Projeto Final de Aprendizagem de Máquina - Analise de Dataset para a Classificação de Frutos de Tâmara

José Cleomon da Silva Junior
cleomon@alu.ufc.br
521239

Matheus do Vale Almeida
matheus.almeida@alu.ufc.br
473219

Samyra Vitória Lima de Almeida
samyraalmeida@alu.ufc.br
521240

Resumo

Por meio do uso de três modelos classificatórios de aprendizagem de máquina, K-Nearest Neighbor (KNN), Árvores de Decisão e Redes Neurais Artificiais, buscou-se analisar os modelos para realizar a atividade de classificação de frutos de tâmara por suas variedades genéticas dados diversos atributos físicos de entrada. Em seguida, utilizando técnicas de redução de dimensionalidade no conjunto de dados, foram realizados experimentos com o intuito de investigar com se dava o comportamento do melhor modelo obtido, quando o mesmo era treinado utilizando menos atributos sobre os padrões. Para este propósito, 898 instâncias de sete tipos diferentes de tâmaras com um total de 34 características, foram obtidas através de um sistema de visão computacional (CSV), e estes dados foram utilizados para compor o Dataset. Os resultados de acurácia alcançados com cada modelo foi de 90.5%, e 92.8%, respectivamente. Logo, após uma análise dos resultados foi verificado que o melhor modelo para a tarefa foi o de Redes Neurais.

1 Introdução

Em suma, com o objetivo de puro aprendizado, foram escolhidos três modelos para a tarefa de classificação de frutas de Tâmara. Sendo estas, frutas comestíveis e nutritivas com cerca de 200 tipos e mais de 2500 espécies[2-4]. Para fazer uma classificação bem sucedida, as semelhanças e diferenças entre classes devem ser tratadas com cautela. Portanto, os estudos de reconhecimento e classificação de frutos têm sido realizados com base nas características visuais extraídas das imagens[1].

Tais modelos escolhidos, K-Nearest Neighbor, Árvores de Decisão e Redes Neurais Artificiais, cada um apresenta suas particularidades, seja na ideia, no treinamento, ou até mesmo na análise dos resultados. Mediante suas diferenças, surge o objetivo de analisar qual o melhor modelo a se utilizar para o problema de classificação de tâmaras. Assim, o proposito advém da curiosidade de ver como cada modelo comporta-se com dados reais de um conjunto de frutas.

Além disso, sabemos que muitos fatores determinam o tipo de fruto e classificar seus tipos apenas observando esses fatores exige experiência[1]. Com isso em mente, surge a dúvida se o uso das 34 características presente no conjunto de dados a ser analisado são realmente todas necessárias, isto é, treinando o modelo que obteve melhor resultado com novos dados gerados através de experimentos e técnicas de redução de dimensionalidade, verificar se o mesmo ainda continuaria com resultados agradáveis.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Características dependendo da aparência externa usada no estudo

Features			Main features
Subfeatures			
Area	Equivalent diameter		Morphological features
Perimeter	Solidity		
Major axis	Convex_area		
Minor axis	Extent		
Eccentricity	Aspect ratio		
Roundness	Compactness		
Shapefactor_1	Shapefactor_3		Shape features
Shapefactor_2	Shapefactor_4		
Mean RR	Mean RG	Mean RR	Color features
Std. dev RR	Std. dev RG	Std. dev RR	
Skew RR	Skew RG	Skew RR	
Kurtosis RR	Kurtosis RG	Kurtosis RR	
Entropy RR	Entropy RG	Entropy RR	
All Daub4 RR	All Daub4 RG	All Daub4 RR	

2 Fundamentação teórica

2.1 Descrição do Problema

Em vista do problema de analisar o melhor modelo para a classificação proposta, a principal métrica utilizada para fazer essa analise é a acurácia (Seção 3.3.1). Sendo esta, utilizada para comparar o desempenho entre os modelos. Ademais, também foram utilizadas matrizes de confusão (Seção 3.3.2) com o objetivo de analisar as predições feitas pelos modelos e com isso realizar conclusões sobre os dados.

2.2 Descrição dos Dados

O conjunto de dados a ser analisado possui um total de 898 instâncias de sete tipos diferentes de tâmaras a serem classificadas. Os tipos selecionados, com base nas informações compartilhadas do artigo no qual este se baseia, "Classification of Date Fruits into 304 Genetic Varieties Using Image Analysis", são: BERHI da região palestina, DEGLET da região da Argélia, SAFAVI, SOGAY, ROTANA e IRAQI da região de Riad e Medina da Arábia Saudita e DOKOL da região do Irã.

Cada tipo de fruto possui um total de 34 características, 12 para características morfológicas, 4 para suas dimensões de formas e outras 18 para feições de cor. Ainda sobre essas características, vale apresentar que cada fruto de tâmara foi examinado separadamente após as imagens obtidas serem convertidas em escala de cinza e imagens binárias para extração de características. Na qual, basicamente, as operações foram realizadas em metodologias de informação de threshold e pixel. Essas características e dados podem ser vistas melhor na Tabela 1.Salientando, todos os créditos na obtenção das medidas ao artigo base.

2.3 Artigo Base

Fazendo uma breve revisão do artigo base (Murat Koklu, 2021) que deu origem aos objetivos desse trabalho. Seu principal objetivo é de classificar os tipos de frutas de tâmara por meio de três métodos diferentes de aprendizagem de máquina. Para esse propósito, foi utilizado o mesmo conjunto de dados obtidos através de um CSV.

Seguindo, os modelos de classificação no artigo base foram desenvolvidos usando métodos de Regressão Logística e de Redes Neurais Artificiais. Obtendo resultados de desempenho de 91.0% e 92.2%, respectivamente. Então, com um modelo de combinação criado pela combinação dos outros dois modelos, o resultado de desempenho aumentou para 92.8%. Concluindo assim, que métodos de

aprendizado de máquina podem ser aplicados com sucesso para a classificação de tipos de frutas de tâmara. Em razão disso, esse trabalho busca alcançar resultados semelhantes ou melhores.

3 Metodologia

3.1 Modelos Escolhidos

Nos tópicos a seguir serão abordados brevemente as fundamentações teóricas de cada uma das ferramentas de aprendizagem de máquina utilizadas no presente trabalho.

