Trabalho Prático de Aprendizado de Máquina

Camila A. Paiva[1] e Isela Mendonza[2]

[1,2]Instituto de Computação – Universidade Federal Fluminense (UFF) Niterói – RJ – Brazil

{camilapaiva, imendoza}@id.uff.br

Abstract. *This work aims to show the results gathered from the practical work. It requires the execution of three tasks of data mining on a database file previously chosen. These tasks are: application of an ensemble-based method, application of a non-parametric method, and construction of a bagging-based ensemble. For each task, different approaches and input parameters are required*

Resumo. *Este trabalho tem como objetivo mostrar os resultados obtidos a partir do trabalho prático. Requer a execução de três tarefas de aprendizado de máquina em um arquivo de banco de dados previamente escolhido. Essas tarefas são: aplicação de um métodos baseado em ensemble, aplicação de um método não paramétrico e construção de um enseble baseado em bagging. Para cada tarefa, diferentes abordagens e parâmetros de entrada são necessários.*

1 Introdução

Por milhares de anos, os seres humanos tentam entender como o cérebro é capaz de perceber, entender, prever e manipular um mundo muito maior e mais complicado do que ele. O campo da inteligência artificial, ou IA, vai ainda mais longe: a inteligência artificial visa entender e construir entidades inteligentes [Russell et al. 2010].

A IA é um dos mais novos campos da ciência e engenharia. Russell et al. afirmam que seu desenvolvimento iniciou logo após a Segunda Guerra Mundial e foi nomeado por volta de 1956. Atualmente, tal campo abrange uma grande variedade de subcampos, tais como, provar teoremas matemáticos, diagnosticar doenças, dirigir carros e jogar xadrez. A IA é relevante para qualquer tarefa intelectual, é verdadeiramente um campo universal.

Existem diversas definições para a IA, dentre elas: o estudo dos cálculos que permitem perceber, argumentar e agir [Winston 1992] ou o estudo das faculdades mentais através do uso de modelos computacionais [Winston 1992].

Nos últimos anos a IA ganhou cada vez mais popularidade. Com a globalização e a informatização dos processos, as organizações geram um grande número de dados diariamente, também chamado de *Big Data* [Simon 2015]. Através de campos, como a IA, algoritmos podem ser utilizados para realizar previsões ou sugestões calculadas com base em um conjunto de dados. Alguns dos exemplos mais comuns de aprendizagem de máquina são: os algoritmos utilizados na plataforma da Netflix,

que sugerem filmes baseados em escolhas passadas, ou algoritmos da Amazon que recomendam livros com base em livros comprados anteriormente.

Segundo Russell et al., os algoritmos de aprendizado de máquina podem ser divididos em 3 categorias: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço. Na aprendizagem supervisionada, são apresentados exemplos de entradas e saídas desejadas (pares *input-output*). Dessa forma, tais algoritmos aprendem uma função, ou regra geral, que prevê a saída para novas entradas. Para a categoria aprendizado não supervisionado, nenhum tipo de saída é dado aos algoritmos, deixando-o sozinho encontrar um padrão nas entradas fornecidas. Por fim, na aprendizagem tem como foco a criação de agentes capazes de tomar decisões acertadas em um ambiente sem que se tenha qualquer conhecimento prévio sobre o tal ambiente.

Os Algoritmos de Aprendizagem de Máquina são aplicados em diversas áreas e para diferentes fins, um exemplo dessa aplicação pode ser constatado em [Bramer 2016], o qual aplica a técnica *S-Transform (ST)* para extrair recursos de sons cardíacos. Esses recursos são utilizados como entrada para o classificador, *Multilayer Perceptron Network*. Como resultados, os autores obtiveram 98% de acerto na classificação. Neste mesmo sentido, este trabalho propõe o uso de duas abordagens de aprendizagem de máquina sobre um conjunto de informações obtidas através do repositório de *Machine Learning do UCI* [Dua, D. and Graff, C. 2019]. A aplicação destas abordagens tem como objetivo comparar diferentes técnicas e expor seus respectivos resultados. Dentre as abordagens estão: aplicação de um métodos baseado em ensemble, aplicação de um método não paramétrico e construção de um enseble baseado em bagging.

2 Base De Dados

Como base de dados, utilizou-se um conjunto de informações dermatológicas, denominada como *Dermatology Database*[1]. A base contém 34 atributos, dos quais 33 são de valores lineares e um deles nominal. Tais informações são resultantes de um problema real na dermatologia, o diagnóstico de doenças eritemato escamosas. Estas compartilham características clínicas de eritema e descamação, com pouquíssimas diferenças. As doenças/classes deste grupo são: psoríase, dermatite seborréica, líquen plano, pitiríase rósea, dermatite crônica e pitiríase rubra pilar. A Tabela 1 e Figura 1 apresentam a distribuição das classes.

Normalmente, uma biópsia é necessária para o diagnóstico, mas infelizmente essas doenças também compartilham muitas características histopatológicas. Outra dificuldade para o diagnóstico diferencial é que uma doença pode apresentar as

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

características de outra doença no estágio inicial e pode apresentar as características características nos estágios seguintes.

Código da Classe	Classe	Número de Instâncias
1	Psoríase	112
2	Dermatite Seborréica	61
3	Líquen Plano	72
4	Pitiríase Rósea	49
5	Dermatite Crônica	52
6	Pitiríase Rubra Pilar	20

Tabela 1. Distribuição das Classes.

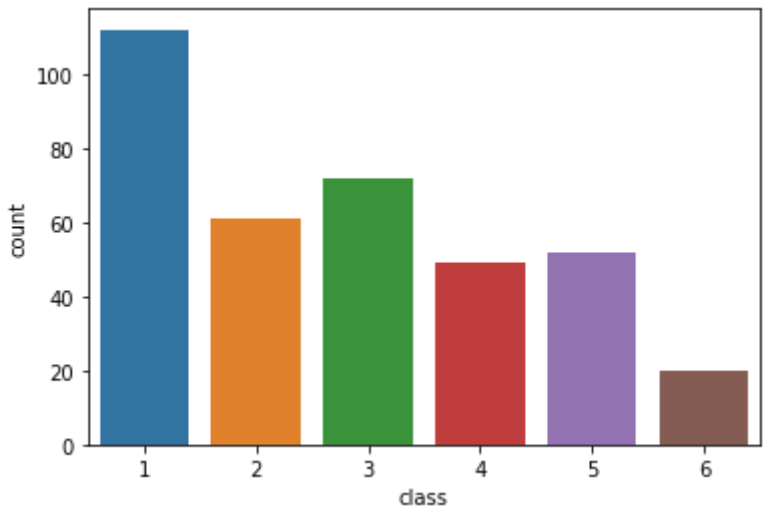


Figura 1. Distribuição das Classes.

Os pacientes foram avaliados clinicamente pela primeira vez através de 12 características, são elas: 1: eritema, 2: dimensionamento, 3: fronteiras definidas, 4: coceira, 5: fenômeno de Koebner, 6: pápulas poligonales, 7: pápulas foliculares, 8: envolvimento da mucosa oral, 9: envolvimento do joelho e cotovelo, 10: envolvimento do couro cabeludo, 11: história familiar, 34: Idade.

