Relatório – Atividade Prática de Mineração de Dados

Este relatório está estruturado em cinco partes. A primeira destina-se ao entendimento de uma base de dados previamente selecionada. Na segunda e terceira efetuam-se técnicas de pré-processamento, dentre elas transformação de dados e redução da dimensionalidade. Na quarta e quinta partes efetuam-se atividades de predição/classificação utilizando-se de técnicas de *machine learning* (aprendizado de máquina) e por fim analisam-se estatisticamente os resultados obtidos. Em linhas gerais este relatório segue a metodologia KDD para geração de conhecimento e validação do conhecimento gerado, metodologia esta abordada nos próximos parágrafos.

Para este experimento utilizaram-se vários softwares de apoio às atividades de mineração de dados, entre elas o ***RStudio***, na versão *1.1.456*, que é uma IDE de programação para a linguagem ***R*** (versão utilizada Rx64 3.5.1), o ***Weka***, na versão *3.8.3*, o ***RapidMiner Studio***, na versão educacional 9.2.001 e o *Libreoffice CALC* versão 5.2.5.1 (x64).

Para fins deste relato assume-se, para simplificação da escrita, que os textos escritos na fonte `courrier new` referem-se à escolha de parâmetros nos softwares Weka, R ou RapidMiner, cujo nome refere-se ao texto escrito **ou** a um clique num botão rotulado com o mesmo texto escrito **ou** mesmo ao nome dado a uma aba das janelas. Quando nestes textos existir o símbolo → assume-se que serão acessadas opções hierarquicamente dependentes.

Durante a disciplina de Mineração de Dados estudaram-se várias técnicas e ferramentas relacionadas ao processo de KDD (*Knowledge Discovery in Databases*), que segundo Fayyad et al (1996) está relacionado a um processo interativo e iterativo de descoberta de conhecimento, em especial em Banco de Dados, que englobam um conjunto de teorias e ferramentas computacionais para extrair informações potencialmente úteis. A Figura 1 ilustra uma visão geral do processo de KDD, com ênfase nos passos iterativos a serem executados para descobertas de conhecimento.

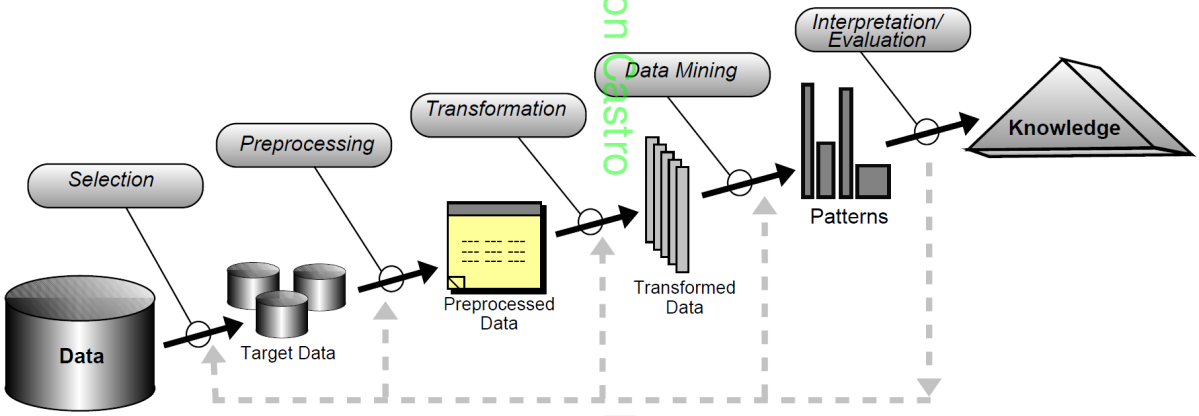


Figura 1: Uma visão geral dos passos que compõem o processo KDD.
Fonte: [Fayyad et al. 1996]

A primeira etapa de um processo de KDD é conhecida como Seleção e nela se escolhem um ou vários conjuntos de dados, que podem ser provenientes de várias fontes, e gera-se um conjunto de dados resultante o qual servirá de base para extração de conhecimento. No tocante a esta etapa trabalhou-se com apenas um conjunto de dados, para este relatório efetuou-se o *download* de um arquivo disponível em https://pistori.weebly.com/datasets.html denominado **aquario28e40i.arff**, que contém um conjunto de treinamento e testes que refere-se a experimentos de classificação de espécies de peixes.

Um arquivo de extensão **.arff,** que é normalmente utilizado pelo software **Weka**, conforme discutido adiante, está logicamente dividido em três seções. Na primeira, que inicia com a palavra reservada @relation, define-se o nome do conjunto de treinamento e testes. Na segunda, listam-se, linha a linha, os nomes de cada um dos atributos constantes no conjunto de treinamento, os quais são precedidos da palavra reservada @attribute. Na última seção, após a palavra reservada @data, apresentam-se, também linha a linha, as instâncias do conjunto de treinamento e testes. Efetuando uma analogia com banco de dados, o conjunto de treinamento e testes assemelhar-se-ia a uma tabela, os atributos aos campos da tabela, e as instâncias a cada um dos registros da tabela, ou seja, cada uma das amostras do conjunto de treinamento.

O objetivo principal deste experimento é efetuar a *comparação do desempenho de quatro técnicas de aprendizado de máquina em predizer espécies de peixes, em duas bases de dados, uma sem pré-processamento e outra pré-processada*. Tem-se neste caso um problema de classificação, no qual todos os dados de entrada da base de dados são numéricos e a resposta esperada é nominal ou categórica (discreta). O atributo *classe* equivale ao atributo preditivo/classe/alvo do processo de classificação. Tem-se portanto um experimento de classificação com **28 classes,** onde cada classe representa uma espécie de peixe distinta.

O conjunto de treinamento e testes original (sem pré-processamento) possui **226 atributos**, sendo que os 225 primeiros são numéricos e o último deles é categórico e refere-se à classe. Analisaram-se todas as **1118 instâncias** deste conjunto de treinamento. Notou-se que o conjunto de treinamento estava quase inteiramente balanceado, pois 27 espécies possuíam 40 instâncias e apenas a espécie *pacu* possuía 2 instâncias a menos, ou seja, 38.

Cada instância/exemplo da base de dados analisada representa um *vetor de atributos*, resultante do processo de extração de atributos de um conjunto de imagens anotadas das várias espécies de peixe, gerado por um sistema de visão computacional. Como dito anteriormente, o último atributo de cada instância refere-se à classe.

De maneira geral, neste tipo de problema, espera-se que as técnicas de classificação gerem modelos a partir das instâncias de exemplo, e dado uma nova instância não presente no conjunto de exemplos, o modelo possa classificá-la corretamente.

Como dito anteriormente, na segunda e terceira partes deste relatório utilizaram-se de técnicas de pré-processamento, dentre elas transformação de dados e redução da dimensionalidade.

Observa-se que neste experimento não foi necessário preencher os valores ausentes da base de dados, pois embora a base tenha dimensionalidade relativamente alta, 226 atributos, os valores destes estavam todos preenchidos. Observa-se contudo, que vários algoritmos de aprendizado de máquina não foram projetados para trabalhar com valores ausentes ou incompletos e nestes casos seria necessário utilizar uma ou várias técnicas para imputação de valores, as quais requerem, para melhoria dos resultados, análises detalhadas dos motivos das inexistências dos valores e também de estimativas do comportamento geral das classes na população como um todo.

Em várias áreas da computação, incluindo a mineração de dados, o termo processar denota transformação, manipulação. Num contexto de KDD transformam-se os dados para melhorar a qualidade deles, diminuir suas complexidades de organização e facilitar suas utilizações posteriores. Tais transformações envolvem operações que alteram os valores dos atributos, tais como a padronização, a normalização, o valor absoluto.

Como dito anteriormente os arquivos .arff são normalmente utilizados pelo **Weka**. Como pretendia-se efetuar as etapas de pré-processamento através do Rapidminer, o qual não efetua uma leitura nativa deste tipo de arquivo, utilizou-se o operador *Read ARFF*, carregou-se o referido arquivo o qual foi posteriormente convertido num arquivo CSV conforme ilustrado na Figura 2. Efetuou-se em seguida a importação do CSV para o repositório local do RapidMiner.

Figura 2: Carregamento do conjunto de dados em formato .arff para o Rapidminer Studio com posterior conversão para CSV.

Quando se necessitam efetuar comparações entre variáveis que encontram-se em diferentes referenciais/diferentes escalas é desejável levá-las para um mesmo referencial, ou de maneira mais técnica, é preciso efetuar uma operação matemática sobre os dados de maneira a normalizá-los. Para tal utilizou-se o operador *Normalize*, do RapidMiner Studio, para efetuar a operação de normalização dos dados – vide Figura 3. Setou-se `atribute filter type` para "subset" e clicando-se no botão `Sellect Attibutes` deste operador escolheu-se de maneira interativa todos os atributos exceto a classe, de maneira que a normalização ocorresse sobre todos os atributos

numéricos. O método de normalização escolhido foi o "z_transformation", também chamado de normalização estatística. Neste método de normalização subtraem-se de todos os dados o valor da média calculada, e em seguida divide-se o valor obtido pelo desvio padrão da amostra ou da população.

Desvio padrão é uma medida de dispersão que mostra o quanto os dados variam em relação à média. Quanto menor o desvio padrão, os dados tendem a estar próximos da média e quanto maior, também maior será a variabilidade dos dados, ou seja, eles estarão mais espalhados em relação à média.

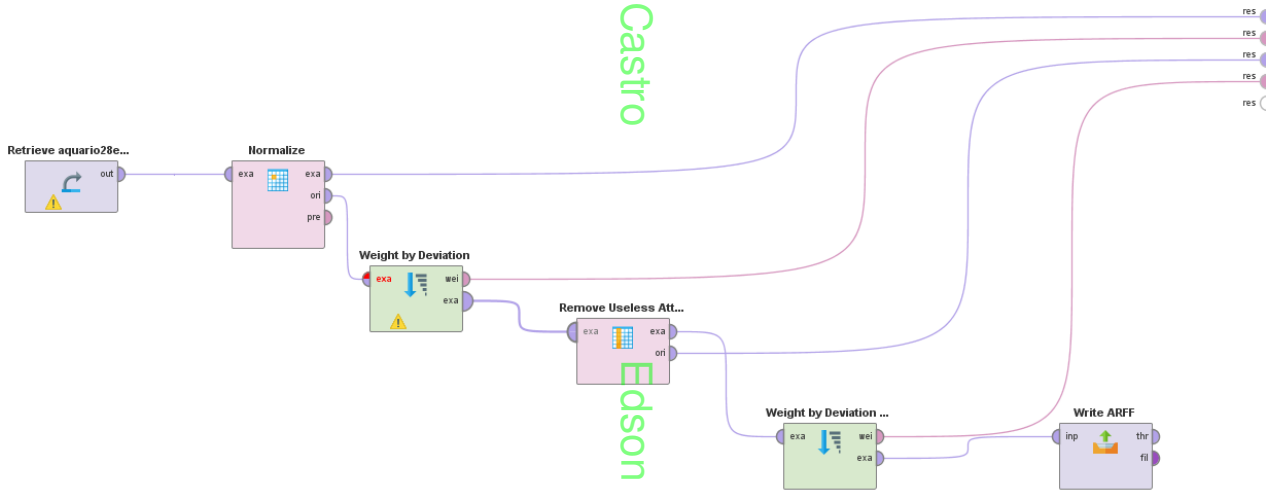


Figura 3: Processos de transformação de dados através do operador *Normalize* e redução da dimensionalidade (remoção dos atributos dispensáveis) através do operador *Remove Useless Attributes* do RapidMiner Studio.
Fonte: Autoria Própria.