3.1.1 KNN

Sendo um modelo de aprendizagem de máquina não-paramétrico o k-vizinhos mais próximos (do inglês, K-Nearest Neighbors - KNN), tem um conjunto ilimitado de parâmetros, ou seja, sua quantidade depende do número de padrões de treinamento. Como também sua complexidade ou flexibilidade cresce com mais dados.

Usado tanto para tarefas de classificação e regressão, suas predições são baseadas nas instâncias de treinamento mais próximas do padrão de teste, para isso, os dados de treinamento precisam ser armazenados para realizar as predições. Tal proximidade é relevante para o KNN, calculada por alguma função de distância (ou métrica) que mede até que ponto duas instâncias são próximas uma da outra:

- Distância **Euclidiana**:

$$\|x_i - x_j\|_2 = \sqrt{\sum_{d=1}^{D}(x_{id} - x_{jd})^2} \tag{1}$$

- Distância **Manhattan**:

$$\|x_i - x_j\|_1 = \sum_{d=1}^{D}|x_{id} - x_{jd}| \tag{2}$$

- Distância **Minkowski**:

$$\|x_i - x_j\|_p = \left(\sum_{d=1}^{D}|x_{id} - x_{jd}|^p\right)^{\frac{1}{p}}, p \geq 1 \tag{3}$$

- Distância **Mahalanobis**:

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T \Sigma^{-1}(x_i - x_j)}, \tag{4}$$

Em que Σ é a matriz de covariância dos dados de treinamento.

O dois principais parâmetros do modelo são o tamanho da vizinhança, K, e a métrica de distância, $d(x_i, x_*)$. A ideia do KNN para classificação consiste em considerar os k-vizinhos mais próximos através da seleção de alguma métrica. Para tal, considerando os K padrões $x_k, k \in \{1, ..., K\}$ mais próximos do padrão de teste x_*:

$$x_{KNN} = arg \min_{x_i \in \{x_1, ..., x_N\}} d(x_i, x_*) \tag{5}$$

O modelo classifica x_{KNN} pela classe que ocorrer com maior frequência entre os k-vizinhos.

Uma característica importante do modelo é o fato de seu aprendizado ser baseado "no quão similar" é um dado do outro. Com isso, é esperado que o modelo consiga agrupar os tipos de tâmaras sem muita dificuldade.

3.1.2 Arvore de Decisão

Assim como o modelo do KNN (Seção 3.1.1), árvores de decisão também são um modelo de aprendizagem de máquina não-paramétrico e são muito utilizadas para resolver problemas de classificação e regressão. E a sua complexidade também cresce conforme o número de dados vai aumentando.

Uma árvore de decisão é composta por nós que se relacionam hierarquicamente. Nela existe o nó raiz, o de maior nível hierárquico (o ponto de partida) e ligações para outros elementos que chamaremos de filhos. Onde esses filhos podem possuir seus filhos e assim sucessivamente. Estes tipos de nós funcionam como uma especie de banco de dados, além de possuírem alguma condição que será utilizada posteriormente para as tomadas de decisão. Nós que não possuem nenhum filho são chamados de nós folha (ou nós terminais) e armazenam os resultados finais, o que pode ser uma classe ou um valor como resposta.

Dada essa estrutura, em uma árvore de decisão, uma escolha é tomada através do caminho a ser percorrido da raiz até um nó folha. Deste modo, cada nó da árvore, exceto os nós folhas, servem para particionar o espaço em sub-regiões de maneira recursiva, funcionando como uma representação dessa divisão. A primeira divisão (representada pelo nó raiz) considera todos o conjunto de dados para encontrar o ponto de corte que maximiza a pureza das sub-regiões. E esse particionamento é repetido para as sub-regiões de maneria recursiva até que se chegue em um nó folha.

O ponto de corte é encontrado usando algum critério de impureza, neste caso podemos utilizar o Índice de Gini ou a Entropia para mensurar essa impureza e decidir o melhor ponto de divisão. Achar esse ponto pode ser uma tarefa computacional inviável, construir a árvore de decisão ótima é um problema NP-completo. Por isso, a árvore de decisão é construída de forma gulosa, realizando escolhas de forma a garantir o ótimo local e não o ótimo global. Segue a descrição e o modulo de calcular cada critério de impureza:

- Entropia: é a taxa de informação gerada por uma fonte de dados, onde dados improváveis fornecem mais informação. Quanto maior a pureza dos dados, menor a entropia, e ela é calculada por:

$$H = -\sum_{k} P(C_k) \log_2 P(C_k) \tag{6}$$

- Índice de Gini: é a frequencia em que um exemplo aleatório é incorretamente classificado e é calculado por:

$$G = 1 - \sum_{k} P(C_k)^2 \tag{7}$$

Cada divisão visa maximizar o Ganho de Informação, informação aprendida sobre os exemplos quando uma região é dividida em sub-regiões. Sendo R a região atual, R_e a sub-região da esquerda e R_d a sub-região da direita e I o indice de impureza da região. O ganho de informação pode ser quantificado por:

$$IG = I(R) - (P(R_e) \cdot I(R_e) + P(R_d) \cdot I(R_d)) \tag{8}$$

Nesse artigo realizaremos o treinamento com os seguintes algoritmos:

- ID3 (Iterative Dichotomizer): Um dos primeiros e mais simples algoritmos de árvore de decisão. Normalmente usa a entropia para escolher novas ramificações.
- C4.5: Versão mais avançada do algoritmo ID3, com suporte a poda e dados discretos, contínuos, faltantes.
- CART (Classification And Regression Tree): Similar ao algoritmo C4.5. Normalmente usa a impureza de Gini para escolher novas ramificações.

O modelo de árvore decisão foi escolhido por ser facilmente interpretável, pois cada exemplo é classificado seguindo um conjunto de regras de decisão. Também são facilmente escaláveis. Conseguem lidar bem com dados faltantes além de não precisarem de muita preparação dos dados. Além de realizarem a seleção automática de atributos importantes.

3.1.3 Redes Neurais

Modelos de redes neurais são modelos de aprendizado de máquina que utilizam combinações das entradas para gerar mapeamentos de regiões de classificação, que por sua vez podem ser combinadas novamente na geração de outras regiões mais complexas até que, utilizando funções de ativação não lineares, sejam capazes de gerar regiões de separação não lineares.