Em seguida, amostras de pele foram coletadas para avaliação de 22 características histopatológicas, são elas: 12: incontinência de melanina, 13: eosinófilos no infiltrado, 14: infiltrado PNL, 15: fibrose da derme papilar, 16: exocitose, 17: acantose, 18: hiperqueratose, 19: paraqueratose, 20: baqueteamento dos cumes rete, 21: alongamento das cristas rete, 22: afinamento da epiderme suprapapilar, 23: pústula espongiforme, 24: microabscesso munro, 25: hipergranulose focal, 26:

desaparecimento da camada granular, 27: vacuolização e dano da camada basal, 28: espongiose, 29: aparência de dentes de serra das redes, 30: plug de chifre folicular, 31: paraqueratose perifolicular, 32: infiltrado mononuclear inflamatório, 33: infiltrado tipo banda. Os valores das características histopatológicas são determinados por uma análise das amostras ao microscópio.

Para o conjunto de dados utilizado, o atributo história familiar tem o valor 1 se alguma dessas doenças foi observada na família e 0 caso contrário. O atributo idade representa a idade do paciente. Todas as outras características (clínicas e histopatológicas) receberam um grau entre 0 e 3. Onde, 0 indica que o atributo não estava presente, 3 indica a maior quantidade possível e 1, 2 indicam valores intermediários relativos.

A base possuía alguns dados faltantes relacionados ao atributo idade, 8 tuplas, cerca de 2.19% das informações. Para que o problema fosse resolvido, como pré-processamento, aplicou-se duas técnicas diferentes. A primeira delas foi a remoção dos itens, resultando em 358 tuplas. A segunda foi a substituição dos valores faltantes pela média do atributo idade, resultando em 366 tuplas.

Uma segunda observação realizada sob a base foi a distribuição das classes, como mostra a Tabela 1. Existe um desbalanceamento entre as quantidades de exemplos de cada classe. Com o objetivo de resolver este problemas, aplicou-se o algoritmo de ADASYN. Optou-se pelo uso de tal algoritmo uma vez que as novas amostras são geradas por através de interpolação. E o ADASYN se concentra em gerar amostras próximas às amostras originais que são classificadas erroneamente usando um classificador k-Nearest Neighbors, resultando em 665 tuplas, como mostra a Figura 2. Além disso, optou-se por um algoritmo de oversample, uma vez que o undersampled poderia diminuir a base ao ponto de termos menos de 200 tuplas, que é um dos requisitos do o problema.

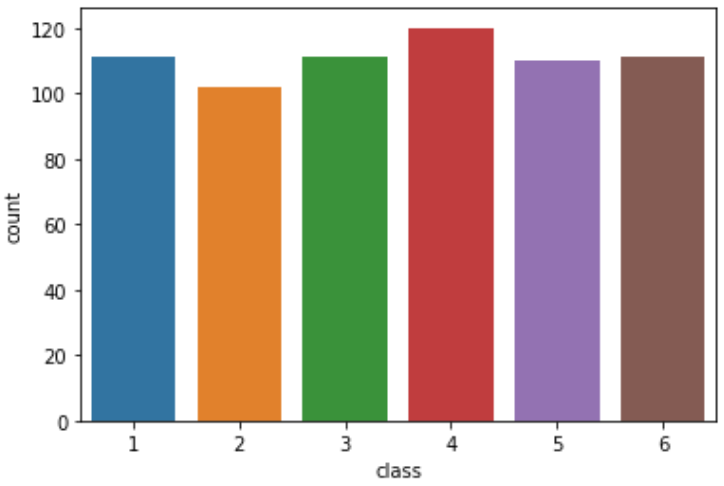


Figura 2. Distribuição das Classes após o balanceamento.

Ao final, ficamos com 3 datastes distintos para aplicarmos os modelos: O primeiro dataset desbalanceado onde retirou-se tuplas com os dados faltantes. O segundo desbalanceado onde substituiu-se pela médias os valores faltantes e, por fim, o terceiro, balanceado onde substituiu-se pela médias os valores faltantes.

3. Método Baseado em Ensemble e Não-paramétrico

3.1 Random Forest

Random Forest é um algoritmo de classificação que utiliza um conjunto de classificadores. Cada classificador é uma árvore de decisão que classifica, independentemente, cada instância e vota em qual classe ela deve ser classificada.

A técnica utiliza um conjunto de N árvores com intuito de ter uma melhor acurácia final. Cada árvore desse conjunto executa a classificação de forma independente e a classe que for escolhida pela maioria das árvores é a classe resultante do processo de classificação.

Para as execuções, ajustou-se um parâmetro com o objetivo de verificar a forma que ele interfere no modelo gerado, tal parâmetro foi o número de árvores. Variou-se o número de árvores entre 5, 10, 30, 50, 100, 500, 1000. Além disso, aplicou-se a técnica simples que modifica as árvores de decisão desbalanceadas através da alteração dos pesos de cada classe, utilizando-se o módulo: 'balanced subsample'.

Para todas as execuções, adotou-se o método *Repeated Stratified Fold* como modelo de avaliação. Optou-se por esse método uma vez que a base de dados é desbalanceada e ele realiza validação cruzada de forma repetida, o que melhora a estimativa de desempenho médio. Foram utilizados os números mais comuns de repetição que são: 3, 5, 10.

Para o primeiro dataset, desbalanceado onde retirou-se tuplas com os dados faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 3. O menor valor de acurácia obtido foi o de 0.94 onde temos o valor de 5 para o número de árvores e 3 para o método *Repeated Stratified Fold.* O maior valor de acurácia obtido foi o de 0.980 onde temos o valor de 100 para o número de árvores e 5 para o método *Repeated Stratified Fold.* Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.95 onde temos o valor de 5 para o número de árvores e 3 para o método *Repeated Stratified Fold.* O maior valor de acurácia obtido foi o de 0.980 onde temos o valor de 50 para o número de árvores e 3 para o método *Repeated Stratified Fold.*

Tendo como base os dois resultados, observa-se que os modelos com poucas árvores, 5 e 10, foram os que obtiveram menores desempenhos. Em contrapartida, os modelos com 50 e 100 árvores foram os que obtiveram melhores desempenhos.