Efetuada a transformação de normalização do conjunto de dados, conferiu-se que a média aritmética dos valores de todos os atributos zerou e os desvios padrões estavam em 1, fato ilustrado na Figura 4. Nota-se também, pela análise desta figura, que os padrões (*shapes*) dos histogramas permaneceram inalterados após o processo de normalização de dados com o método *z_transformation*.

Após a normalização de dados efetuou-se a análise com intuito de remoção de "*atributos dispensáveis*", processo feito no Rapdiminer Studio e ilustrado na Figura 3. Para tal utilizou-se o operador *Remove Useless Attributes*. Calcularam-se inicialmente os pesos por desvio para todos os atributos da base de dados, através do operador *Weight by Deviation* do RapidMiner Studio, processo também ilustrado na Figura 3 e resultados ilustrados no gráfico da Figura 5.

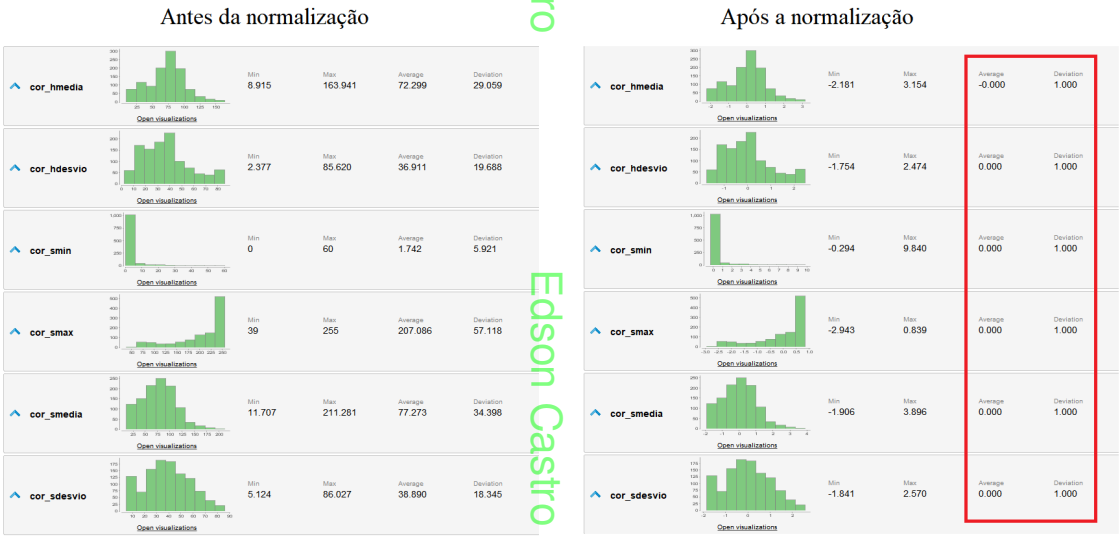


Figura 4: Comparação dos histogramas, médias dos valores e desvios padrões dos atributos antes e após o processo de normalização de dados.

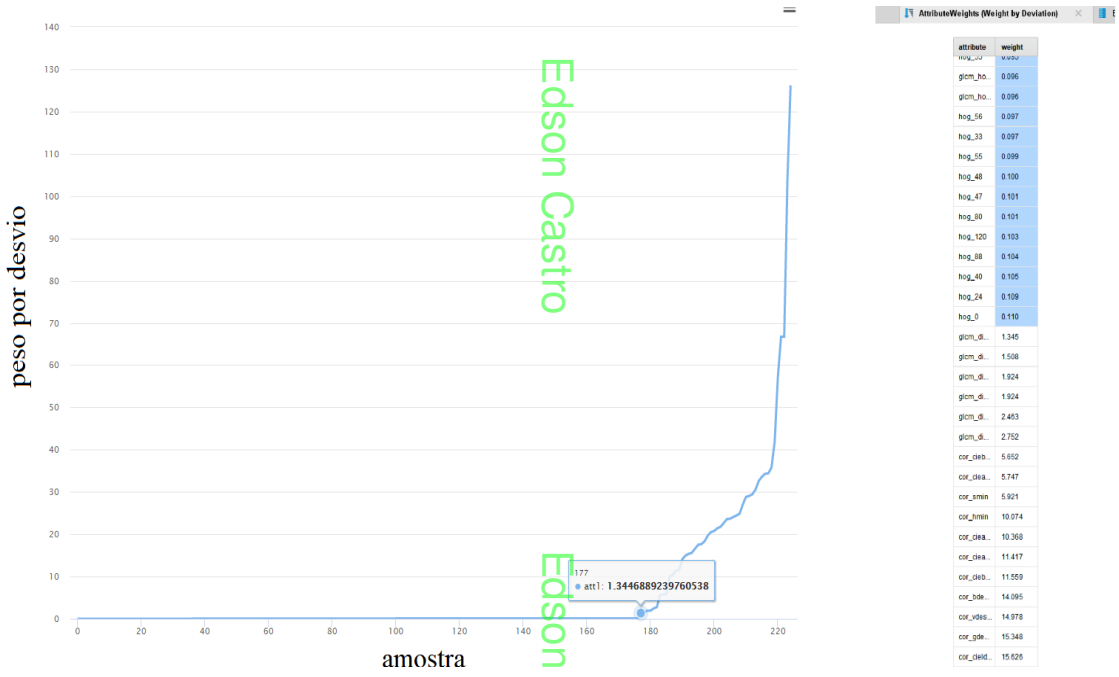


Figura 5: À esquerda gráfico resultante do cálculo dos pesos por desvios para todos os 225 atributos da base de dados. Plotaram-se as amostras ordenadas crescentemente pelo valor dos pesos. À direita um subconjunto da tabela com os pesos por desvios calculados para os atributos.
Fonte: Autoria Própria

Analisando-se o gráfico da Figura 5 nota-se que dos 225 atributos da base de dados cerca de 180 deles apresentavam pesos por desvios próximos a zero. Isto representa que tais atributos provavelmente teriam baixo poder preditivo, pois dispersam muito pouco em relação à média e desta maneira poderiam ser considerados "*atributos dispensáveis*" e portanto seriam removidos.

Utilizou-se o operador *Remove Useless Attributes* do Rapdminer, com `numerical min deviation` (desvio padrão numérico mínimo) setado para 1, o que resultou num conjunto de dados com apenas 49 atributos. Dado que o conjunto de dados original possuía um total de 225 atributos, 176 deles foram removidos pelo operador por serem considerados dispensáveis.

Terminada a etapa de pré-processamento através do Rapidminer o conjunto de dados resultante foi gravado num arquivo através do operador *Write ARFF*.

Descrevem-se a seguir a quarta e quinta partes deste relatório nas quais efetuaram-se atividades de predição/classificação utilizando-se de técnicas de aprendizado de máquina e analisaram-se estatisticamente os resultados obtidos. As atividades de classificação foram efetuadas na ferramenta Weka e para a análise estatística dos resultados dos experimentos utilizou-se o software *RStudio*.

É importante observar, que efetuaram-se experimentos com dois conjuntos de dados, gravados inicialmente em dois arquivos .arff. No primeiro utilizaram-se os dados sem tratamento e no segundo os dados foram pré-processados conforme descrito anteriormente. Efetuou-se também um comparativo de desempenho classificatório entre eles.

Inicialmente, para os dois conjuntos de dados utilizou-se a aplicação `Explorer`, que se refere ao primeiro botão da tela principal do Weka. Carregaram-se os arquivos que contém os dados dos conjuntos de treinamento e teste (arquivos **.arff** descritos anteriormente). Este carregamento se deu através do botão `Open File` da aba `Preprocess`.

Utilizaram-se quatro técnicas diferentes de aprendizado de máquina supervisionada para classificação das instâncias relacionadas às imagens de peixes, técnicas estas que também serão referenciadas por algoritmos de aprendizado supervisionado, ou simplesmente por classificador. Essas técnicas de aprendizado utilizadas e disponíveis no Weka são: RandomForest, SMO, J48 e IBk, as quais serão comentadas adiante.

Para escolha das técnicas de classificação clicava-se no botão `classifier`→`choose` da aba `Classify` e procedia-se a escolha do classificador pretendido. É possível também editar os parâmetros dos classificadores, clicando na caixa de texto ao lado deste botão.

No intuito de verificar as capacidades dos algoritmos predizerem novas instâncias que não constam no conjunto de treinamento e teste efetuaram-se, para os dois conjuntos de dados e para os quatro algoritmos de classificação, testes com 10 dobras aleatórias com validação cruzada de dados. Para tal, na aba `Classify` informou-se o valor 10 para `Folds` na opção `Test options`→`Cross-validation`. Isto significa que os algoritmos de classificação primeiramente ordenam o conjunto de instâncias de forma aleatória e particionam-nas em 10 subconjuntos de tamanho aproximadamente igual; daí procede-se de forma iterativa, onde em cada iteração utiliza-se

um dos subconjuntos para testes e os outros nove para treinamento; após variar entre todos os subconjuntos de instâncias combinam-se os dados de classificação. É importante salientar que os algoritmos "*enxergam*" as classificações dos dados dos subconjuntos de treinamento e tentam predizer as classes das instâncias presentes nos conjuntos de testes.

Descreve-se agora, sucintamente, cada um dos algoritmos utilizados nos experimentos.

No Weka, IBk refere-se a uma classe de algoritmos de aprendizado baseados em instâncias (*Instance Based Learning*). O algoritmo de aprendizagem supervisionada KNN (k vizinhos mais próximos), é um exemplo desta classe de algoritmos. Segundo Amaral (2016, p. 96) no aprendizado baseado em instâncias não é construído um modelo prévio de classificação, de maneira que a classificação ocorre em memória com a utilização dos dados históricos. No exemplo do KNN, a cada nova instância a classificar escolhe-se a classe cujas instâncias mais se aproximam desta nova instância, segundo uma métrica de distância, que pode ser a euclidiana, consideradas as k instâncias mais próximas.

O algoritmo J48 é uma implementação em Java de um algoritmo de aprendizado baseado em árvores de decisão denominado C4.8, que é um melhoramento do algoritmo C4.5, um dos mais conhecidos para aprendizado de máquina com base em árvores. Algoritmos baseados em árvore de decisão são comumente implementados em Ciência da Computação. No C4.5 e suas variantes a classificação de uma nova instância ocorre nos nós folha. Segundo Amaral (2016, p.99) esses algoritmos baseados em árvores de decisão necessitam definir quais atributos de comparação para cada nó, a partir do nó raiz, e quais as condições de particionamento (criação de novos nós filhos). Segundo ele a definição do particionamento depende do grau de pureza dos filhos. Neste contexto uma das métricas mais importantes a ser considerada para a definição dos atributos a serem utilizados nestas árvores de decisão é exatamente aquela que gere uma diminuição da entropia dos dados, com consequente aumento do ganho de informação. Desta maneira, pode-se estabelecer um limite para a profundidade da árvore baseada no aumento ou não do ganho de informação.