O problema de classificação em análise trata-se de uma classificação multiclasse, ou seja, na saída do modelo de rede neural é aplicada a função soft max, a qual possibilita a predição da probabilidade de um padrão pertencer a cada uma das classes, sendo a classe de maior probabilidade a escolhida pelo modelo.

Redes neurais são treinadas utilizando o algoritmo Backpropagation, o qual propaga os erros obtidos na camada de saída para as camadas mais internas até a camada de entrada. Diversos hiperparâmetros são utilizados para realizar o treinamento do modelo de rede neural, dentre eles, foram utilizados no treinamento do modelo do presente trabalho os seguintes: Taxa de aprendizado; Fator de Momentum, tamanho do Mini-Batch, quantidade de neurônios na camada oculta, Fator de regularização e Quantidade de Épocas Máxima.

A taxa de aprendizado especifica o quão agressiva é a aproximação que o modelo busca fazer daquele mínimo local. Já o fator de Momentum visa minimizar as oscilações até atingir esse mínimo local. O tamanho do Mini-Batch estipula a quantidade de elementos utilizados na amostragem para realizar a atualização dos pesos dos neurônios. O Fator de Regularização busca reduzir a possibilidade de overfitting por parte do modelo. A quantidade de épocas máxima também foi escolhido como um hiperparâmetro, pois em caso de não convergência do modelo, é necessário especificar o momento de parada do treinamento, além disso, em alguns casos, como na ocorrência de overfitting, continuar o treinamento pode acabar por reduzir a taxa de acerto do modelo.

O modelo de redes neurais foi utilizado como uma das possibilidades para solução do problema em análise principalmente por sua característica de separação de regiões de classificação não-lineares. Espera-se que esse modelo seja capaz de realizar separações de regiões complexas para as classes do problema, caso seja necessário.

3.2 Redução de Dimensionalidade

3.2.1 Score de Fisher

O problema em análise apresenta 34 atributos para classificar as instâncias em 7 classes diferentes. Observando a quantidade elevada de atributos, buscou-se verificar a possibilidade de redução desses atributos através da análise dos mesmo utilizando o score de Fisher. Essa métrica, para cada atributo, pode ser calculada da seguinte forma:

$$S_d = \frac{\sum_{k=1}^{K} N_k \left(\mu_{dk} - \mu_d\right)^2}{\sum_{k=1}^{K} N_k \sigma_{dk}^2} \tag{9}$$

Onde K é cada uma das classes, d é cada atributo, N_k é o total de elementos em cada classe, μ_{dk} é a média de cada atributo em cada classe, μ_d é média geral de cada atributo, independente de classe e σ_{dk} é a covariância de cada atributo em cada classe.

Essa expressão visa quantificar o quanto um determinado atributo varia entre as classes. Dessa forma, atributos que apresentarem um baixo Score de Fisher estarão contribuindo menos para a classificação, pois variam menos em diferentes classes. Essa métrica foi utilizada para realizar testes comparativos entre os modelos treinados com todos os atributos e com apenas os mais relevantes.

3.2.2 PCA

A análise das componentes principais (PCA) visa possibilitar a projeção de dados com muitas dimensões, como é o caso do problema em análise, em espaços de dimensões reduzidas. No presente trabalho, essa técnica foi utilizada para verificar como os dados ficavam dispostos no espaço de duas dimensões, o qual é possível ser analisado graficamente.

O PCA analisa as direções de maior variância dos dados para realizar a projeção, utilizando como matriz de projeção os autovetores correspondentes aos maiores valores de autovalores da matriz de covariância obtida dos dados. A quantidade de autovetores adotada especifica a dimensão da projeção dos dados da análise.

Após a projeção dos dados, com o objetivo de verificar a boa separação das classes no espaço projetado, foi utilizado o método de classificação KNN utilizando como parâmetro um único vizinho mais próximo e distância euclideana para o cálculo da distância. Em caso de acurácia elevada para essa aplicação do KNN, significava que as classes estavam sendo bem separadas.

3.3 Métricas Utilizadas

Na seção a seguir, serão descritos as principais métricas utilizadas para avaliação e comparação dos modelos.

3.3.1 Acurácia

A acurácia foi a principal métrica utilizada para comparar a qualidade entre os modelos, inclusive na própria validação cruzada para avaliar quais hiperparâmetros eram melhores para o modelo e o problema escolhido.

A acurácia é facilmente calculada relacionando a quantidade de acertos, contando todas as classes, com a quantidade total de elementos, como segue:

$$Acc = \frac{Acertos}{Erros + Acertos} \tag{10}$$

3.3.2 Matriz de Confusão

A Matriz de Confusão é uma visualização interessantes para avaliar quais classes estão sendo preditas com maior assertividade e quais classes estão sendo trocadas na predição em caso de erro. No eixo y do diagrama, são apresentadas as classes originais do padrões utilizados na análise, já no eixo x, estão presentes as classes que foram preditas pelos modelos. Os números na diagonal principal indica a quantidade de elementos de cada classe que foi predita corretamente, já os números fora da diagonal principal indicam a quantidade de elementos que foram preditos errados e a relação entre a classe verdadeira e a predita de forma incorreta. A visualização das matrizes de confusão foram utilizadas para verificar se as classes que estavam sendo preditas erradas eram sempre as mesmas, independente do modelo, o que poderia indicar uma proximidade muito grande entre dados de classes diferentes, ou se eram apenas erros por falta de precisão dos modelos.

4 Experimentos

Nesta seção serão abordados os principais aspectos utilizados para treinamento e análise dos modelos utilizados para o problema de classificação em análise.

4.1 Pré-processamento dos Dados

Para garantir que todos os modelos fossem submetidos aos mesmos dados, foi realizada a separação destes em uma etapa de pré-processamento. Nesta etapa os dados foram separados em dados para treinamento e dados para teste, em uma proporção de 80% para treino e 20% para teste.A quantidade total de dados disponíveis para serem utilizados na solução do problema foi de 898 instância, portanto, após a separação dos dados, 718 padrões de entrada ficaram disponíveis para o treinamento e 180 para a etapa de testes.