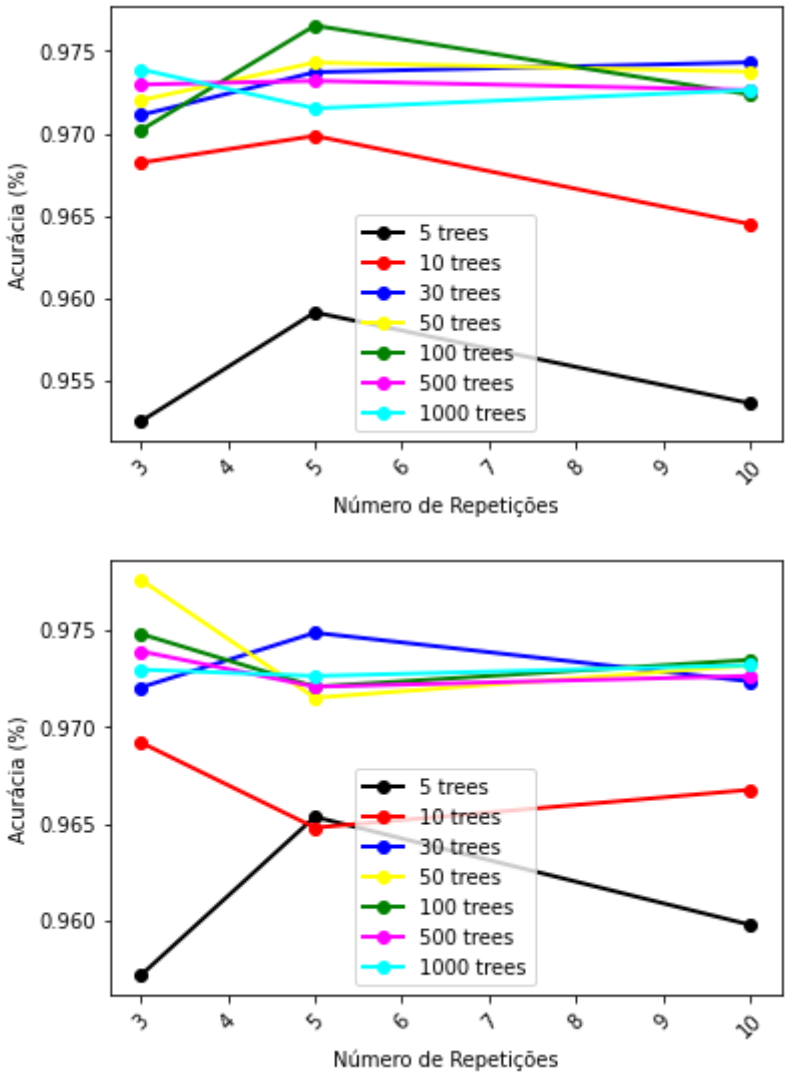


Figura 3. Resultados Random Forest Dataset 1.

Para o segundo dataset, desbalanceado onde substituiu-se pela médias os valores faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 4. O menor valor de acurácia obtido foi o de 0.9575 onde temos o valor de 5 para o número de árvores e 5 para o método *Repeated Stratified Fold.* O maior valor de acurácia obtido foi o de 0.98 onde temos o valor de 50 para o número de árvores e 10 para o método *Repeated Stratified Fold.* Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.9575 onde temos o valor de 5 para o número de árvores e 3 para o método *Repeated Stratified Fold.* O maior valor de acurácia obtido foi o de 0.9775 onde temos o valor de 50 para o número de árvores e 3 para o método *Repeated Stratified Fold.*

Tendo como base os dois resultados, observa-se que os modelos com poucas árvores, 5 e 10, foram os que obtiveram menores desempenhos. Em contrapartida, os modelos com 30, 50 e 100 árvores foram os que obtiveram melhores desempenhos.

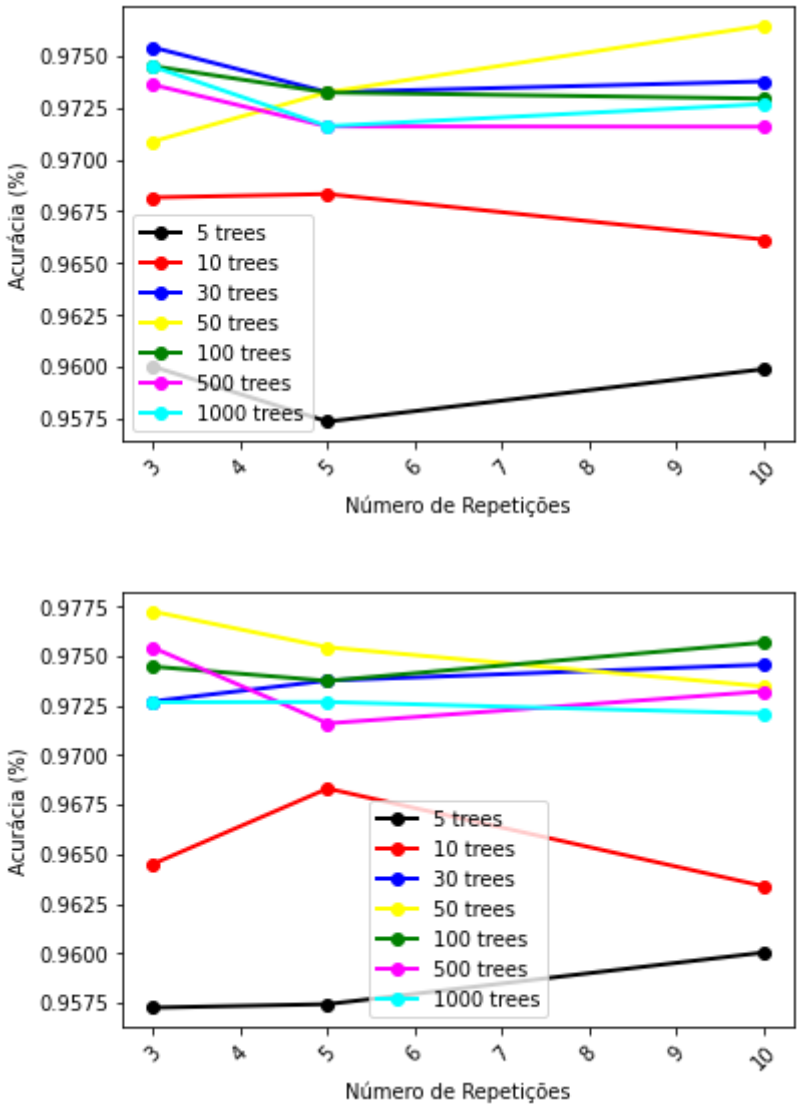


Figura 4. Resultados Random Forest Dataset 2.

Para o terceiro dataset, balanceado onde substituiu-se pela médias os valores faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 5. O menor valor de acurácia obtido foi o de 0.945 onde temos o valor de 5 para o número de árvores e 10 para o método *Repeated Stratified Fold.* O maior valor de acurácia obtido foi o de 0.975 onde temos o valor de 30 para o número de árvores e 10 para o método *Repeated Stratified Fold.* Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.965 onde temos o valor de 5 para o número de árvores e 5 para o método *Repeated Stratified Fold.* O maior valor de acurácia obtido foi o de 0.98 onde temos o valor de 50 para o número de árvores e 10 para o método *Repeated Stratified Fold.*

Tendo como base os dois resultados, observa-se que os modelos com poucas árvores, 5 e 10, foram os que obtiveram menores desempenhos. Em contrapartida, os modelos com 30, 50 e 100 árvores foram os que obtiveram melhores desempenhos.