Como algoritmos baseados em árvores de decisão costumam ser instáveis, sujeitos a variação dos dados de novas instâncias, surgiu a ideia das florestas de árvores, técnica utilizada no algoritmo Random Forest, implementado no Weka. Segundo as informações adicionais deste classificador, disponível no próprio Weka, o algoritmo constrói um conjunto de árvores aleatórias (uma floresta). Neste processo, primeiramente subdivide-se o conjunto de instâncias e geram-se uma árvore para cada subconjunto. Dado uma nova instância a classificar, submete-a a todas as árvores geradas e escolhe-se a classificação com maior número de ocorrências dentre todas as árvores.

O algoritmo SMO trabalha com máquinas de vetores de suporte. Segundo Amaral (2016, p.102) os algoritmos de classificação baseados em vetores de suporte mostram-se eficientes pois minimizam superajustes e suportam grande número de atributos. As máquinas de vetores de suporte mostram-se mais eficientes que as regressões lineares pois definem a fronteira que melhor segrega duas classes, através de um hiperplano. A ideia principal dos hiperplanos é maximizar as margens que separam as instâncias do conjunto de treinamento. Uma nova instância será então classificada tomando-se a menor distância entre as margens do hiperplano, ou seja, qual das classes ela mais se aproxima.

Após escolhidos os algoritmos de aprendizagem de máquina supervisionada, para cada um deles executou-se a experimentação através do botão `Test Options→Start` da aba `Classify`. O Weka disponibiliza os resultados das experimentações em *Classifier output.* A Figura 6 sintetiza as porcentagens de instâncias classificadas correta e incorretamente, para cada técnica de classificação, discriminadas para os dois conjuntos de dados. Estas informações foram extraídas da seção *Summary*.

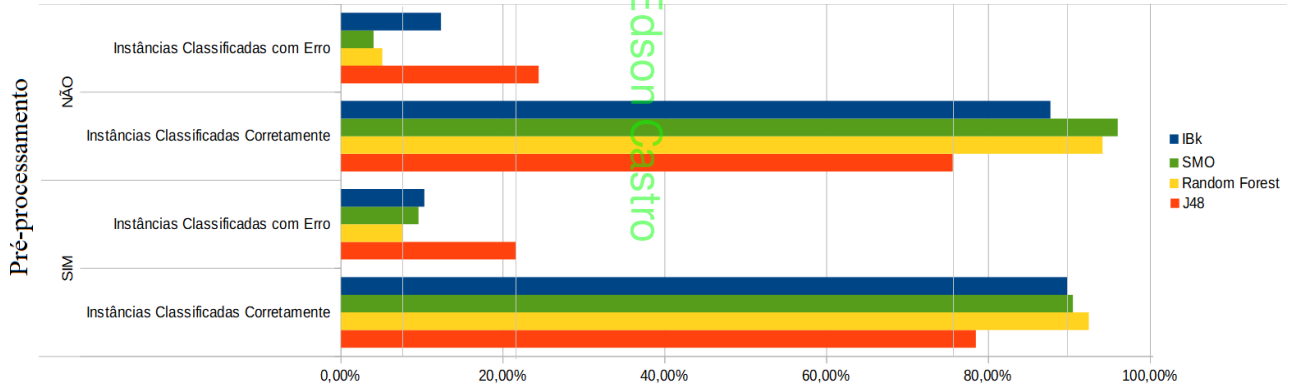
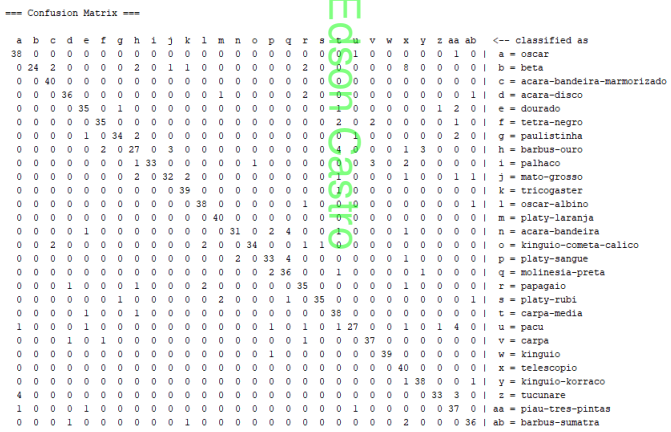


Figura 6: Gráfico dos resultados das classificações com as quatro técnicas de aprendizado de máquina supervisionada. Na parte superior da figura as classificações sem a utilização de pré-processamento. Na parte inferior as classificações resultantes da base de dados pré-processada.

Analisando-se as informações da Figura 6 observa-se que, no geral, o algoritmo J48 teve desempenho de classificação bastante inferior aos demais algoritmos. Como esperado, o algoritmo Random Forest apresentou melhor desempenho que o J48. Observa-se que a etapa de pré-processamento melhorou o desempenho preditivo dos algoritmos J48 e IBk. Todavia, os algoritmos SMO e Random Forest apresentaram pior desempenho na base de dados pré-processada do que na base de dados na qual os dados não foram normalizados nem se retiraram os atributos ditos *dispensáveis.* Os percentuais de instâncias classificadas corretamente para os algoritmos IBk, Random Forest e SMO são muito próximos o que não permite afirmar, observando apenas estas informações, qual é o melhor algoritmo de classificação, fato que será explorado adiante.

Num processo de classificação, o Weka também gera matrizes de confusão, que permite a visualização detalhada do desempenho dos algoritmos para cada classe. Observe inicialmente a Figura 7, que mostra a matriz de confusão para o algoritmo IBk, na base de dados sem pré-processamento, e em seguida a Figura 8, que realça elementos importantes observados na mesma matriz de confusão.



```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z aa ab   <-- classified as
 38  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0 | a = oscar
  0 24  2  0  0  0  0  2  0  1  1  0  0  0  0  0  0  2  0  0  0  0  0  8  0  0  0  0 | b = beta
  0  0 40  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | c = acara-bandeira-marmorizado
  0  0  0 36  0  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0  0  0  0  0  0  0  1  0 | d = acara-disco
  0  0  0  0 35  0  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  2  0  0 | e = dourado
  0  0  0  0  0 35  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  2  0  0  0  1  0  0 | f = tetra-negro
  0  0  0  0  1  0 34  2  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0 | g = paulistinha
  0  0  0  0  0  2  0 27  0  3  0  0  0  0  0  0  0  0  0  4  0  0  0  1  3  0  0  0 | h = barbus-ouro
  0  0  0  0  0  0  0  0  1 33  0  0  0  0  0  1  0  0  0  0  3  0  2  0  0  0  0  0 | i = palhaco
  0  0  0  0  0  0  0  0  2  0 32  2  0  0  0  0  0  0  0  0  0  0  0  1  0  1  1  0 | j = mato-grosso
  0  0  0  0  0  0  0  0  0  0  0 39  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | k = tricogaster
  0  0  0  0  0  0  0  0  0  0  0  0 38  0  0  0  0  0  1  0  0  0  0  0  0  0  1  0 | l = oscar-albino
  0  0  0  0  0  0  0  0  0  0  0  0  0 40  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | m = platy-laranja
  0  0  0  0  1  0  0  0  0  0  0  0  0  0 31  0  2  4  0  0  0  0  0  1  0  0  0  0 | n = acara-bandeira
  0  0  2  0  0  0  0  0  0  0  0  2  0  0 34  0  0  1  1  0  0  0  0  0  0  0  0  0 | o = kinguio-cometa-calico
  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0 33  4  0  0  0  0  0  0  1  0  0  0  0 | p = platy-sangue
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2 36  0  0  1  0  0  0  0  1  0  0  0 | q = molinesia-preta
  0  0  0  1  0  0  0  1  0  0  0  2  0  0  0  0  0 35  0  0  0  0  0  1  0  0  0  0 | r = papagaio
  0  0  0  0  0  0  1  0  0  0  0  0  2  0  0  0  1  0 35  0  0  0  0  0  0  0  1  0 | s = platy-rubi
  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0 38  0  0  0  0  0  0  0  0 | t = carpa-media
  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0  1  0  1 27  0  0  1  0  1  4  0 | u = pacu
  0  0  0  1  0  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0 37  0  0  0  0  0  0  0 | v = carpa
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 39  0  0  0  0  0  0 | w = kinguio
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 40  0  0  0  0  0 | x = telescopio
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1 38  0  0  1 | y = kinguio-korraco
  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 33  3  0 | z = tucunare
  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 37  0 | aa = piau-tres-pintas
  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0 36  0 | ab = barbus-sumatra
```

Figura 7: Matriz de confusão gerada pelo Weka para o algoritmo IBk (base de dados sem pré-processamento).
Fonte: autoria própria.

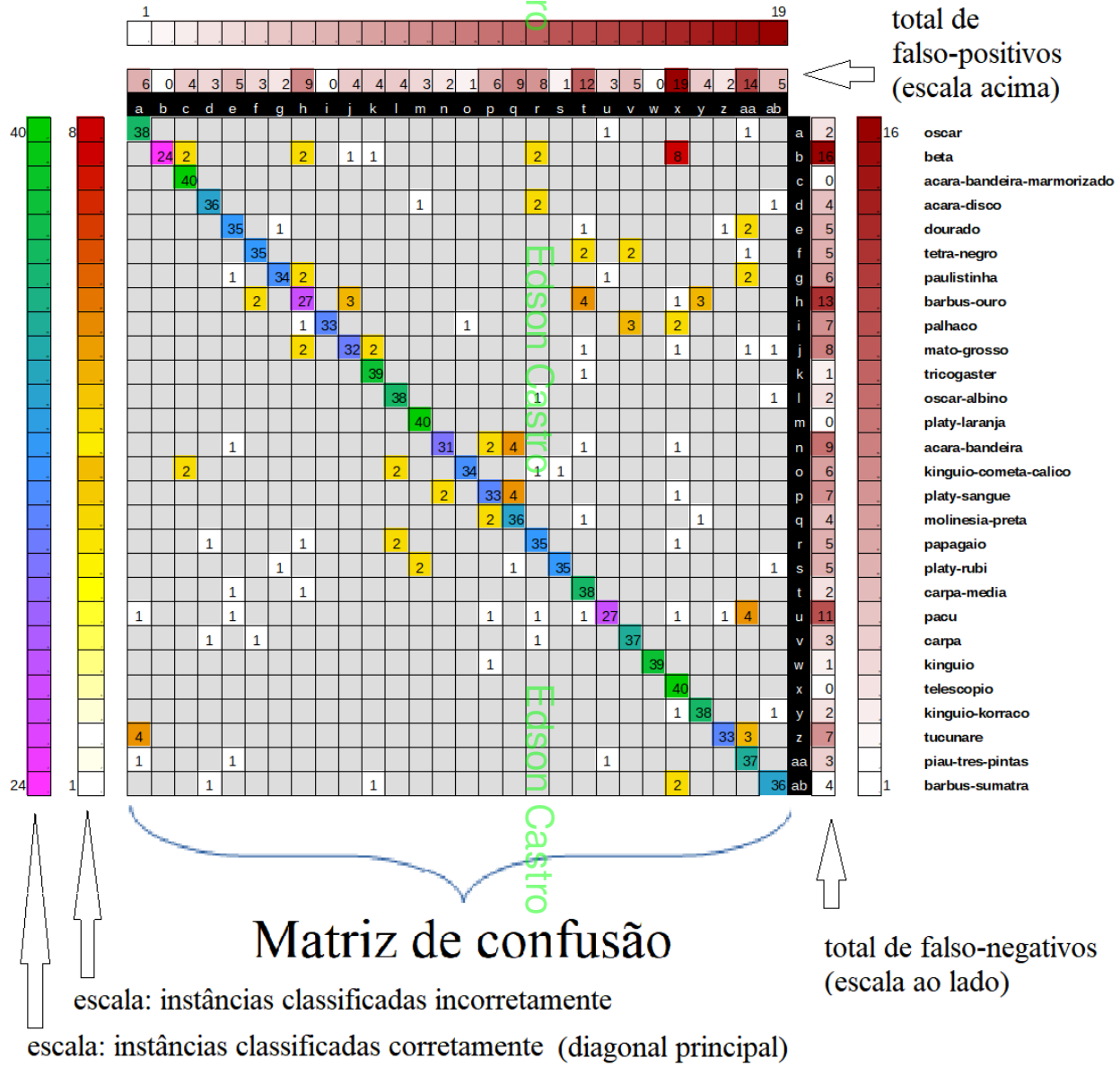


Figura 8: Matriz de confusão para o algoritmo IBk com realce de regiões importantes de análise. (base de dados sem pré-processamento).
Fonte: autoria própria.