Como todos os modelos foram treinados utilizando validação cruzada, era necessário que todos recebessem os dados particionados da mesma forma, para treinamento e validação com dados idênticos, apenas modificando o modelo escolhido. Assim, as comparações entre as métricas de performance obtidas para cada modelo seriam justas e equivalentes. Sendo assim, durante a etapa de pré processamento, foram separadas as partições de dados que seriam utilizadas na validação cruzada durante o treinamento de cada um dos modelos. Os dados de treinamento foram separados em 10

Table 2: KNN - Resultados obtidos após treinamento

Parâmetros		
Distancia	K-vizinhos	Acurácia
Manhattan	26	$\sim$0.87992
Euclidiana	28	$\sim$0.87529

partições, considerando o total de 718 padrões disponíveis para o treinamento, buscando uma divisão equilibrada de dados nas partições, obtiveram-se 8 partições com 72 instâncias e 2 partições com 71.

Outros ajustes menores foram realizados nos dados para que os mesmo pudessem ser utilizados para treinamento, como a codificação de cada uma das classes de tâmaras em números e a disponibilização dessa informação para decodificação futura padrões classificados pelos modelos.

4.2 Treinamento dos Modelos

Cada um dos modelos adotados para solucionar o problema de classificação em análise apresenta suas peculiaridades no treinamento e análise dos resultados. A seguir serão apresentas as abordagens utilizadas para treinamento de cada modelo, bem como metodologias de testes utilizadas para análise de cada um deles.

4.2.1 KNN

Após a realização do pré-processamento dos dados, foi necessário resolver dois problemas. O primeiro, qual valor de K escolher para o trinamento, visto que, valores muito altos podem incluir informação de dados muito distantes e simplificam a região de decisão, e em contrapartida, valores muito baixos podem ser sensíveis a ruído e tornam a região de decisão mais complexa. O segundo, qual métrica de distância escolher.

Para a solução, foi criada uma lista de n inteiros num alcance de 1 à 40 para os valores de K. A escolha de duas distâncias, Euclidiana e Manhattan. A primeira é a distância entre dois pontos quaisquer, calculada usando a trigonometria pitagórica. E a segunda, também conhecida como "City Block", na qual as distâncias são definidas como a soma das distâncias ao longo de cada dimensão. Em outras palavras, as diferenças em cada um dos recursos são medidas independentemente e, em seguida, todas as diferenças são somadas.

Também vale ressalta, que afim de evitar que alguns atributos sejam tratados como mais importantes por terem magnitude muito maior que outros, foi feita uma normalização dos dados durante o treinamento do modelo.

Em seguida, com o uso do grid-search houve o teste dos valores dos hiperparâmetros para decidir os melhores parâmetros para o modelo durante o treinamento, com base em suas acurácias. Isto é, para cada distância de cada valor de K, através da técnica de validação cruzada, treinar um modelo KNN com esses valores e obter sua acurácia média.

Com base nos dados obtidos da Tabela 2, foi escolhido o melhor modelo para realizar o retreino com os dados de treinamento. Posteriormente, foi realizada a predição utilizando a distância Manhattan com 26-vizinhos próximos com os dados de teste e calculado sua acurácia média, que por sua vez atingiu um valor de **90.5%**. Além disso, também foi realizada a construção da matriz de confusão presente na Figura 1 em razão de uma melhor visualização das predições realizadas.

4.2.2 Árvore de Decisão

Como foi comentado na Seção 3.1.2, as árvores de decisão realizam a seleção automática dos atribuídos a serem utilizados em cada um dos nós da árvore. Contudo, durante o treinamento do modelo, quatro hiperparâmetros foram definidos, visando testar diferentes modelos de árvores de decisão. Sendo primeiro deles o critério de impureza, variando entre entropia e o índice de Gini. O seguindo é o tamanho máximo da árvore (`max_depth`) variando de 1 até 10 níveis de profundidade. O terceiro é o número mínimo de amostras necessárias para dividir um nó interno (`min_samples_split`), chamaremos esse hiperparâmetro de MSS. E por ultimo o número mínimo

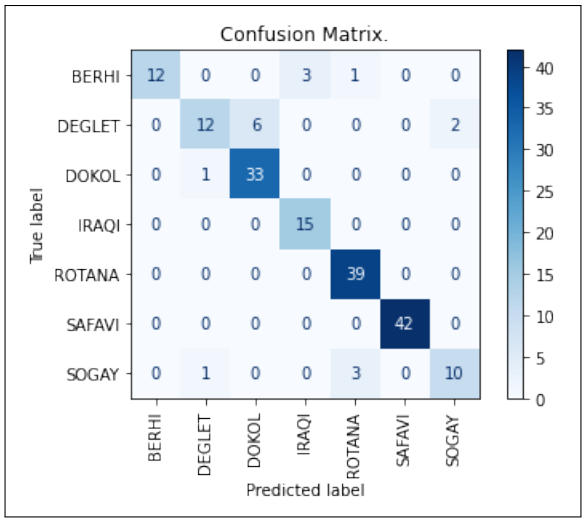

Figure 1: Matriz de confusão do KNN.

Table 3: Árvore de Decisão - Resultados obtidos após treinamento

Parâmetros				
Critério	Altura Máxima	MSS	MSF	Acurácia
Entropia	8	5	2	∼0.84675
Entropia	8	9	2	∼0.84673
Entropia	8	5	1	∼0.84259
Gini	8	4	1	∼0.83564
Gini	9	2	2	∼0.83292
Gini	9	3	2	∼0.83292

de amostras necessárias para estar em um nó folha, iremos nos referir a esse hiperparâmetro como MSF. Além disso, também foi definido um `random_state` como afim de tornar os dados replicáveis.

Esses hiperparâmetros foram definidos para testar diferentes modelos, além de possuir um maior controle de que tipo de árvore de será gerada com o intuito de evitar o overfitting do modelo. Como alguns dados possuiam uma ordem de grandeza muito grande em relação a outros, os exemplos de entrada foram normalizados durante o treinamento do modelo.

Para a escolha dos hiperparâmetros, todas as possibilidades 576 combinações fornecidas pelo grid search foram testadas utilizando o método da validação cruzada. Vale ressaltar que o particionamento dos dados foi feito na etapa de pré-processamento e que são as mesmas em todos os modelos testados. Os hiperparâmetros foram escolhidos através do modelo que apresentou a maior acurácia média ao final da validação cruzada.