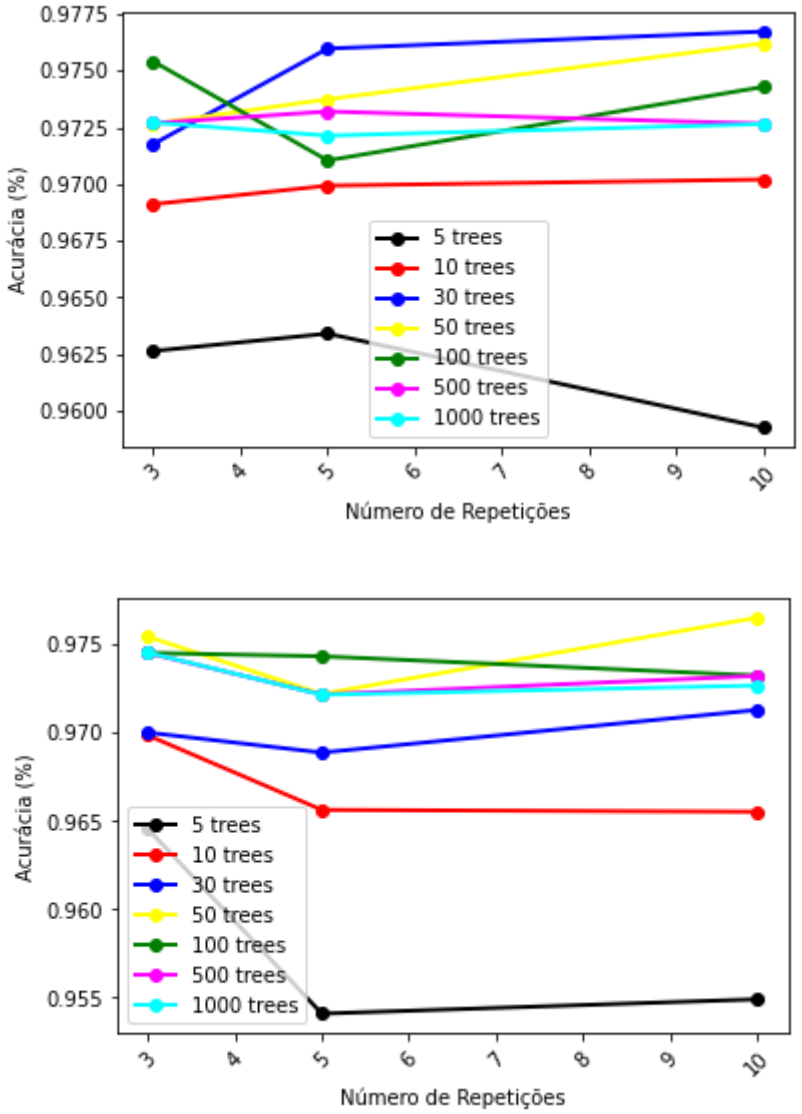


Figura 5. Resultados Random Forest Dataset 3.

De modo geral, obtivemos valores altos de acurácia para os três datasets. Comparando todos os resultados obtidos, tivemos que o menor valor de acurácia obtido foi o de 0.94 e o maior valor de acurácia obtido foi o de 0.980, logo encontra-se um baixo desvio padrão entre as médias encontradas.

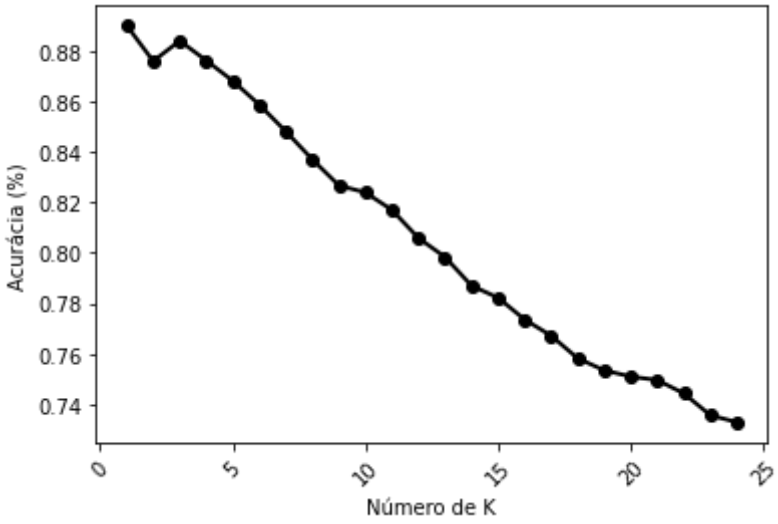
3.2 KNN

O KNN (*k-nearest neighbors algorithm*) é uma técnica de classificação *lazy* e, por esse motivo, difere da anteriormente utilizada. Para cada nova instância, o algoritmo executa todo o processo de classificação. Enquanto que no Random Forest, o modelo é criado em um primeiro momento e depois as novas instâncias são classificadas utilizando esse modelo. O KNN utiliza a ideia de vizinhança, ou seja, verifica os K vizinhos mais próximos baseando-se no cálculo de distância para classificar uma nova instância. A distância pode ser calculada utilizando algumas

abordagens diferentes, inclusive, a distância Euclidiana, calculada por: $dist(X1, X2) = \sqrt{\sum_{i=1}^{n} (x1_i - x2_i)^2}$.

Para a técnica de KNN, foram realizados alguns ajustes no parâmetro K para verificar de que forma ele interfere nos resultados gerados. Inicialmente, o valor atribuído foi K = 1 e testes foram realizados a partir deste.

Para o primeiro dataset, desbalanceado onde retirou-se tuplas com os dados faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 6. O menor valor de acurácia obtido foi o de 0.74 onde temos o valor de 25 para o número de K**.** O maior valor de acurácia obtido foi o de 0.890 onde temos o valor de K foi 1 (equivalente ao *nearest neighbor algorithm*). Observa-se que os valores de acurácia estão decaindo. Para esse teste contamos com 366 registros e a técnica de validação *Repeated Stratified Fold* com 5 partições e 10 repetições. Para uma segunda execução, os valores se mantiveram os mesmos, e decaindo quanto maior o número de K a partir de 3.