Analisando-se a Figura 8 constata-se que muita informação pode ser extraída de uma matriz de confusão. Nesta figura destacaram-se os elementos da diagonal principal, que representam as instâncias classificadas corretamente. Estabeleceu-se uma escala de cor que permite a visualização gradual das instâncias melhores ou piores classificadas. À direita da matriz de confusão listam-se as classes reais de espécies de peixe. Já acima da matriz ilustram as classes que foram classificadas pelos algoritmos. Para exemplificar, analisando-se a linha identificada com a letra ***h***, que refere-se à espécie de peixe *barbus-ouro*, nota-se que das 40 instâncias que deveriam ser classificadas como desta classe, apenas 27 foram corretamente classificadas (estão na diagonal).

As matrizes de confusão foram geradas utilizando-se o Libreoffice CALC. Optou-se por omitir todos os valores zerados para melhorar a visualização das demais informações.

Todos os valores da matriz que não pertencem à diagonal principal representam erros de classificação. Estabeleceu-se também uma segunda escala de cor que permite a visualização gradual das instâncias classificadas incorretamente. Os valores de uma linha da matriz de confusão que não pertencem à diagonal principal representam falso-negativos. O total de falso-negativos para cada classe foi apresentado à direita da matriz de confusão. Já os valores de uma coluna da matriz que não pertencem à diagonal principal representam falso-positivos. Listam-se o total de falso-positivos para cada classe acima da matriz de confusão. Apresenta-se também uma escala de cor que permite a visualização gradual dos falso-negativos e outra para os falso-positivos.

Continuando o exemplo da espécie de peixe *barbus-ouro*, classe da linha ***h***, observa-se um total de 13 falso-negativos, dos quais 4 foram classificados como sendo da classe t, que refere-se à espécie de peixe *carpa-media*. Constata-se também que 9 instâncias de outras classes foram classificadas como sendo da classe *barbus-ouro*, ou seja, representam falso-positivos.

A Figura 9 ilustra um comparativo de desempenho do Algoritmo IBk, através de matrizes de confusão, com a base de dados sem/com pré-processamento. Como dito anteriormente, a etapa de pré-processamento melhorou o desempenho preditivo geral do IBk. Para exemplificar, analisando-se a linha identificada com a letra **b**, que refere-se à espécie de peixe *beta*, nota-se que das 40 instâncias que deveriam ser classificadas como desta classe, sem a etapa de pré-processamento apenas 24 foram corretamente classificadas e com a etapa de pré-processamento 34 instâncias tiveram êxito de classificação, ou seja, o percentual de classificação correta do IBk para a classe *beta* subiu de 60% para 85%, o que é um incremento substancial. Todavia, o ganho de desempenho não ocorreu para todas as classes e para exemplificar toma-se a classe *oscar* identificada pela linha de letra **a**, que teve uma redução expressiva no seu poder preditivo: enquanto na base sem pré-processamento apenas duas instâncias foram incorretamente classificadas, na base pré-processada esse número subiu para 8 e além disto o número de falso-positivos aumentou consideravelmente de 6 para 17. Nas Figuras 9, 10 e 11 comparam-se os desempenhos dos outros três algoritmos.

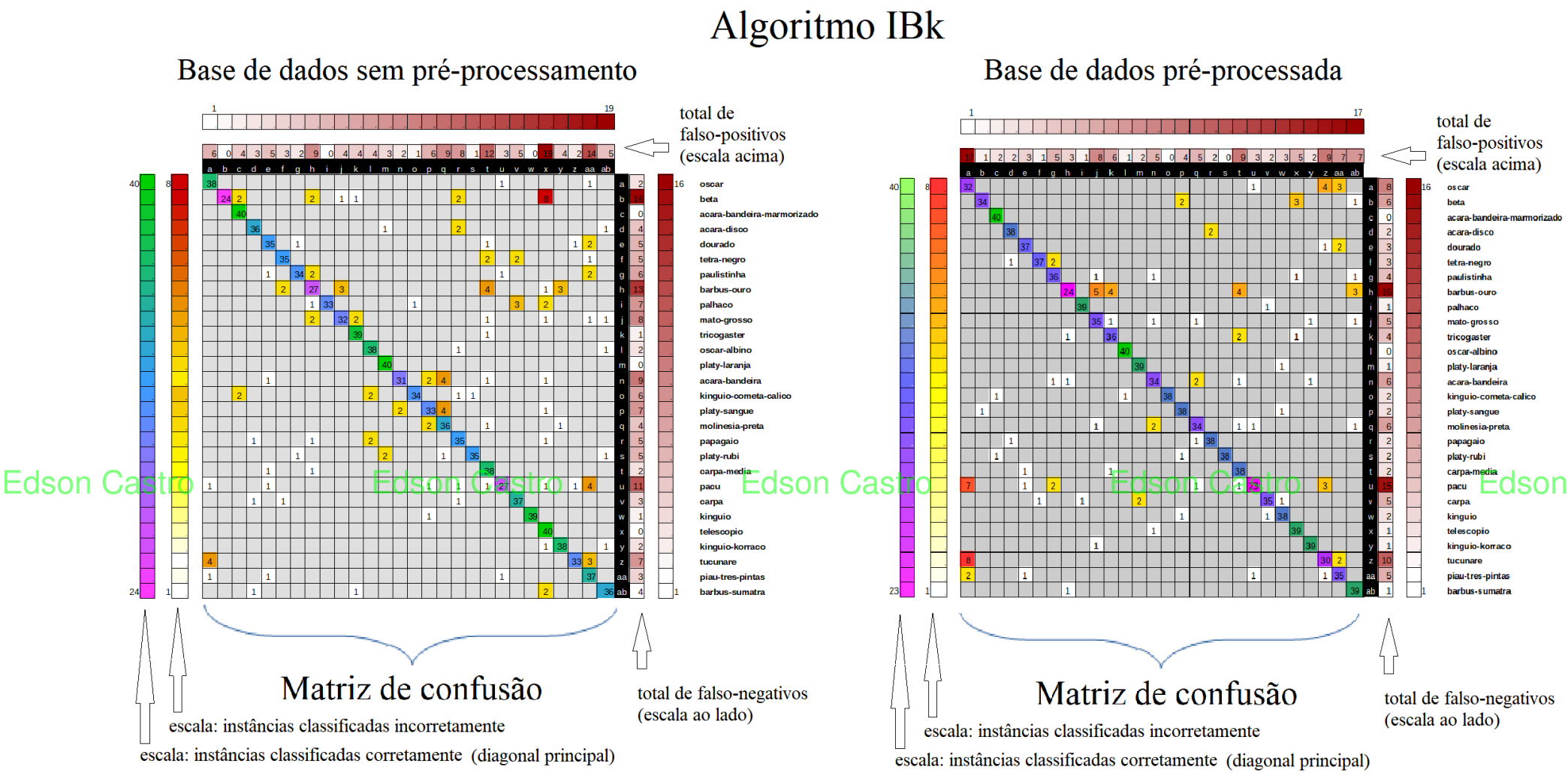


Figura 9: Comparativo de desempenho do Algoritmo IBk, através de matrizes de confusão, com a base de dados sem/com pré-processamento.
Fonte: autoria própria.

Algoritmo J48

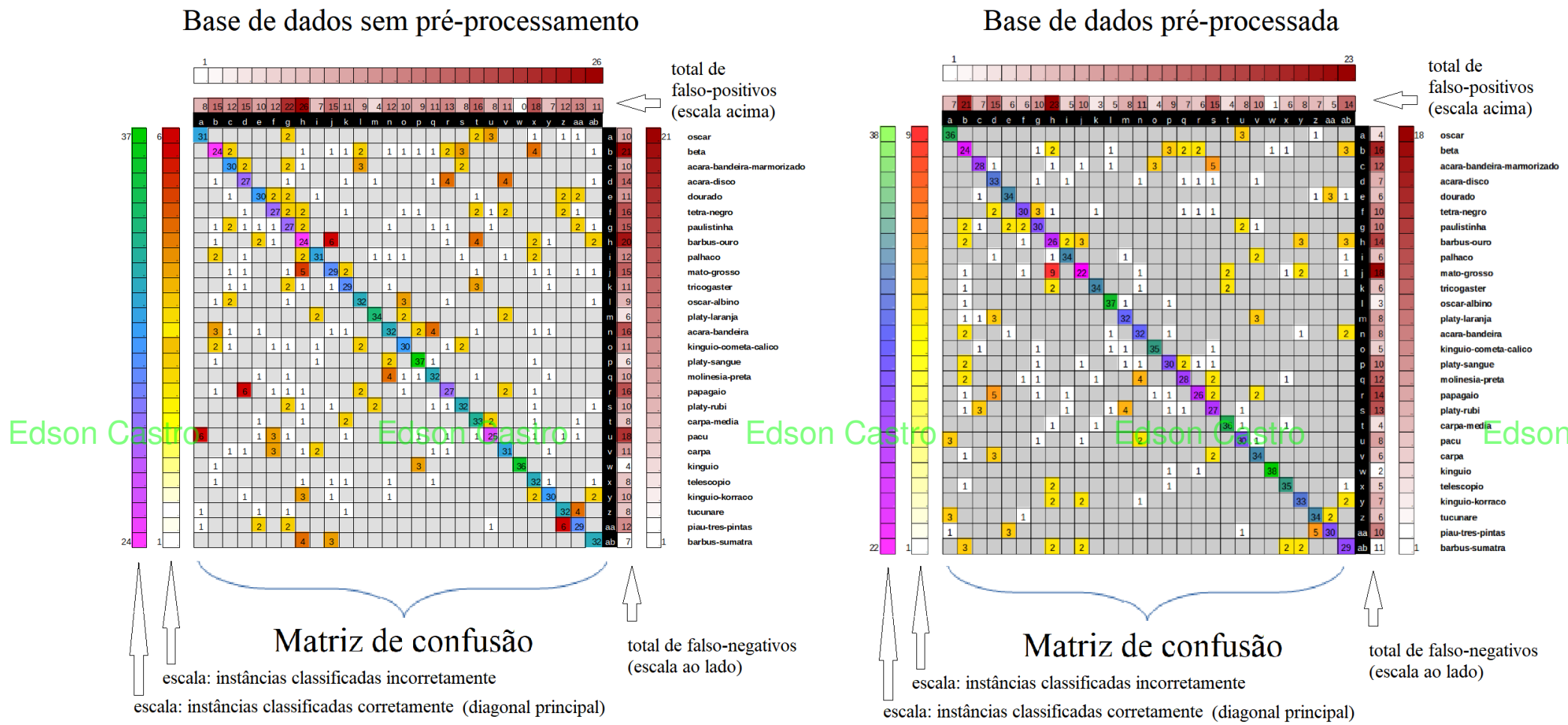


Figura 10: Comparativo de desempenho do Algoritmo J48, através de matrizes de confusão, com a base de dados sem/com pré-processamento.
Fonte: autoria própria.