Com base na Tabela 3, em que estão representados três melhores resultados para cada critério de impureza. É possivel aferir que a melhor escolha de hiperparâmetros está representada pela primeira linha da tabela. Após a escolha dos hiperparâmetros, foi realizado um novo treinamento e uma predição com os dados de teste, que por sua vez atingiu uma acurácia de $86,11\%$. Por fim, a matriz de confusão (Figura 2) foi construída com o intuito de aferir a qualidade das predições realizadas pelo modelo.

4.2.3 Redes Neurais

Como abordado na seção 3.1.3, os modelos de redes neurais apresentam diversos hiperparâmetros tornando muito demorado o treinamento desses modelos utilizando a abordagem de grid-search, como no treinamento do modelo 3.1.1. Para contornar tal problema, foi adotada a estratégia de Random Search, onde os hiperparâmetros foram sorteados e então a rede neural era treinada. Dessa

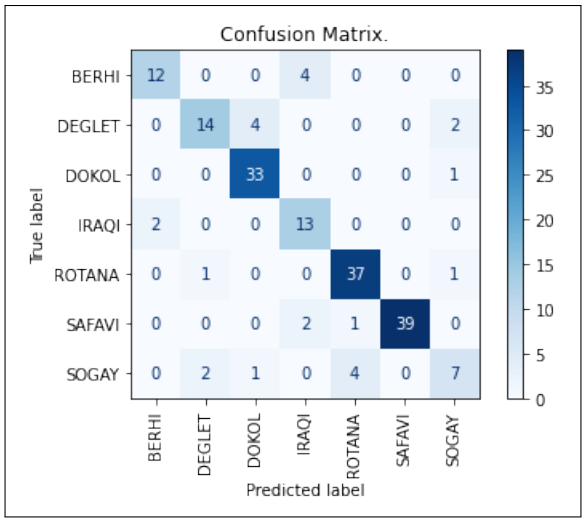

Figure 2: Matriz de confusão da Árvore de Decisão.

forma, diferentes combinações de modelos podem ser testadas e a solução do problema pode ser analisada com um menor custo computacional.

Foram realizadas combinações de hiperparâmetros para geração de 10 modelos diferentes. Alguns hiperparâmetros foram mantidos fixos, como a função de ativação, a qual foi escolhida 'relu', pois tende a apresentar uma melhor performance em problemas de classificação, o solver foi o gradiente descendente estocástico e foi utilizada apenas uma camada oculta. Todos os demais hiperparâmetros, os quais foram detalhados na seção 3.1.3, foram utilizados no Random Search.

Para definir a escolha de hiperparâmetros, todos os modelos especificados pelo random search foram treinados utilizando validação cruzada, com as partições de dados separados previamente na etapa de pré-processamento. A cada etapa da validação cruzada, ou seja, quando 9 das 10 partições eram utilizadas para teste e 1 para validação, era analisada a acurácia do modelo para os dados de validação. Esse processo se repete até que todas as partições tenham sido utilizadas para validação. Ao final do processo, foi analisada a acurácia média de cada modelo.Sendo assim, o modelo escolhido foi aquele que apresentou a maior acurácia média após a validação cruzada.

Escolhido os hiperparâmetros do melhor modelo, um novo treinamento foi realizado, dessa vez utilizando todos os dados de teste disponíveis. A acurácia então era obtida a partir dos dados de teste, também separados durante a etapa de pré-processamento. A partir do resultado dessa métricas que foram comparados os resultados do modelos diferentes utilizados para tentar solucionar o problema de classificação em análise.

Durante a última etapa de treinamento foi coletada a curva de aprendizado do modelo, para verificar se estava acontecendo o processo de overfitting nos dados de teste. Foi constatado que não estava havendo overfitting, A curva pode ser observada na Figura 3.

Durante a etapa de treinamento com validação cruzada, o melhor modelo obteve acurácia média de **92,3 %**. Após o retreino, utilizando todos os dados de treinamento, e analisando a acurácia nos dados de teste, a acurácia obtida foi de **92,8 %**. Na Figura 4 pode ser observado a matriz de confusão do gerada pela predição das classes utilizando os dados de teste.

4.2.4 Análise dos resultados

O modelo que apresentou melhor acurácia foi o modelo de Redes Neurais, o qual conseguiu resultados de aproximadamente 93% para esta métrica. Observando a matriz de confusão presente na Figura 4, podemos constatas que 3 classes foram preditas com 100 % de acerto, enquanto que as outras classes apresentaram resultados muito bons ainda assim, com destaque para a classe IRAQI, que apresenta poucos dados na classe e apenas um padrão foi classificado errado. É possível observar também que boa parte dos erros aconteceu na classe DOKOL.

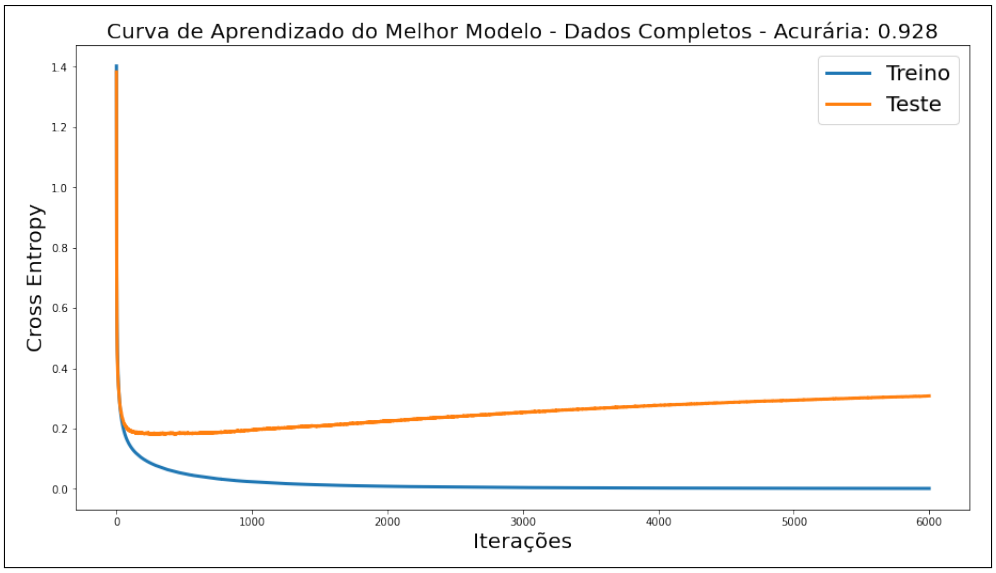

Figure 3: Curva de treinamento do melhor modelo de Rede Neural.