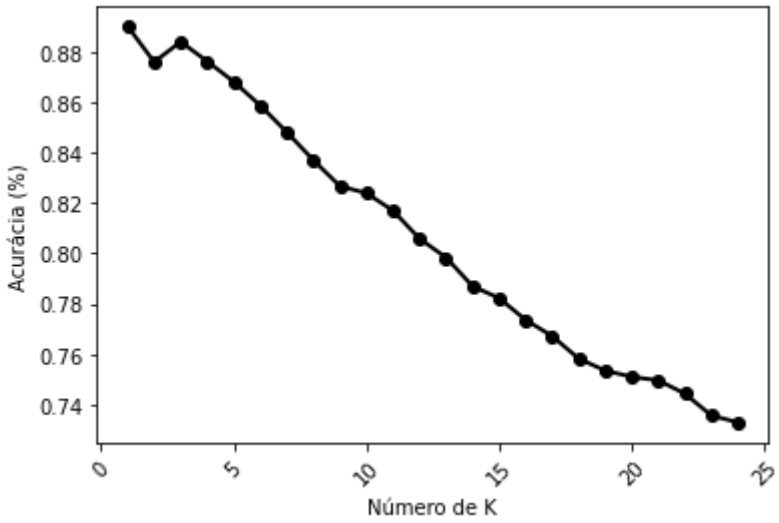
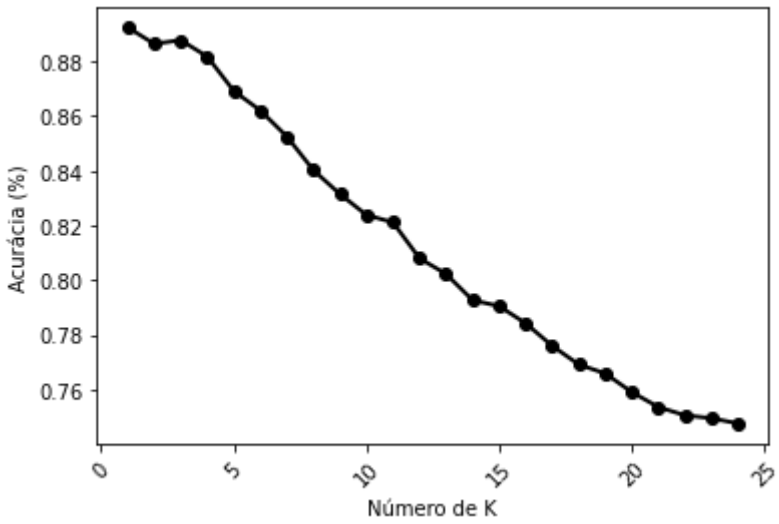


Figura 6. Resultados KNN Dataset 1.

Para o segundo dataset, desbalanceado onde substituiu-se pela médias os valores faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 7. O menor valor de acurácia obtido foi o de 0.75 onde temos o valor de 25 para o número de K*.* O maior valor de acurácia obtido foi o de 0.890 onde temos o valor de K foi 1*.* Observa-se que os valores de acurácia estão decaindo. Para esse teste contamos com 358 registros e a técnica de validação *Repeated Stratified Fold* com 5 partições e 10 repetições. Para uma segunda execução, os valores se mantiveram os mesmos, e decaindo quanto maior o número de K a partir de 3.



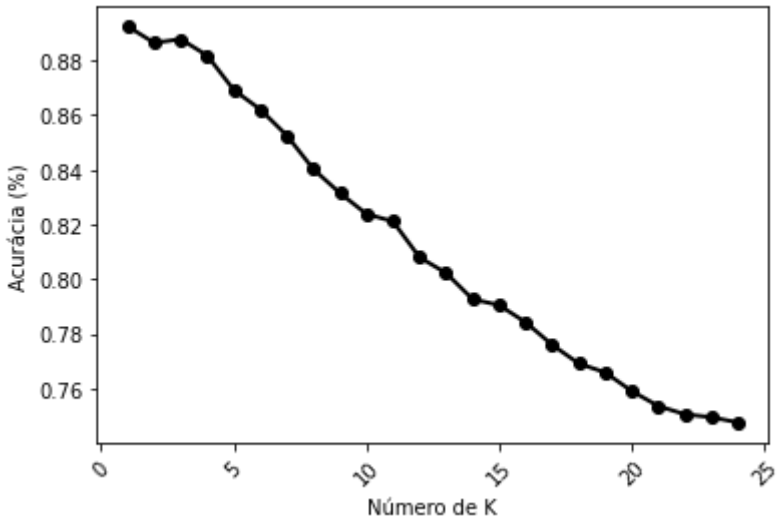
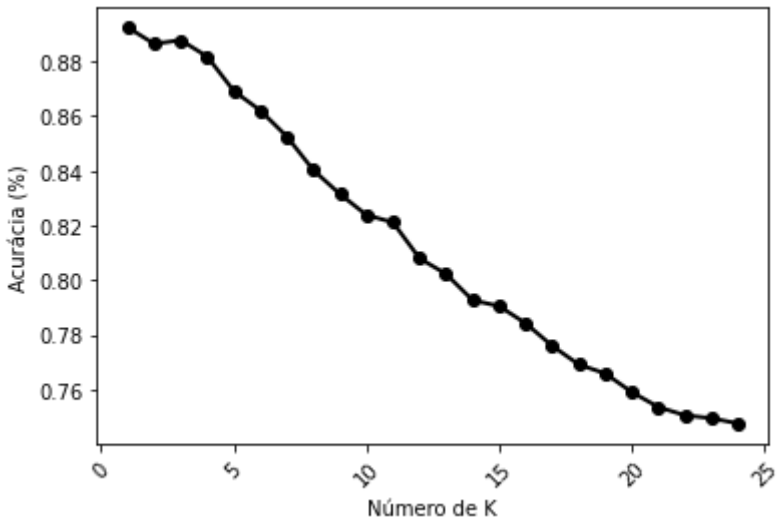


Figura 7. Resultados KNN Dataset 2.

Para o terceiro dataset, balanceado onde substituiu-se pela médias os valores faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 8. O menor valor de acurácia obtido foi o de 0.75 onde temos o valor de 25 para o número de K*.* O maior valor de acurácia obtido foi o de 0.890 onde temos o valor de K foi 1*.* Observa-se que os valores de acurácia estão decaindo. Para esse teste contamos com 665 registros e a técnica de validação *Repeated Stratified Fold* com 5 partições e 10 repetições. Para uma segunda execução, os valores se mantiveram os mesmos, e decaindo quanto maior o número de K a partir de 3.



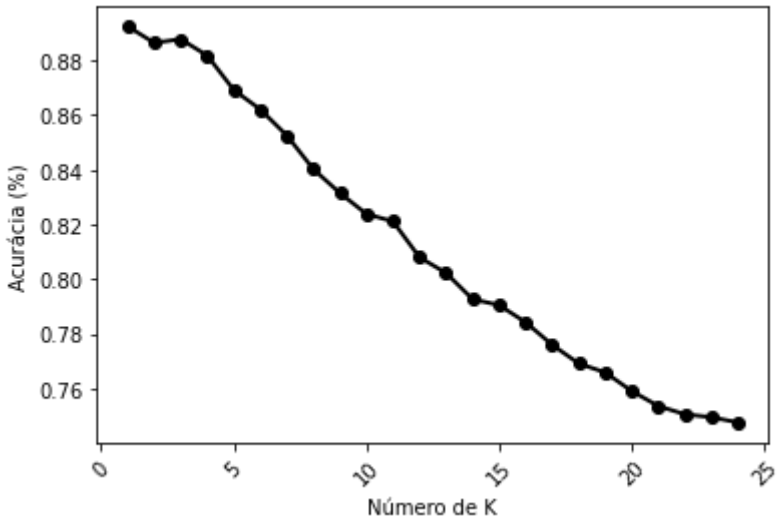


Figura 8. Resultados KNN Dataset 3.

Para os três exemplos de datasets que utilizamos, obtivemos resultados muito próximos de acurácia máxima. Comparando os dois modelos utilizados, Random Forest e o KNN, obtivemos melhores acurácias com o modelo Random Forest, que foram de 0.980.