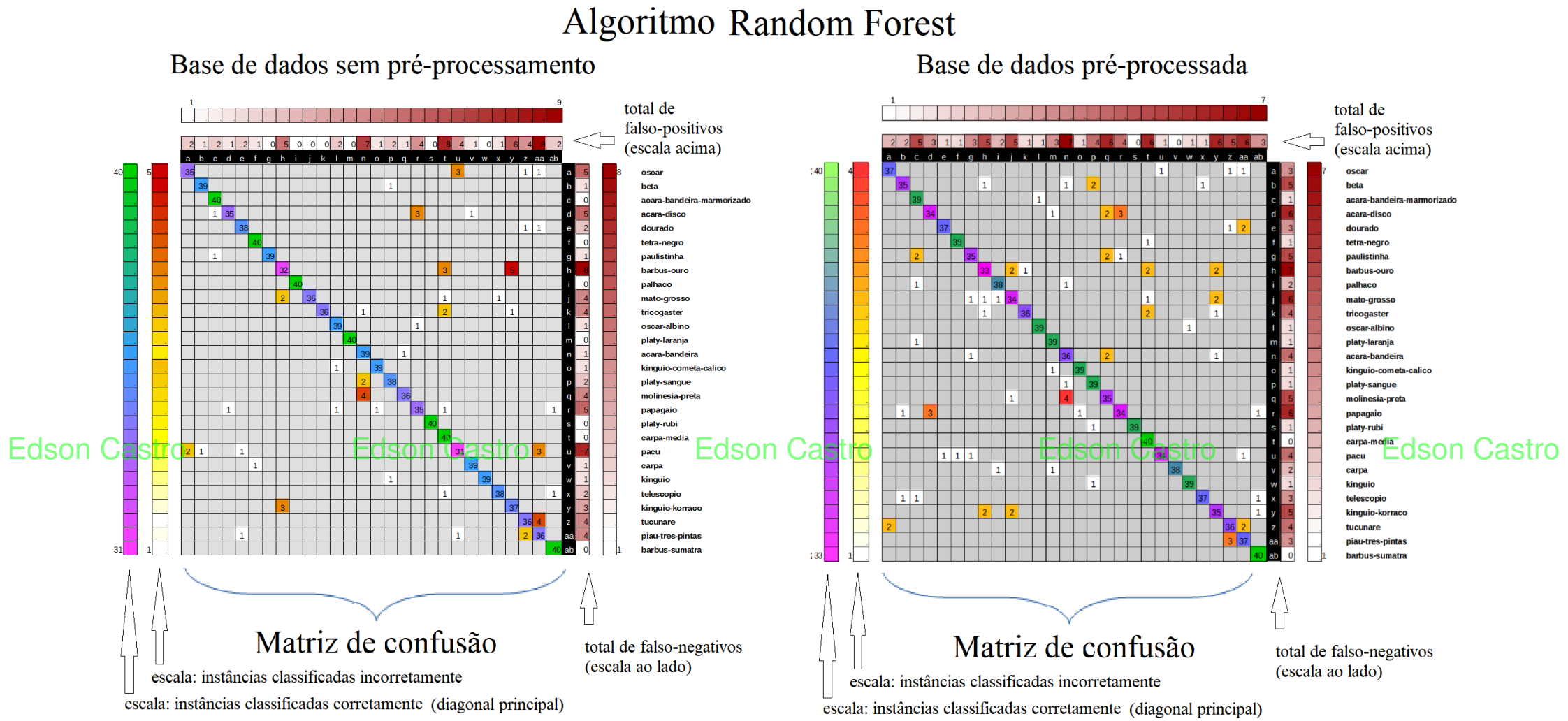


Figura 11: Comparativo de desempenho do Algoritmo Random Forest,
através de matrizes de confusão, com a base de dados sem/com pré-processamento.
Fonte: autoria própria.

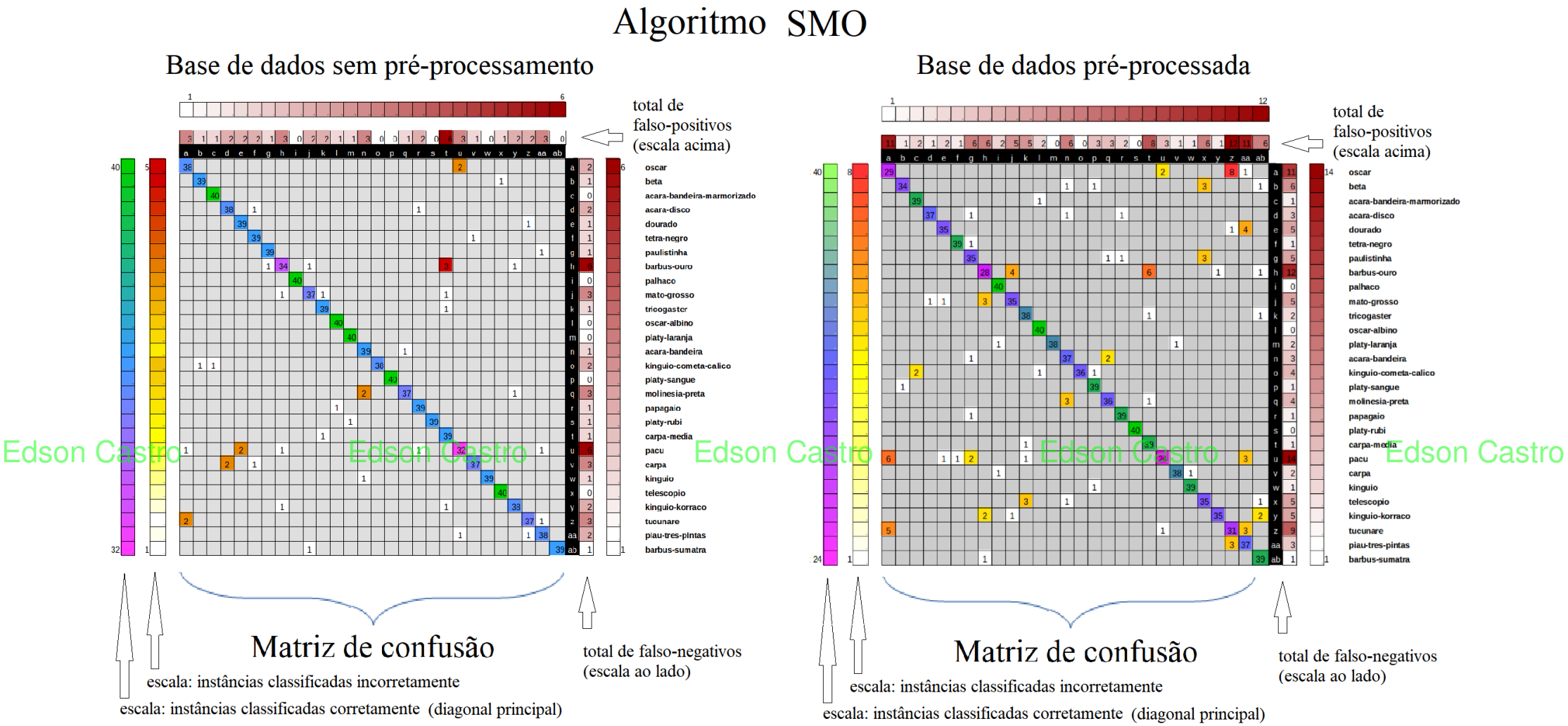
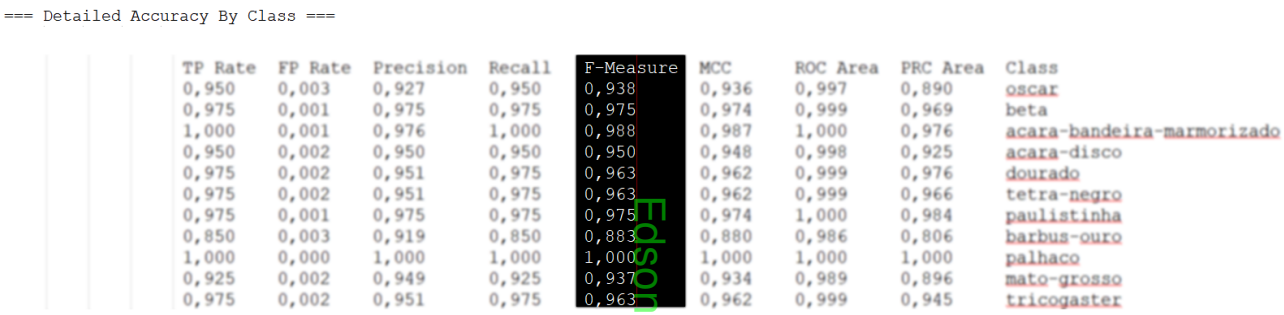


Figura 12: Comparativo de desempenho do SMO, através de matrizes de confusão, com a base de dados sem/com pré-processamento.
Fonte: autoria própria.

Comparando-se os desempenhos dos Algoritmos através das matrizes de confusão apresentadas nas Figuras 9, 10, 11 e 12, nota-se que cada uma das quatro técnicas de aprendizado possuem respostas diferentes de classificação. Os resultados mudaram de algoritmo para algoritmo e também quando se aplicou ou não técnicas de pré-processamento. Lembrando que todos os valores de uma matriz de confusão que não pertencem à diagonal principal representam erros de classificação, nota-se, numa rápida análise destas figuras, que os algoritmos IBk, Random Forest e SMO apresentaram desempenho de classificação muito superior ao J48. Este fato corrobora as estatísticas apresentadas na Figura 6, na qual apresenta estes algoritmos com maiores percentuais de instâncias classificadas corretamente. Nota-se também, pelas análises destas matrizes de confusão que o desempenho destes foi melhor não só no geral, mas também em predizer diversas classes em específico. Esse tipo de análise será melhor explorado adiante com testes de hipóteses ANOVA e o pós-teste de Tukey.

Para as atividades descritas a seguir utilizou-se o software Rstudio e a linguagem de programação R, que é dedicada à computação estatística e gráficos, largamente utilizada na mineração de dados e para desenvolvimento de software estatísticos.

Antes de efetuar as análises e os testes de hipóteses necessitou-se gerar um arquivo auxiliar, que foi utilizado como entrada para o programa em R, e contém os dados de desempenho das quatro técnicas de aprendizado, para os dois conjuntos de dados. A Figura 13 exemplifica como se deu a geração deste arquivo. Já a figura 14 apresenta um exemplo do conteúdo deste arquivo.