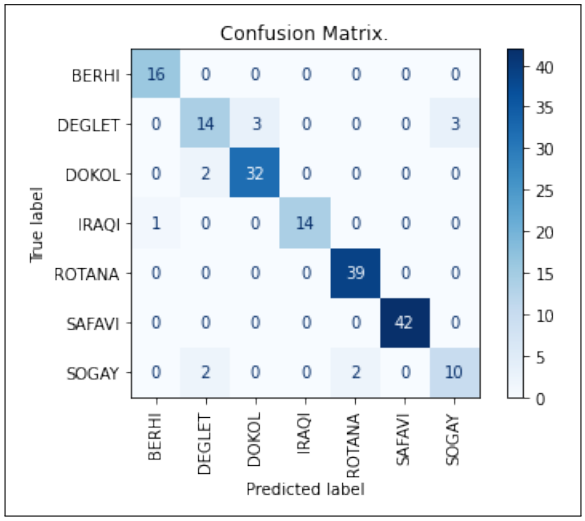

Figure 4: Matriz de Confusão da Rede Neural.

4.3 Redução de Dimensionalidade

Observando que haviam muitos atributos no problema, buscou-se averiguar a possibilidade de tratar o problema com uma gama reduzida de atributos, para isso, utilizou-se o Score de Fisher para identificar quais atributos seriam menos impactantes para serem removidos. Outros teste realizados foram as projeções dos dados, com originais e com menos atributos, em um espaço de dimensão 2. Os tópicos a seguir abordam os principais resultados obtidos.

4.3.1 Score de Fisher e PCA

Como abordado na seção 3.2.1, o score de Fisher busca indicar quais atributos variam menos entre as classes. A Figura 5 apresenta o resultado obtido para cada um dos 34 atributos.

A partir da análise do Score de Fisher para cada Atributo, propôs-se duas abordagens, a primeira delas foi a remoção de 9 dimensões, totalizando ao final da retirada 25 dimensões. E na segunda abordagem, mais agressiva, removeu-se 24 atributos, restando apenas 10 para realizar a tarefa de classificação. Buscou-se observar qual seria o impacto na acurácia do modelo escolhido com maior

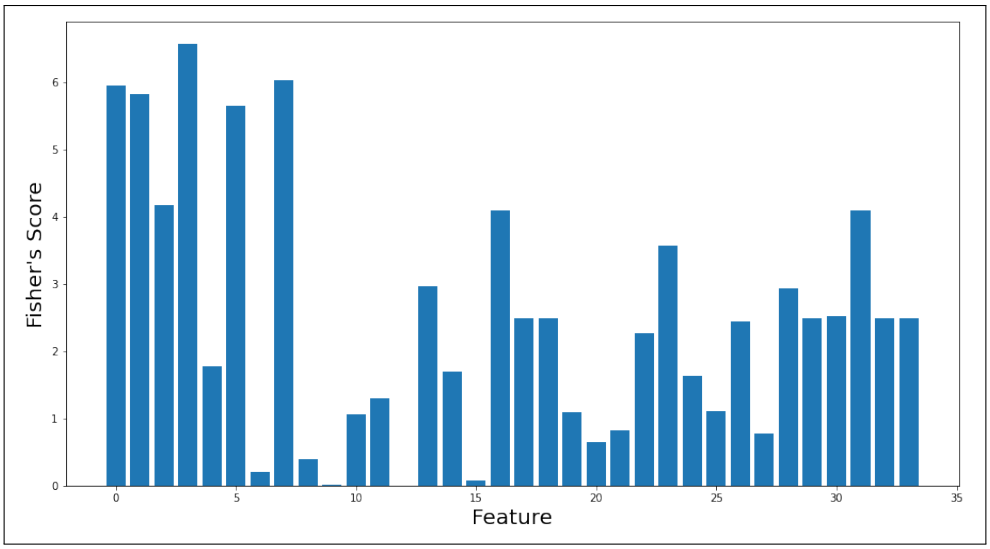

Figure 5: Score de Fisher.

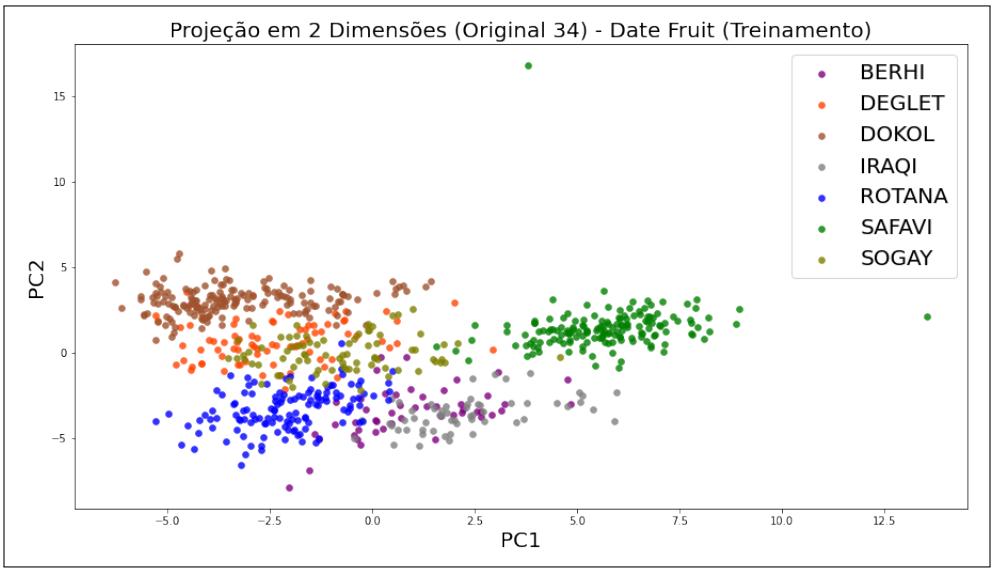

Figure 6: Dados projetados em duas dimensões - 34 dimensões originalmente.

acurácia quando o treinamento foi realizado utilizando todas as dimensões, no caso, o modelo de redes neurais.