4 Ensemble baseado em Bagging

A tomada de decisão de através de comitês, em que os membros apresentam suas opiniões baseados em seus conhecimentos e experiências são aplicadas diante de problemas complexos. Este mesmo princípio, vem sendo aplicado para problemas de aprendizado supervisionado.

Através dele, pode-se obter um resultado para um problema de classificação através de comitês de classificadores, denominadas usualmente como ensembles. Ensemble é um paradigma de aprendizado em que um grupo finito de propostas alternativas é utilizado para a solução de um dado problema. O uso da abordagem tem sido bastante explorado na última década, por se tratar de uma técnica simples e capaz de aumentar a capacidade de generalização de soluções baseadas em aprendizado de máquina.

O método Bagging é um algoritmo para construção de classificadores agregados. Este consiste em gerar subconjuntos de exemplos através de sorteio simples com reposição, sobre o conjunto de dados de treinamento original. Cada subconjunto amostrado é utilizado para a construção de um novo classificador.

A classificação final é realizada por um sistema de votação, em que usualmente se atribui para uma nova instância a classe com maior número de votos entre os classificadores. Uma vez que a classe prevista resulta da combinação das decisões individuais dos classificadores, demonstra-se que as previsões se tornam mais

confiáveis à medida que se têm mais votos. Dessa forma, o método bagging aposta na expansão da quantidade de classificadores a fim de se reduzir a variância do conjunto, especialmente na presença de dados ruidosos.

Como ponto inicial para resolução do problema, optou-se pela implementação da definição das partições dado os exemplos e em seguida a criação dos modelos baseados em bagging. As seções 4.1 e 4.2 descrevem as abordagens utilizadas.

4.1 Definição de partições de exemplos

Para a definição de partições de pontos ou observações, foi implementado para o presente projeto o algoritmo *K-Means*. Este é um método de agrupamento que tem como objetivo, a partição de um conjunto de n observações em k grupos em que cada observação pertence ao grupo cujo centróide é o p mais próximo. Como função de distância foi utilizada a distância euclidiana. A função que implementa este algoritmo chama-se *k_means.*

Adicionalmente, foi desenvolvido um algoritmo próprio com o objetivo de obter partições que contenham exemplos menos similares, ou seja, pontos ou observações diferentes/distantes uns dos outros. O algoritmo utiliza a ideia de sempre adicionar ao conjunto de observações aquela observação mais distante do conjunto até aquele momento, ou seja, a maior distância daquela observação. Para este método também utilizou-se a distância euclidiana. A função que implementa este algoritmo chama-se *get_ dissimilar_partitions,* e o pseudocódigo é apresentado a seguir.

Pseudocódigo, algoritmo de obtenção de partições dissimilares:
1. Obter os pontos iniciais p_1, p_2, ..., p_k.
2. Criar os grupos iniciais {p_1}, {p_2}, ..., {p_k}, cada um dos quais contém no início um ponto inicial. Criar o conjunto *all_points* dos pontos que por enquanto não estão em nenhum grupo.
3. Percorrer de forma circular os grupos (i = 1, ..., k), enquanto haja pontos em *all_points*:
   3.1. Para o grupo p_i, encontrar, dentre os pontos em *all_points*, um ponto *point_to_add* cuja distância ao ponto de p_i mais próximo, seja a maior.
   Observação: este será o "ponto mais distante" de p_i.
   3.2. Adicionar *point_to_add* a p_i.
   3.2. Remover *point_to_add* do conjunto all_points.
4. Retornar os grupos ou partições {p_1}, {p_2}, ..., {p_k}, cujos elementos são diferentes uns dos outros.

Para testar a correção dos algoritmos, *k_means* e *get_ dissimilar_partitions,* foi criada uma base pequena de teste *testData de* 29 instâncias e como os atributos X e Y, referentes às coordenadas no plano, e a classe ou grupo do ponto 1 ou 2,. Com esta base foi possível visualizar o funcionamento correto dos algoritmos, verificando

que conseguem classificar de forma adequada os pontos de acordo aos critérios de cada algoritmo.

A Figura 9 representa o resultado da classificação do algoritmo *k_means* para o dataset de teste *testData.* A Figura 10 representa o resultado da classificação do algoritmo *get_ dissimilar_partitions* para o dataset de teste *testData.* Definindo como os dados seriam particionados, iniciou-se a implementação dos algoritmos de bagging, descritos na seção seguinte 4.2.

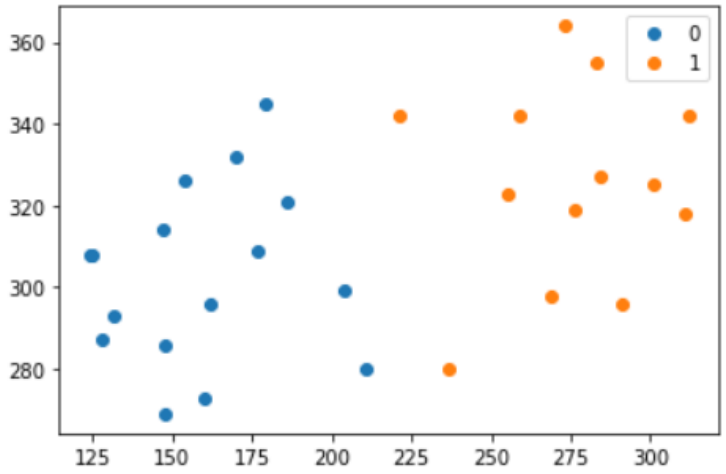


Figura 9. Classificação do k-means

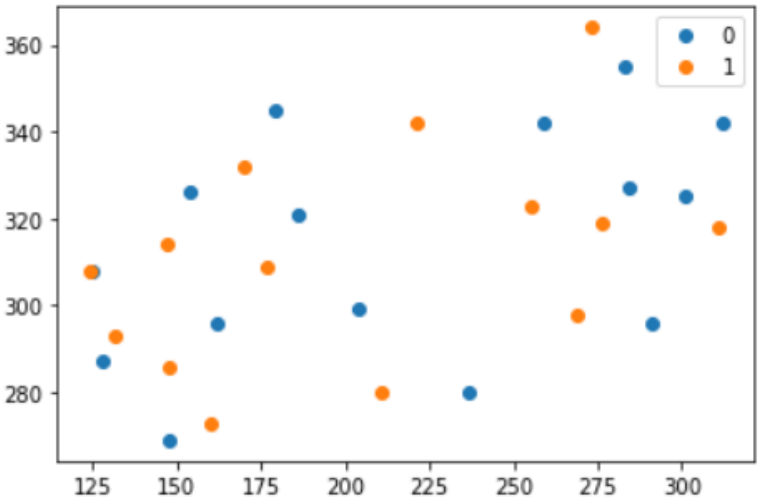


Figura 10. Clasificação do get_ dissimilar_partitions

4.2 Proposta do algoritmo baseado em bagging

Para a primeira proposta de construção do algoritmo baseado em bagging, utilizou-se árvores de decisão. Como parâmetros da função BaggingClassifier adicionamos como parâmetro *base_estimator=DecisionTreeClassifier().*

As Árvores de Decisão são um método de aprendizado supervisionado não paramétrico usado para classificação e regressão. O objetivo é criar um modelo que preveja o valor de uma variável de destino aprendendo regras de decisão simples inferidas a partir dos recursos de dados. Uma árvore pode ser vista como uma aproximação constante por partes.