=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0,950	0,003	0,927	0,950	0,938	0,936	0,997	0,890	oscar
0,975	0,001	0,975	0,975	0,975	0,974	0,999	0,969	beta
1,000	0,001	0,976	1,000	0,988	0,987	1,000	0,976	acara-bandeira-marmorizado
0,950	0,002	0,950	0,950	0,950	0,948	0,998	0,925	acara-disco
0,975	0,002	0,951	0,975	0,963	0,962	0,999	0,976	dourado
0,975	0,002	0,951	0,975	0,963	0,962	0,999	0,966	tetra-negro
0,975	0,001	0,975	0,975	0,975	0,974	1,000	0,984	paulistinha
0,850	0,003	0,919	0,850	0,883	0,880	0,986	0,806	barbus-ouro
1,000	0,000	1,000	1,000	1,000	1,000	1,000	1,000	palhaco
0,925	0,002	0,949	0,925	0,937	0,934	0,989	0,896	mato-grosso
0,975	0,002	0,951	0,975	0,963	0,962	0,999	0,945	tricogaster

Figura 13: Fragmento da seção de detalhamento da acurácia por classe, gerada pelo Weka. Utilizou-se o campo F_Measure (*Medida-F*) para gerar um arquivo auxiliar, que correlaciona o desempenho das técnicas de aprendizado em relação à predição para as várias classes de espécies de peixes.
Fonte: autoria própria.

Utilizou-se a Medida-F para comparação do desempenho das técnicas de aprendizado supervisionado deste experimento. A Medida-F é uma métrica interessante pois resulta numa média harmônica que leva em consideração tanto a precisão quanto a revocação. Por outro lado, se fosse

utilizada a precisão penalizar-se-iam apenas os falso-positivos, e se fosse utilizada a revocação seriam penalizados apenas os falso-negativos.

Classe_2	classe	algoritmo	F_Measure
OSC	oscar	RandomForest_1	90.9
BET	beta	RandomForest_1	97.5
ABM	acara-bandeira-marmorizado	RandomForest_1	97.6
OSC	oscar	SMO_1	93.8
BET	beta	SMO_1	97.5
ABM	acara-bandeira-marmorizado	SMO_1	98.8
OSC	oscar	J48_1	78.5
BET	beta	J48_1	60.8
ABM	acara-bandeira-marmorizado	J48_1	74.1
OSC	oscar	Ibk_1	90.5
BET	beta	Ibk_1	75.00
ABM	acara-bandeira-marmorizado	Ibk_1	95.2

Figura 14: Exemplo do conteúdo do aquivo auxiliar com o desempenho das técnicas de aprendizado em predizer as várias classes de espécies de peixes, gerado a partir das precisões das técnicas, conforme Figura 13. Neste exemplo, ilustram-se apenas três classes por técnica.
O campo classe_2 é uma abreviatura da classe.
Fonte: autoria própria.

Segundo Ross (2004, p. 27) um *boxplot*, ou diagrama de caixa e bigode, é frequentemente utilizado para representar graficamente o resumo das estatísticas de um conjunto de dados. A Figura 15 exemplifica um *boxplot*.

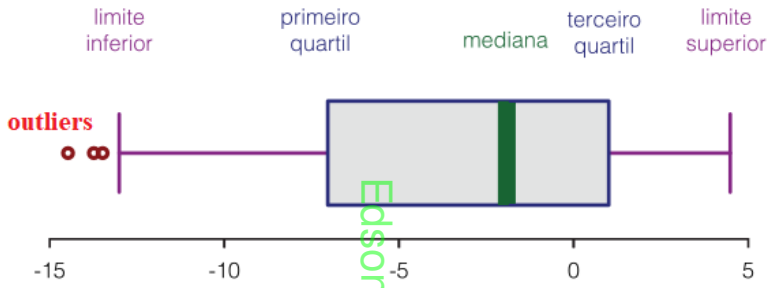


Figura 15: Elementos de um boxplot.
Fonte: adaptado de Wikipedia.

Para compreender como um *boxplot* permite resumir a visualização do conjunto de dados, observe também a Figura 16, com comentários a seguir.

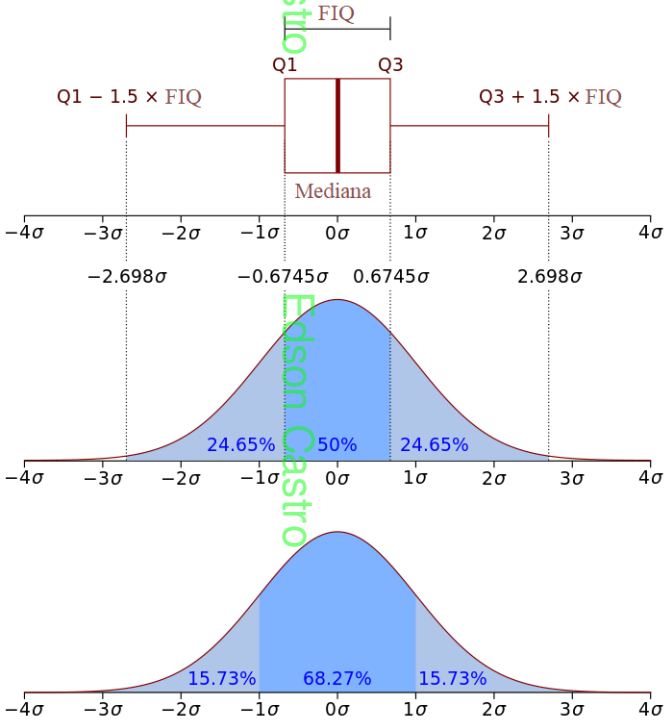


Figura 16. *Boxplot* e função densidade de probabilidade de uma população normal.
σ refere-se a um desvio padrão em relação à mediana.
Fonte: adaptada de Wikipedia.

Ross (2004, p. 25-26) apresenta a definição de “quartis”, importante para construção dos *boxplots*. Ele afirma que os quartis dividem um conjunto de dados em quatro subconjuntos, de maneira que cerca de 25% dos dados encontram-se em cada quartil, ou seja, cerca de 25% dos dados encontram-se abaixo do primeiro quartil, outros 25% entre o primeiro e o segundo, e assim por diante, como observado na Figura 16.

Observando-se as Figuras 15 e 16, nota-se que um *boxplot* é construído em torno da mediana. A mediana pode ser entendida, de maneira simples, como o valor central de um conjunto de dados. Diferente da média, a mediana é considerada uma medida de dispersão estatística robusta, pouco influenciada por *outliers*. Em estatística, *outilier* é um valor atípico, normalmente inconsistente, e que dista demasiadamente dos outros valores da população amostrada.

Ross (2004, p. 25) afirma que o percentil 50 da amostra representa a mediana da amostra. Na construção de um *boxplot* cria-se uma “caixa” que se inicia no primeiro e finda no terceiro quartil, e destaca-se com uma linha o valor do segundo quartil, que representa a mediana amostral.

Para definição dos limites inferior e superior de um *boxplot*, ilustrados nas Figuras 15 e 16, é importante o conceito de faixa interquartil (FIQ), que refere-se ao tamanho da caixa, que vai do primeiro ao terceiro quartil. Conforme observado na Figura 16, para chegar nestes dois limites aplicam-se um deslocamento no valor de 1,50xFIQ, em relação ao primeiro e ao terceiro quartil.

Num *boxplot* os valores menores que o limite inferior ou maiores que o limite superior são considerados outliers.

A Figura 17 ilustra os *boxplots* resultantes da análise de desempenho das quatro técnicas de aprendizado de máquina supervisionada em predizer espécies de peixes.

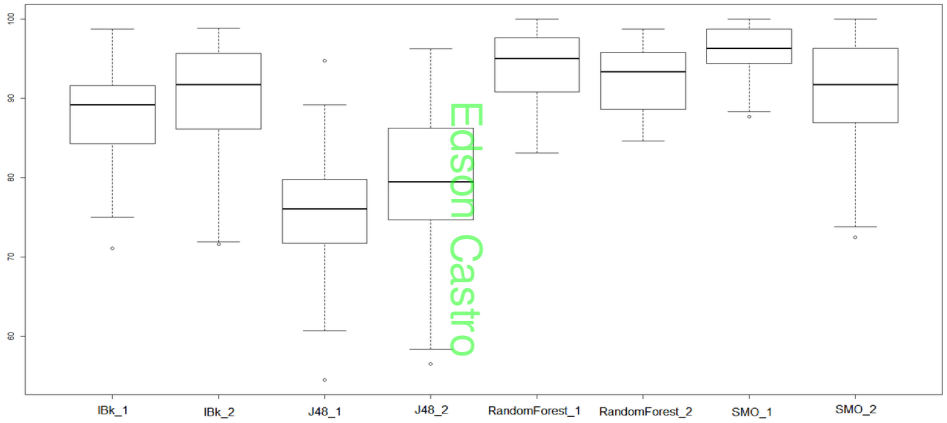


Figura 17: *Boxplots* resultantes da análise de desempenho dos quatro algoritmos de aprendizado de máquina em predizer espécies de peixes. Os *sufixos* _1 e _2 nos nomes dos algoritmos representam conjuntos de dados sem pré-processamento e processados, respectivamente.
Fonte: autoria própria – gerados a partir do Rstudio.

Necessitou-se instalar dois pacotes adicionais no software RStudio: o ***forcats*** que provê ferramentas para resolver problemas relacionados a variáveis categóricas e o ***ggplot2***, que permite gerar visualizações de dados com um alto nível de abstração.

A Figura 18 ilustra *boxplots* também resultantes da análise de desempenho das quatro técnicas de aprendizado de máquina supervisionada em predizer espécies de peixes, gerados com o pacote ***ggplot2***, vide observações na Figura.

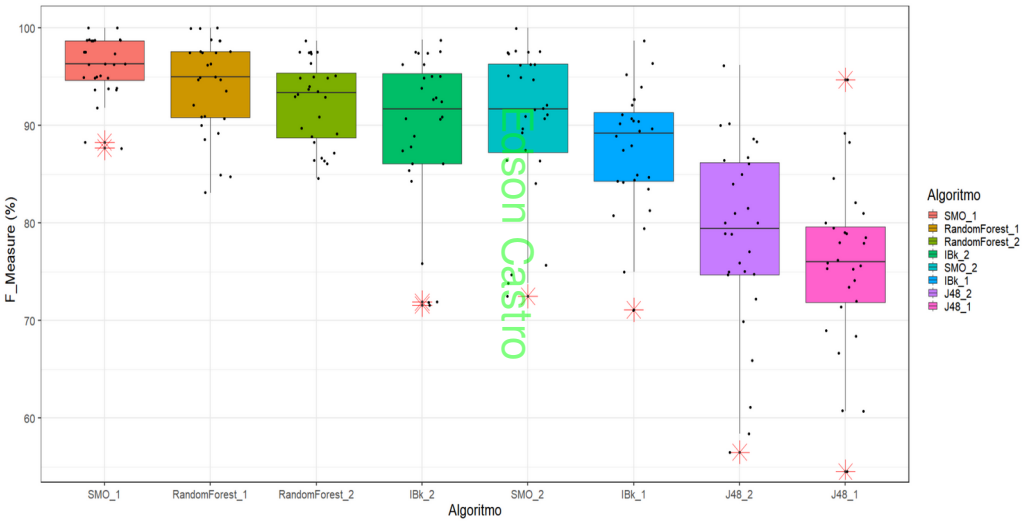


Figura 18: *Boxplots* resultantes da análise de desempenho dos quatro algoritmos de aprendizado de máquina em predizer espécies de peixes, gerados com o pacote ***ggplot2***. Os *boxplots* estão ordenados descendentemente pela mediana. Apresentam-se as caixas coloridas. Apresentam-se

também todos os valores de desempenho observados para as instâncias. Destacam-se também os *outliers* com asteriscos em vermelho.