Buscando observar o comportamento dos dados em uma dimensão reduzida, utilizou-se o PCA para projetar as três condições descritas previamente em uma espaço de duas dimensões, no caso, uma redução de 34 dimensões para 2 dimensões, de 24 para 2 dimensões e de 10 para duas dimensões. Inicialmente, busca-se observar se os dados projetados apresentam uma boa separação entre si e entre as classes diferentes, algo que poderia indicar na redução de dimensionalidade há uma compressão muito grande dos dados e se estão diferentes mesmo em um espaço de dimensão reduzida, em caso afirmativo, seria possível relacionar com os dados no espaço de dimensão original.

A Figura 6 apresenta a projeção dos dados originais, com 34 dimensões no espaço de duas dimensões.

Como se pode observar, as classes apresentam uma boa separação nos dados projetados, com exceção das classes DEGLET e SOGAY, bem como as classes IRAKI e BERHI onde há um pouco mais de contaminação entre as classes.

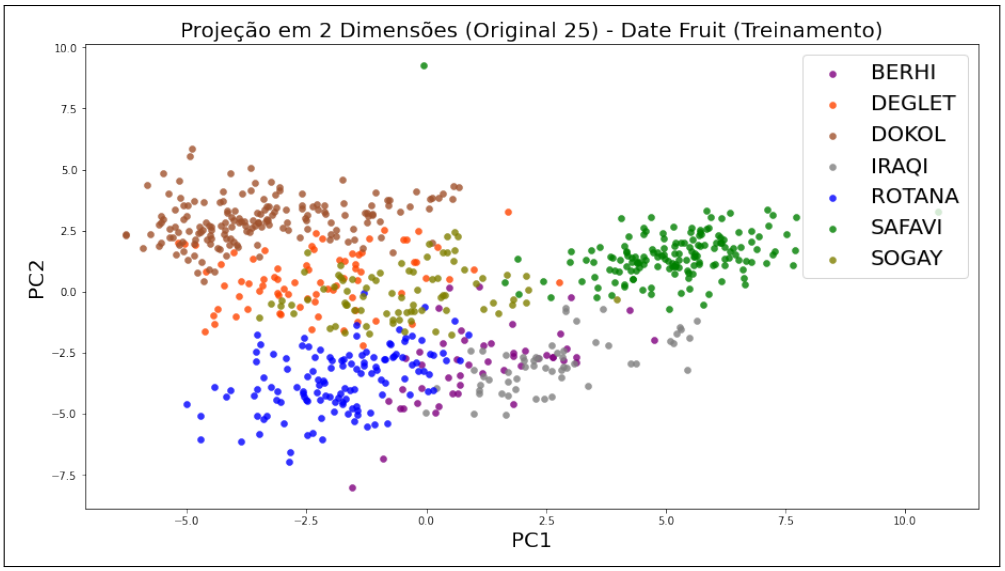

Figure 7: Dados projetados em duas dimensões - 25 dimensões originalmente.

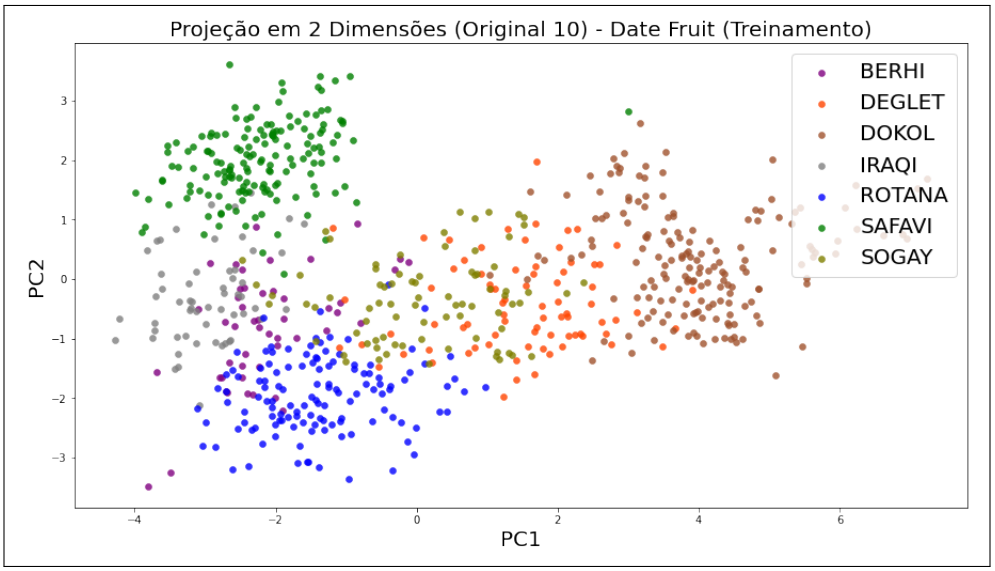

Figure 8: Dados projetados em duas imensões - 10 dimensões originalmente.

Foram também realizadas as projeções dos dados após a análise do Score de Fisher, ou seja, após serem removidas a quantidade especificada de dimensões. Os dados utilizados foram os mesmos para a análise com 34 dimensões. As Figuras 7 e 8 apresentam os resutlados obtidos para os dados projetados com menos dimensões.

Como pode-se observar, mesmo com a quantidade de dimensões reduzidas na entrada, os modelos continuaram se mantendo distintos no espaço projetado, o que seria um bom indicativo de que a redução de dimensões não implicaria em uma redução considerável na acurácia dos modelos quando treinados utilizado dados com menos dimensões, pois estes continuaram bem separados mesmo quando projetados no espaço de duas dimensões.