Para a técnica de bagging utilizamos um conjunto de N árvores com intuito de obter a melhor acurácia final. Cada árvore deste conjunto executa a classificação de forma independente, através de um subconjunto de elementos similares (função K_means desenvolvida) ou dissimilares (função get_dissimilar_partitions desenvolvida) e, ao final, a média das acurácias é calculada.

Para as execuções, ajustou-se um parâmetro com o objetivo de verificar a forma que ele interfere no modelo gerado, tal parâmetro foi o número de árvores. Variou-se o número de árvores entre 5, 10, 30, 50, 100.

Além disso, adotou-se o método *Repeated Stratified Fold* como modelo de avaliação. Optou-se por esse método uma vez que a base de dados é desbalanceada e ele realiza validação cruzada de forma repetida, o que melhora a estimativa de desempenho médio. Foram utilizados os números mais comuns de repetições e folders que são: 3, e 5, respectivamente.

Para o primeiro dataset, desbalanceado onde retirou-se tuplas com os dados faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 11. Para este experimento, utilizou-se os dados similares obtidos através da função K Means desenvolvida. Optou-se por realizar K=5 partições dados que teríamos 5 modelos, variando o número de árvores: 5, 10, 30, 50, 100. O menor valor de acurácia obtido foi o de 0.889 onde temos o valor de 5 para o número de K**.** O maior valor de acurácia obtido foi o de 0.966 onde temos o valor de K foi 50. Observa-se que o valor de acurácia decaiu na execução seguinte quando o K tinha como valor 100 árvores. Para esse teste contamos com 366 registros e a técnica de validação *RepeatedStratifiedKFold* com 5 partições e 3 repetições.

Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.875 onde temos o valor de 30 para o número de K**.** O maior valor de acurácia obtido foi o de 0.976 onde temos o valor de K foi 100. Observa-se que o valor de acurácia decaiu nas execução onde K tinha como valor 10 e 30, mas logo voltou a crescer com os valores para K iguais a 50 e 100. Para esse teste contamos com 366 registros e a técnica de validação *RepeatedStratifiedKFold* com 5 partições e 3 repetições.

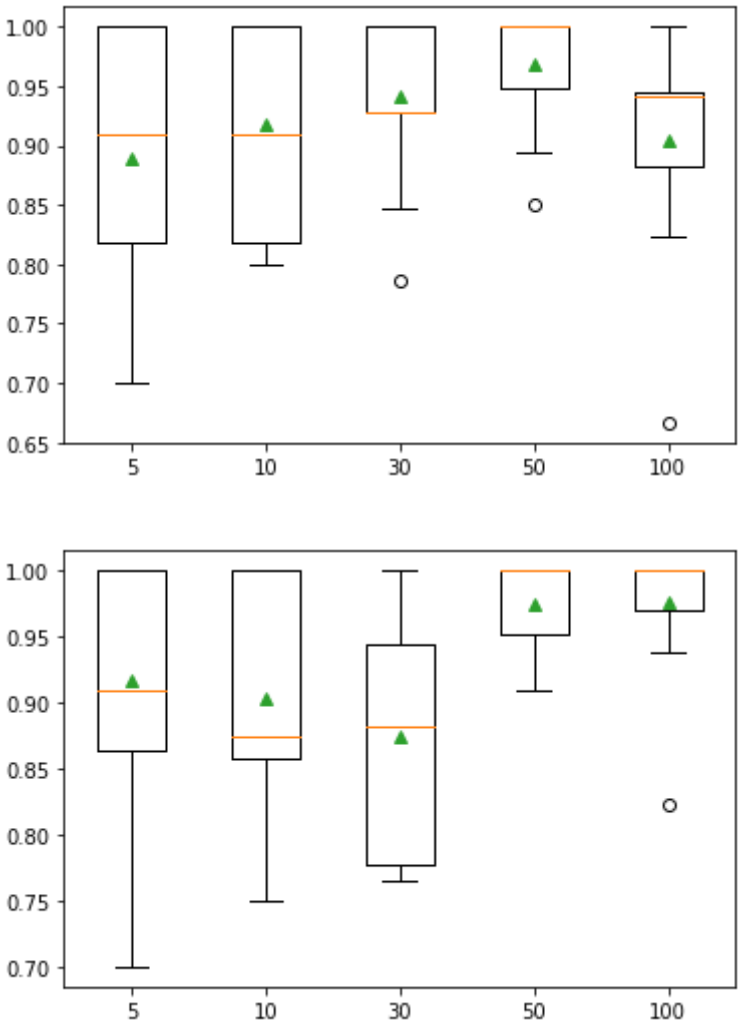


Figura 11. Resultados K Means Bagging.

Para o primeiro dataset, desbalanceado onde retirou-se tuplas com os dados faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 12. Para este experimento, utilizou-se os dados dissimilares obtidos através da função get_dissimilar_partitions desenvolvida. O menor valor de acurácia obtido foi o de 0.875 onde temos o valor de 5 para o número de K**.** O maior valor de acurácia obtido foi o de 0.920 onde temos o valor de K foi 100. Observa-se que o valor de acurácia não decaiu como na execução anterior. Para esse teste contamos com 366 registros e a técnica de validação *Repeated Stratified Fold* com 5 partições e 3 repetições.

Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.86 onde temos o valor de 5 para o número de K**.** O maior valor de acurácia obtido foi o de 0.954 onde temos o valor de K foi 100. Observa-se que o valor de acurácia cresceu com os valores para K maiores que 5. Para esse teste contamos com 366 registros e a técnica de validação *Repeated Stratified Fold* com 5 partições e 3 repetições.