Fonte: autoria própria – gerado a partir do Rstudio.

Analisando-se os *bloxplots* resultantes da Figura 19, nota-se que os algoritmos SMO_1 e RandomForest_1 apresentaram praticamente a mesma mediana. Todavia, a amplitude dos dados resultantes do SMO_1 é menor que do que aqueles resultantes do RandomForest_1. Notam-se dois *outliers* para o algoritmo SMO_1, cujo valor representa um desempenho igual ou superior a três classificações pelo RandomForest_1, que estavam dentro dos limites do *boxplot* e para aquela distribuição não foram considerados *outliers*.

A ordenação descendente pela mediana, apresentado na Figura 19, facilita a comparação de desempenho dos algoritmos. Nota-se que o desempenho mediano, levando em consideração a técnica da medida F, foi de cerca de 95% para os algoritmos SMO_1, RandomForest_1 e RandomForest_2. Nota-se também que com os dados pré-processados os algoritmos IBk_2 e SMO_2 tiveram desempenho mediano muito próximo além de distribuições (tamanhos das caixas) muito parecidas. Pelos *boxplots* nota-se que o algoritmo J48 apresentou desempenho bastante inferior aos demais. *Será que esta diferença é significativa?* Esta pergunta será respondida adiante, através dos Testes de Hipóteses ANOVA e o pós-teste de Tukey.

Através dos *boxplots* é possível analisar a simetria das distribuições. O J48_1 apresentou distribuição praticamente simétrica, pois a mediana encontra-se praticamente no centro da *caixa* (retângulo). Já o IBk_1 apresentou uma distribuição assimétrica, pois a mediana encontra-se mais próxima do terceiro quartil do que do primeiro.

A Figura 19 ilustra *boxplots* também resultantes da análise de desempenho de predição para cada espécie de peixe, gerados com o pacote ***ggplot2***, vide observações na Figura.

Analisando-se os *bloxplots* resultantes da Figura 19, nota-se inicialmente que algumas espécies de peixe são bem mais fáceis de se predizer do que algumas outras. Isto se explica facilmente analisando os valores medianos e as amplitudes dos *boxplots*. A ordenação descendente pela mediana ajudou consideravelmente a análise de desempenho das predições, em função do grande número de classes a analisar: 28 espécies de peixes, conjuntos sem pré-processamento e processados, que equivale a 56 resultados.

No tocante à simetria das distribuições constatou-se tanto distribuições simétricas quanto assimétricas em relação aos valores medianos.

Além dos *outliers*, representados com asteriscos vermelhos na Figura 19, as classes cujas distribuições de dados possuem grande amplitude indicam que o desempenho dos algoritmos em predizê-las possivelmente diferem significativamente.

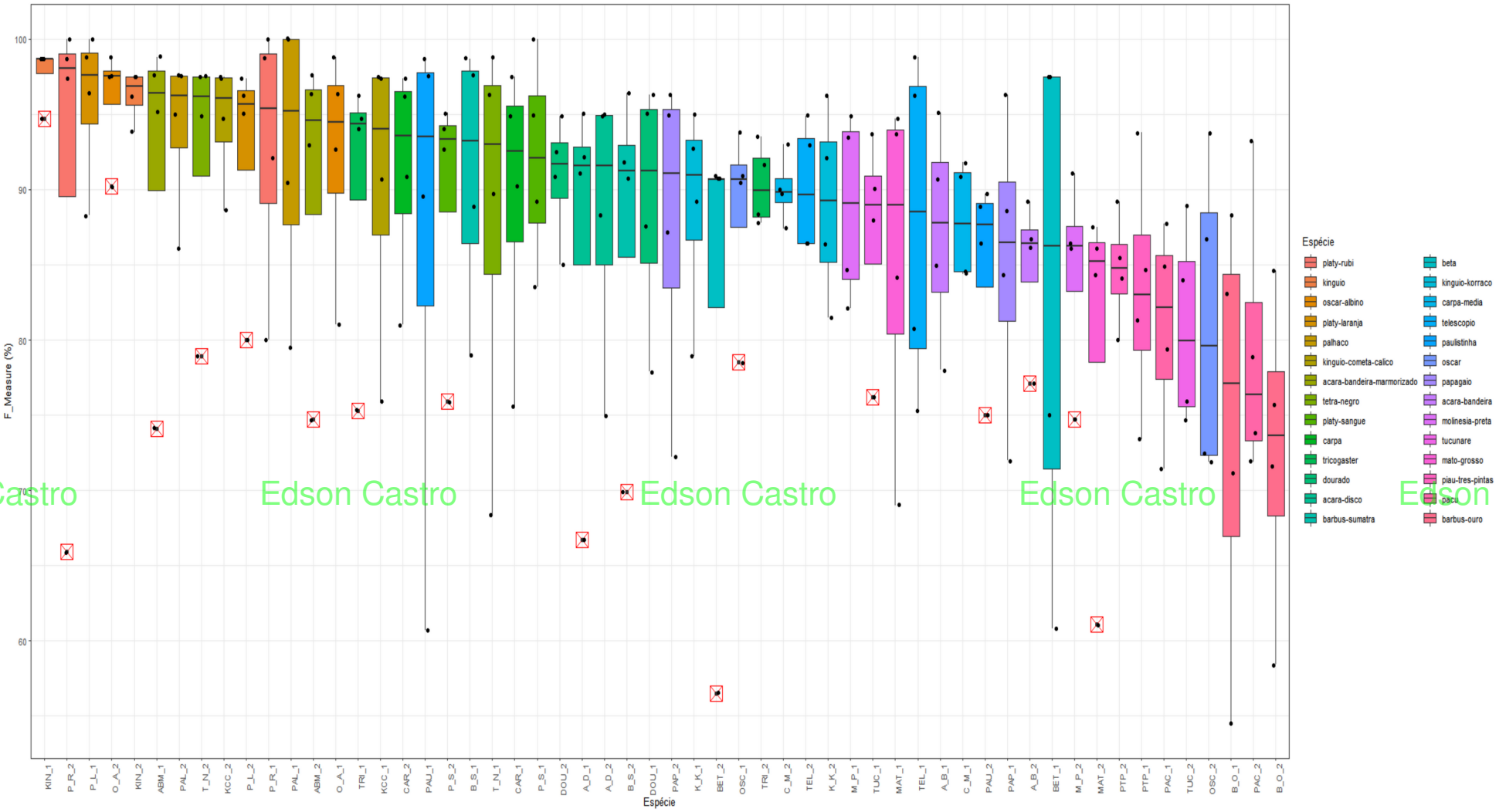


Figura 19: *Boxplots* resultantes da análise de desempenho em predizer as espécies de peixes, gerados com o pacote ***ggplot2***. Os *boxplots* estão ordenados descendentemente pela mediana. Apresentam-se as caixas coloridas. Apresentam-se também todos os valores de desempenho observados para as instâncias. Destacaram-se também os *outliers* com asteriscos em vermelho. Os *sufixos* _1 e _2 nas siglas das classes representam conjuntos de dados sem pré-processamento e processados, respectivamente.
Fonte: autoria própria – gerada a partir do Rstudio.

Embora seja muito fácil concluir, apenas pela análise dos *boxplots*, que o desempenho em predizer algumas classes foi significativamente melhor que algumas outras, como por exemplo as duas primeiras em relação às duas últimas, noutros casos necessitam-se de ferramentas estatísticas para efetuar tal conclusão. Para tal, neste experimento utilizaram-se dos Testes de Hipóteses ANOVA e do pós-teste de Tukey, que serão discutidos a seguir.

Segundo Ross (2004, p. 440) através dos testes de hipóteses ANOVA, no qual efetuam-se análise de variança, é possível testar se as médias de várias populações são estatisticamente iguais. Neste tipo de testes de hipóteses Ross (2004, p.442) afirma que necessitam-se de uma hipótese nula (H_0), de que todas as médias populacionais são iguais, e de uma hipótese alternativa (H_1 ou H_a), na qual pelo menos uma das médias difere significativamente das demais. Supondo que ***m*** é o número de populações, ele escreve:

$$\mathbf{H_0}: \mu 1 = \mu 2 = \cdots = \mu m$$

versus

$$\mathbf{H_1}: \text{nem todas as médias são iguais}$$

Segundo Ross (2004), neste tipo de teste de hipóteses, analisam-se as variâncias entre as classes e as variâncias entre as instâncias. A comparação do valor P (***p-value***) com o valor de significância estatística esperado, é apontado por Ross (2004, p. 446) como uma das formas de se efetuar este tipo de teste de hipóteses.

O valor P, descrito na forma de probabilidade, estima uma medida de quão evidente é a *rejeição da hipótese nula*. Quanto menor o valor de ***p-value*** maior a probabilidade de que a hipótese nula deva ser descartada.

A Tabela 1 apresenta o resumo dos resultados dos testes de hipóteses ANOVA para este experimento de classificação de espécies de peixes.

Tabela 1: Resultado dos testes de hipóteses ANOVA neste problema de classificação de espécies de peixes, com as quatro técnicas de aprendizado de máquina supervisionada.

Pré-processamento	**Classe/Algoritmo**	Graus de liberdade	Valor F	P-value `Pr(>F)`
NÃO	**dados$classe** (espécies de peixe)	27	4,03	6.14**e**-07 ***
	dados$algoritmo (técnica de aprendizado)	3	110,23	< 2**e**-16 ***
SIM	**dados$classe** (espécies de peixe)	27	3,24	2,41**e**-05 ***
	dados$algoritmo (técnica de aprendizado)	3	28,205	1,38**e**-12 ***

Signif. codes: 0 '*** ' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Fonte: autoria própria – estatísticas geradas no Rstudio.

Costuma-se assumir que valores de *p-value* menores que 0,001 indicam forte evidência para rejeição da hipótese nula. Analisando-se a Tabela 1, nota-se que os valores de *p-value* indicam claramente que as hipóteses nula devam ser descartadas, tanto em relação às espécies de peixe quanto em relação às técnicas de aprendizado, e esta constatação vale tanto para o conjunto sem pré-processamento quanto para o conjunto pré-processado. Nota-se também que o software R apresenta legenda para o nível de significância resultante; por ela entende-se *** como 0 de significância, ou 100% de confiança. Entende-se, daí, que existe diferença significativa em, pelo menos, uma das técnicas de aprendizado utilizadas. Contudo, a análise de variância ANOVA não indica quais tratamentos (classes) apresentam tais diferenças, de forma que necessitam-se de uma análise complementar, que neste experimento foi feita através do pós-teste de Tukey.

O pós teste de Tukey necessita do teste de hipóteses ANOVA como entrada. Este teste estatístico verifica se as diferenças são honestamente significativas (do inglês, *Honestly Significant Difference* – HSD). Para tal, leva em consideração o número de tratamentos (classes), o número de graus de liberdade (que depende do número de instâncias), a média dos quadrados entre tratamentos (já constante nos resultados da ANOVA) e o número de participantes em cada grupo.

A Figura 20 apresenta o resumo dos resultados do pós teste de Tukey para este experimento de classificação de espécies de peixes, nos quais comparam-se, duas a duas, as técnicas de classificação.

