feito com ambas as métricas(2) para comparação e com altura máxima(1) muito grande para não ser um fator. Observou-se que em ambos os casos o valor mínimo, foi o ótimo e este foi o valor usado.





Mesmo depois de escolher esse valor, foi observado também o impacto que esse parâmetro tinha no tamanho das árvores geradas e com os resultados obtidos foi decidido não deixar a altura máxima interromper o crescimento da árvore e deixamos esse parâmetro em 20





Algoritmos implementados

O parsing do arquivo de dados, assim como a definição de cada instância foi feita em **C++**. Com ajuda da biblioteca ***pybind11***[2], foi feito o binding desta biblioteca para **Python**, de modo a ter um ambiente mais flexível para implementação dos algoritmos, mas ainda se beneficiando da performance de uma linguagem de baixo nível.

Ambos os algoritmos estão implementados como classes que herdam de uma classe abstrata “Algorithm”, que possui dois métodos “virtuais”: **fit**, que treina o modelo com dados de treinamento, e **predict**, que prediz os rótulos para novos dados de teste. Além disso, ambos os algoritmos são validados por uma implementação única de **k-fold**.

[2] https://github.com/pybind/pybind11

KNN (K-Nearest Neighbour)

A função de fit simplesmente armazena os dados de teste. Por isso, podemos considerar o tempo de treinamento irrisório, já que não há uma cópia, mas sim uma passagem de referência em Python.

A função de predição cria uma matriz de distâncias aonde o elemento ij representa a distância entre a i-ésima instância de teste e a j-ésima instância de treinamento. Assim, é obtido o índice das k instâncias de treinamento mais próximas de uma dada instância de treinamento. Assim, é obtido o rótulo dessas k instâncias e desses, o rótulo mais comum. O tempo de execução, contudo, depende da métrica de proximidade. Para distância euclideana, a função de predição leva em média 10ms para ser executada. Já para a distância de Hamming, a função de predição leva em média 308μs. Não é de se surpreender, pois a função realiza menos operações aritméticas.

Além disso, observou-se que para a distância de Hamming, a acurácia foi de 68,0% e para a distância euclideana foi de 66,0%, como foi visto já.

Árvores de Decisão

A função fit é o que desencadeia a construção da árvore de decisão. Trata-se de um processo recursivo onde cada nó decide como melhor particionar/classificar o seu conjunto de dados e para cada partição criada se chama o mesmo procedimento recursivamente, onde cada uma dessas novas árvores criadas vai ser um filho do nó original.

Por ser um processo recursivo que precisa iterar sobre todo o seu conjunto de dados para cada atributo diferente(mesmo que atributos categóricos que já foram escolhidos por algum ancestral possam ser pulados), é de se esperar que esse processo seja relativamente lento, e sensível tanto ao número de atributos quanto ao de instâncias de treino.

Os cálculos da entropia e impureza de Gini são similares, e não constituem muito do tempo de execução, porém ao compará-los observa-se uma significativa diferença tanto em acurácia quanto tempo. Isso provavelmente se dá pela baixa correlação entre os atributos e as classificações deste dataset: utilizando a entropia o algoritmo de seleção de atributos pode remover mais features “irrelevantes”, melhorando tanto a acurácia (61.85% contra 53.61%) quanto o tempo de fit (4.26ms contra 5.17ms) quando comparados com o uso da impureza de Gini.

Para a classificação, basta fazer um percurso pela árvore, começando pela raiz e sempre descendo para o filho baseado no atributo que o nó corrente usou para sua subdivisão. Esse processo tem complexidade linear na altura da árvore, o que se compara bem favoravelmente com a classificação do KNN, que é linear no tamanho da base de treino. Os tempos obtidos para as medidas de impureza entropía e impureza de Gini respectivamente foram 0.67ms e 0.94ms.

Resultados finais

KNN (K-Nearest Neighbour)

	Distância de Hamming	**Distância euclideana**
Atributos	Todos exceto “name”, “religion”, “colours”, “mainhue”, “saltires”, “icon” e “text”.	Todos exceto “name”, “population”, “religion”, “red”, “crosses”, “sunstars”, “crescent”, “animate” e “text”
Tempo gasto no processo de treinamento (em média, por instância)	7.1 ns (apenas armazena dados de treinamento)	
Tempo gasto no processo de classificação (em média, por exemplo desconhecido)	30.8 μs	1 ms
Taxa de reconhecimento	68,0%	66,0%

Árvores de Decisão

	Índice de Gini	**Entropia**
Atributos	Todos exceto “name”, “religion”,'area', 'population','blue'	Todos exceto “name”, “religion”, 'zone','area', 'population','stripes' 'mainhue','topleft'
Tempo gasto no processo de treinamento (em média, por instância)	5.17ms	4.26ms
Tempo gasto no processo de classificação (em média, por exemplo desconhecido)	0.94ms	0.67ms
Taxa de reconhecimento	53.61%	61.85%