Visando quantificar a análise descrita previamente, foi aplicado nos dados projetados o algoritmo KNN, presente na seção 3.1.1, utilizando apenas 1 vizinho mais próximo para classificar os dados projetados em suas respectivas classes, com o intuito de verificar quão bem agrupados os dados de uma mesma classe estariam. Para realizar a projeção, obtenção da matriz de transformação do PCA,

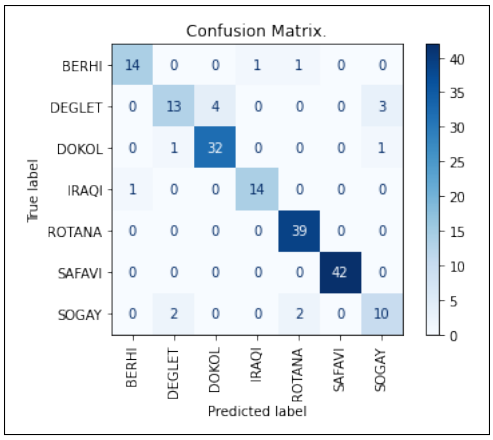

Figure 9: Matriz de Confusão - modelo de Rede Neural - dados com 25 atributos.

foram utilizados os dados de treinamento, bem como para treinamento do KNN com 1 vizinho, a verificação da acurácia dos dados projetados foi realizada utilizando os dados de teste. Os resultados obtidos foram:

- 34 Dimensões (Acurácia do 1NN para os dados projetado em 2 dimensões): **80,0 %**:
- 25 Dimensões (Acurácia do 1NN para os dados projetado em 2 dimensões): **85,0 %**:
- 10 Dimensões (Acurácia do 1NN para os dados projetado em 2 dimensões): **78,3 %**:

Os dados originais com menos dimensões, com apenas 25 acabaram por deixar as classes mais bem separadas do que os dados com os 34 atributos originais. Ao mesmo tempo que os dados com apenas 10 dimensões acabaram por continuar apresentando uma boa separação entre as classes, muito próxima dos dados originais, porém com uma redução expressiva na quantidade de atributos.

4.3.2 Redes Neurais com Menos Dimensões

Após a observação da preservação da qualidade dos dados no PCA mesmo com menos atributos, optou-se por realizar um novo treinamento para o modelo de redes neurais utilizando dados com menos dimensões que os dados originais utilizados na seção 3.1.3. A metodologia adotada para o treinamento foi o mesmo descrito na seção 3.1.3.

Para os dados com apenas 25 atributos, a acurácia média obtida na validação cruzada para o melhor modelo ficou em **91,8 %** e **91,1 %** quando realizado o retreino com todos os dados de treinamento e comparados com os dados de teste. Na Figura 9 está presente a matriz de confusão obtida para os dados de teste e o modelo de rede neural treinado com os dados com apenas 25 dimensões.

De forma semelhante, foi realizado o treinamento do modelo de Redes Neurais utilizando os dados com apenas 10 atributos. Durantes a etapa de validação cruzada, a acurácia média do melhor modelo treinado foi de **85,0 %**, enquanto que durante a etapa de treinamento, foi obtida uma acurácia de **86,7 %**. Na Figura 10 pode-se observar a matriz de confusão para o modelo utilizando os dados com 10 atributos.

4.3.3 Análise dos Resultados

O primeiro ponto a ser comentado é que mesmo quando os dados apresentaram apenas 10 dimensões, o modelo apresentou uma acurácia alta, algo que pode ser vantajoso quando se pensa na redução de dados que precisará ser coletada das amostras e no custo computacional do número elevado de atributos por amostra.

Outro ponto a ser observado é que nas 3 abordagens para treinamento do modelo, o modelo foi capaz de predizer com boa acurácia 3 classes, a DOKOL, a ROTANA e a SAFAVI, as quais são a mais bem separadas, quando observadas nos gráficos da projeção em duas dimensões, ou seja, a projeção tem sido um bom indicativo de separação de dados.

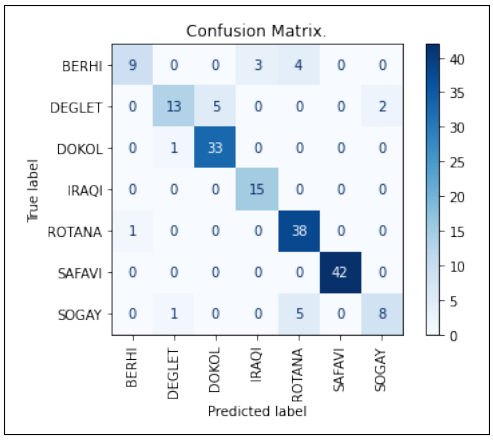

Figure 10: Matriz de Confusão - modelo de Rede Neural - dados com 10 atributos.

Nos casos dos dados com maior quantidade de dimensões (34 e 25), a maior parte dos erros ocorreu entre as classes DEGLET e DOKOL, o que pode ser constatado pela contaminação entre as duas classes, a qual se manteve semelhante entre os dois casos.

Por fim, pode-se observar que a grande piora da acurácia do modelo treinado com 10 dimensões se dá nos erros da predição da classe BERHI, que, quando projetada em 2 dimensões, pode-se observar uma grande mistrura com a classe ROTANA e IRAQI, que são justamente as classes que indicam erros na predição na matriz de confusão.

5 Conclusão

Nesse contexto, este trabalho apresentou uma avaliação de desempenho dos modelos KNN, Árvores de Decisão e Redes Neurais no problema de classificação de frutas de tâmara. Demonstrando ao final que o modelo de Redes Neurais Artificiais é o melhor para predição de classes no conjunto de dados estudado.

Também, foi visto que apesar de alguns erros de predição o modelo ainda apresenta uma acurácia alta mesmo quando treinado com menos dimensões de dados, que por sua vez tem suas vantagens, se levado em conta a redução de dados que precisam ser coletados ou o custo computacional do elevado número de atributos.

Resultados mais bem sucedidos podem ser obtidos com a utilização de mais métricas para avaliar os modelos, como métricas de precisão, revocação e f1-score. Bem como, aplicar a redução de dimensionalidade em atributos específicos, a exemplo: verificar se características de feições morfológicas e dimensões da fruta são suficientes para classificar as tâmaras entre os setes tipos presente no conjunto de dados.

Referências

[1] KOKLU, M., KURSUN, R., TASPINAR, Y. S. and CINAR, I. (2021). Classification of Date Fruits into Genetic Varieties Using Image Analysis. Mathematical Problems in Engineering, Vol.2021, Article ID: 4793293.

[2] MURPHY, Kevin P. (2012). Machine Learning: A Probabilistic Perspective.

[3] "sklearn.neighbors.KNeighborsClassifier". Scikit-Learn, 2022, https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[4] "sklearn.neural$_n etwork.MLPClassifier$". $Scikit - Learn$, 2022, $https : //scikit - learn.org/stable/modules/generated/sklearn.neural_n etwork.MLPClassifier.html$

[5] "sklearn.tree.DecisionTreeClassifier". https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[6] "sklearn.decomposition.PCA". Scikit-Learn, 2022, https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html