Técnicas Comparadas	Diferença	Mínimo	Máximo	Valor P ajustado (p adj)
SMO_1-J48_1	20,200	15,832	24,568	0,0000000
RandomForest_1-J48_1	18,336	13,968	22,703	0,0000000
SMO_1-J48_2	17,443	13,075	21,811	0,0000000
RandomForest_2-J48_1	16,646	12,279	21,014	0,0000000
RandomForest_1-J48_2	15,579	11,211	19,946	0,0000000
SMO_2-J48_1	14,582	10,214	18,950	0,0000000
RandomForest_2-J48_2	13,889	9,522	18,257	0,0000000
SMO_2-J48_2	11,825	7,457	16,193	0,0000000
J48_2-IBk_1	-9,061	-13,428	-4,693	0,0000000
J48_2-IBk_2	-11,054	-15,421	-6,686	0,0000000
J48_1-IBk_1	-11,818	-16,186	-7,450	0,0000000
J48_1-IBk_2	-13,811	-18,178	-9,443	0,0000000
SMO_1-IBk_1	8,382	4,014	12,750	0,0000005
RandomForest_1-IBk_1	6,518	2,150	10,886	0,0002291
SMO_1-IBk_2	6,389	2,022	10,757	0,0003344
SMO_2-SMO_1	-5,618	-9,986	-1,250	0,0028176
RandomForest_2-IBk_1	4,829	0,461	9,196	0,0189839
RandomForest_1-IBk_2	4,525	0,157	8,893	0,0363969
SMO_2-RandomForest_1	-3,754	-8,121	0,614	0,1503560
SMO_1-RandomForest_2	3,554	-0,814	7,921	0,2044829
RandomForest_2-IBk_2	2,836	-1,532	7,203	0,4913406
SMO_2-IBk_1	2,764	-1,603	7,132	0,5253573
J48_2-J48_1	2,757	-1,611	7,125	0,5287769
SMO_2-RandomForest_2	-2,064	-6,432	2,303	0,8331377
IBk_2-IBk_1	1,993	-2,375	6,361	0,8570156
SMO_1-RandomForest_1	1,864	-2,503	6,232	0,8947948
RandomForest_2-RandomForest_1	-1,689	-6,057	2,678	0,9352191
SMO_2-IBk_2	0,771	-3,596	5,139	0,9994101

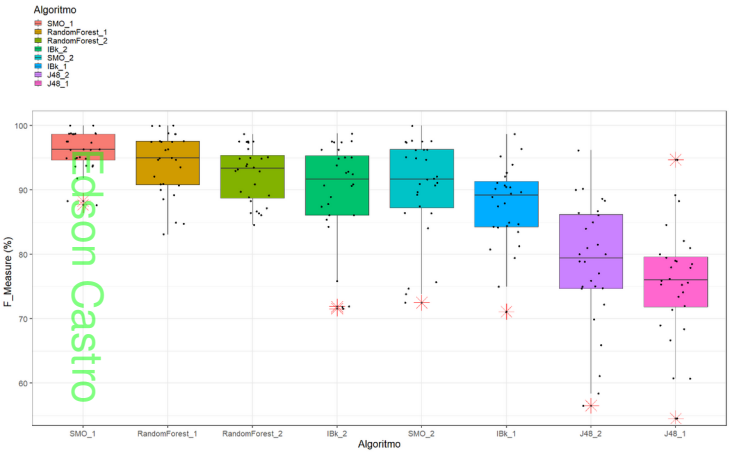


Figura 20: À esquerda os resultados do pós teste de Tukey para este experimento de classificação de espécies de peixes. Os valores mínimos, máximos e da diferença foram arredondados para 3 casas.

Ordenaram-se os dados pelo Valor P ajustado e pela Diferença (ordem descendente). Na direita a Figura 18 é reapresentada para facilitar a análise.
Fonte: autoria própria – estatísticas geradas no Rstudio.

Para interpretação do resultado do pós teste de Tukey pode-se utilizar o **Valor P ajustado**, análogo ao valor P (**p-value**) do teste de Hipóteses ANOVA, só que ajustado para múltiplas comparações. Analisando-se a Figura 20 nota-se que existem diferenças significativas de desempenho nas técnicas de aprendizado de máquina supervisionada em predizer espécies de peixes. Nota-se que em dezoito entradas da tabela constante nesta figura as técnicas comparadas apresentam **Valor P ajustado** próximos de zero e em todos esses casos pode-se afirmar qual o melhor algoritmo de aprendizagem de máquina. Nesses 18 casos, basta observar o valor da Diferença observada na tabela: caso positiva, o algoritmo à esquerda possui melhor desempenho e caso negativa, o algoritmo à direita é melhor. No caso da comparação entre os algoritmos SMO_2 (com base pré-processada) e RandomForest_1 (com dados sem pré-processamento) o Valor P ajustado é de 0,150356, o que equivale dizer que eles apresentam desempenho igual num nível de significância de 15,03%, ou seja, RandomForest_1 é melhor que SMO_2 com apenas 84,97% de confiança. Num último exemplo pode-se afirmar que os algoritmos SMO_2 e IBk_2 apresentam desempenho estatisticamente semelhantes, pois o Valor P ajustado observado é de 99,94%.

Por fim efetuou-se um comparativo para verificar se ocorreu aumento de desempenho preditivo dos classificadores ou diminuição destes com o processo de pré-processamento. O resultado deste comparativo encontra-se na Tabela 2. Para gerá-la observaram-se, para cada classe e para cada classificador, o valor da Medida-F depois e antes do pré-processamento e tomou-se a diferença dos percentuais observados. Estabeleceu-se uma escala de cor para facilitar a visualização do aumento ou da diminuição de desempenho preditivo.

Embora pela análise da Tabela 2 nada se possa dizer acerca do desempenho preditivo dos classificadores em relação aos demais classificadores, por outro lado, nela se consegue verificar qual a influência da presença ou falta do referido pré-processamento no desempenho de cada classificador em específico, em predizer as diversas classes. Como mencionado acerca da Figura 6, que sintetiza as porcentagens de instâncias classificadas correta e incorretamente, o J48 e o IBk tiveram melhora preditiva decorrente do pré-processamento dos dados. O algoritmo SMO foi o que teve maior influência negativa do processamento dos dados, e com exceção das espécies *platy-rubi* e *carpa* todas as demais tiveram piora de desempenho, que pela análise das matrizes de confusão ilustras na Figura 12 ocorreu aumento tanto dos falso-positivos quanto dos falso-negativos. Pela Tabela 2 verifica-se que a espécie de peixe *oscar* é a que teve maior piora de desempenho e pode-se constatar por essas matrizes de confusão que enquanto na base de dados sem pré-processamento o

SMO conseguiu predizer esta espécie com 95% de instâncias classificadas corretamente, após o pré-processamento este percentual caiu para apenas 72,50%.

Tabela 2: Resultados comparativos do aumento/decremento de desempenho preditivo dos classificadores, levando em consideração a Medida-F, com o processo de pré-processamento.

classe	RandomForest	SMO	J48	IBK
oscar	2,80	-21,30	8,20	-18,60
beta	-6,60	-6,80	-4,30	15,70
acara-bandeira-marmorizado	-4,70	-2,50	0,60	2,40
acara-disco	-3,80	-0,10	8,30	3,90
dourado	-0,10	-5,40	7,10	5,00
tetra-negro	-1,30	1,20	10,50	5,20
paulistinha	-9,00	-11,10	14,30	-0,60
barbus-ouro	1,50	-12,60	3,90	0,50
palhaco	-5,00	-2,40	6,60	7,10
mato-grosso	-8,60	-6,20	-7,90	0,10
tricogaster	-1,20	-4,70	13,00	-6,20
oscar-albino	1,20	-1,20	9,20	6,10
platy-laranja	-4,90	-1,40	-8,30	-0,10
acara-bandeira	-4,00	-5,90	-0,90	1,20
kinguio-cometa-calico	0,00	-2,70	12,70	6,70
platy-sangue	-1,00	-4,90	-13,30	9,20
molinesia-preta	-7,10	-3,80	-7,40	1,40
papagaio	-1,40	0,00	0,20	10,70
platy-rubi	-1,30	1,30	-14,10	5,30
carpa-media	2,10	-2,10	5,40	3,00
pacu	8,30	-13,90	7,50	-7,50
carpa	-0,10	1,30	5,40	0,70
kinguio	-1,20	-1,20	1,50	-4,90
telescopio	-1,30	-12,40	11,10	12,10
kinguio-korraco	-2,80	-2,90	2,60	3,60
tucunare	-1,10	-19,00	7,80	-12,10
piau-tres-pintas	4,50	-9,70	6,60	4,10
barbus-sumatra	-1,20	-6,90	-9,10	1,80

Fonte: Autoria Própria.

Neste experimento exploraram-se vários dos principais passos da descoberta de conhecimento envolvidas num processo KDD. Nele efetuaram-se, com o auxílio das ferramentas de apoio à mineração de dados Weka, RapidMiner Studio, IDE RStudio (com linguagem de programação R) e Libreoffice CALC, técnicas de transformação de dados e redução de dimensionalidade, atividades de geração de modelos de classificação utilizando-se de técnicas de aprendizado de máquina e validação dos modelos gerados com apoio de ferramentas estatísticas tais como matrizes de confusão, teste de Hipóteses ANOVA e pós-teste de Tukey.

A utilização de *boxplots* mostrou-se bastante eficaz para representar graficamente os resumos das estatísticas dos conjuntos de dados. Da maneira como foram apresentadas, as matrizes de confusão, apoiadas pelas escalas de cores permitiram visualizar gradualmente as instâncias

melhores ou piores classificadas. Essas técnicas permitiram sumarizar os conhecimentos presentes naqueles conjuntos de dados e interpretá-los com maior facilidade.

Notou-se que cada uma das quatro técnicas de aprendizado apresentam respostas diferentes de classificação, e que no geral, o algoritmo J48 teve desempenho de classificação bastante inferior aos demais algoritmos. Notou-se também que a etapa de pré-processamento incrementou o desempenho preditivo dos algoritmos J48 e IBk enquanto que o SMO e Random Forest apresentaram pior desempenho na base de dados pré-processada.

Conclui-se, finalmente, com base na análise de variança ANOVA que existem diferenças significativas de desempenho nas técnicas de aprendizado de máquina supervisionada em predizer espécies de peixes e que elas variam suas capacidades preditivas com a utilização ou não de pré-processamento dos dados. Com o pós-teste de Tukey verificou-se que é possível afirmar quais algoritmos ou quais classes apresentam capacidades preditiva significantemente melhores que os demais.

Ressalva-se que na prática a escolha de qual algoritmo utilizar pode depender de vários fatores. Pode-se querer privilegiar os algoritmos por alguma determinada métrica: percentual de instâncias classificadas corretamente, percentual de instâncias classificadas erroneamente, percentual de falso-positivos, percentual de falso-negativos, percentual de verdadeiro-positivos, Medida-F, etc. É importante ressaltar que em alguns casos a preferência seja por melhores desempenhos dos algoritmos para todas as classes e em outros a preferência seja por melhores desempenhos em classes específicas.

Referências Bibliográficas

AMARAL, Fernando. Introdução à Ciência de Dados: Mineração de Dados e Big Data. 1. ed. Alta Books Editora, 2016.

ROSS, Sheldon M. Introduction to probability and statistics for engineers and scientists. 3. ed. Academic Press, 2004.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMITH, P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In: Proceedings of the Second International Conference on Data Mining and Knowldege Discovery, AAAI Press, Menlo Park, US; 1996.