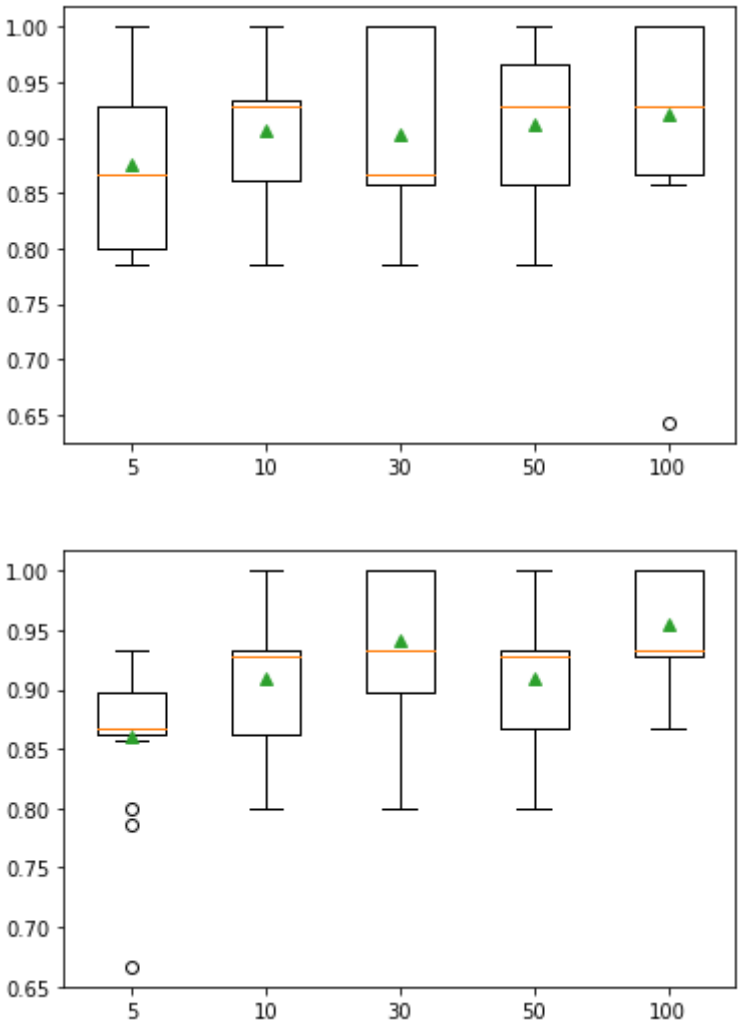


Figura 12. Resultados Dissimilar Bagging.

Dado que obtivemos valores de acurácia altos para os pontos dissimilares, realizamos uma segunda proposta de construção do algoritmo. Foi implementado um Bagging próprio criando uma classe chamada de *Bagging Model* que possui três métodos. O primeiro método é o método *__init__* que somente inicializa uma lista de árvores. O próximo é o método train_bagging_model que realiza o treinamento do modelo utilizando o *DecisionTreeClassifier()*. Tal método, recebe como parâmetro as informações sobre os dados, o número de árvores e o método que será utilizado para definição das partições são, podendo ser o *k_means,* que agrupa os elementos de conjunto de dados pelos mais semelhantes, ou *get_dissimilar_partitions,* que agrupa os elementos pelos menos semelhantes. Por último, o método *predict()* que, a partir de uma observação, retorna como resultado a predição do modelo baseado em votação, ou seja, retorna a resposta mais frequente entre as votações de todas as árvores. Para a validação do próprio algoritmo de Bagging foi necessário implementar o algoritmo de *cross-validations*.

Para o primeiro experimento, utilizando os dados similares e o dataset desbalanceado onde retirou-se tuplas com os dados faltantes, tem-se os resultados

obtidos por cada execução mostrados no Figura 13. Para este experimento, utilizou-se os dados similares obtidos através da função K-Means desenvolvida. O menor valor de acurácia obtido foi o de 0.2 onde temos o valor de 2 para o número de K**.** O maior valor de acurácia obtido foi o de 0.8 onde temos o valor de K foi 1. Observa-se que o valor de acurácia decaiu nas execuções seguintes. Para esse teste, contamos com 366 registros e a técnica de validação *Cross-Validations* com 10 partições.

Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.2 onde temos o valor de 6 para o número de K**.** O maior valor de acurácia obtido foi o de 0.85 onde temos o valor de K foi 10. Observa-se que o valor de acurácia teve muitas variações entre as 10 árvores. Para esse teste contamos com 366 registros e a técnica de validação *Cross-Validations* com 10 partições.

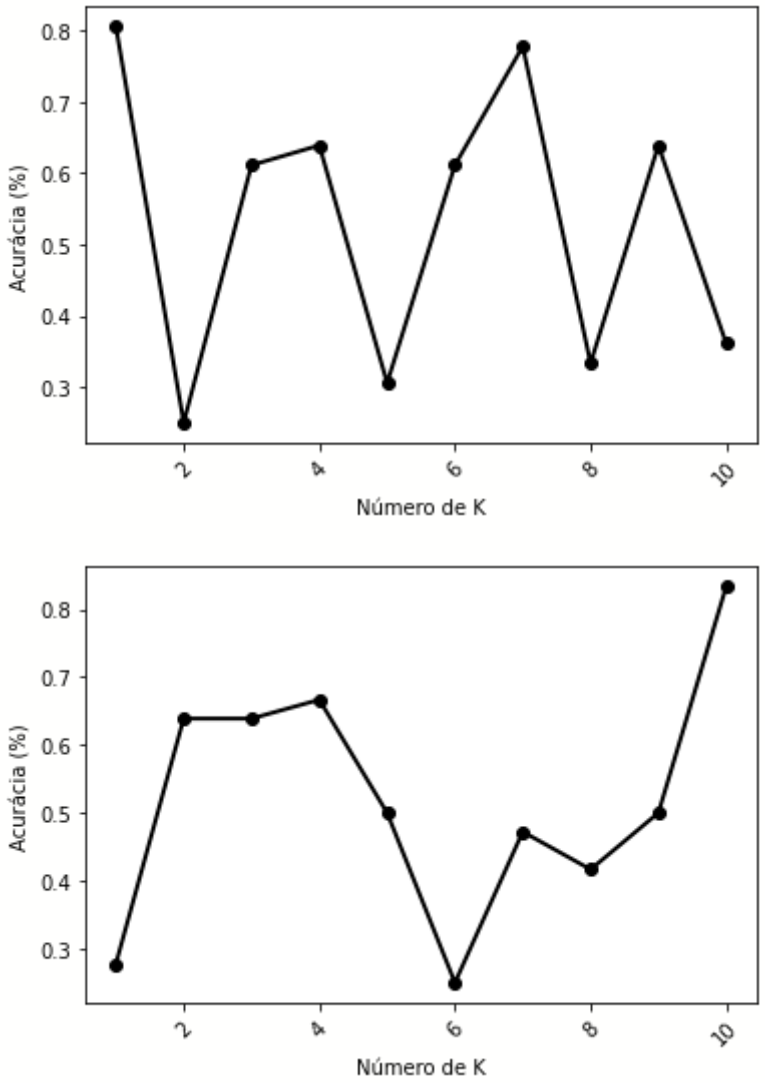


Figura 13. Resultados K Means Bagging Model.

Para o primeiro experimento, utilizando os dados dissimilares e com o dataset, desbalanceado onde retirou-se tuplas com os dados faltantes, tem-se os resultados obtidos por cada execução mostrados no Figura 14. Para este experimento, utilizou-se os dados dissimilares obtidos através da função get_dissimilar_partitions desenvolvida. O menor valor de acurácia obtido foi o de 0.1 onde temos o valor de 1 para o número de K**.** O maior valor de acurácia obtido foi o de 0.45 onde temos o valor de K foi 2. Observa-se que o valor de acurácia teve muitas variações entre as 10 árvores. Para esse teste contamos com 366 registros e a técnica de validação *Cross-Validations* com 10 partições.

Para uma segunda execução, o menor valor de acurácia obtido foi o de 0.22 onde temos o valor de 1 para o número de K**.** O maior valor de acurácia obtido foi o de 0.36 onde temos o valor de K foi 2. Observa-se que o valor de acurácia decaiu quando o valor de K foi maior que 2. Para esse teste contamos com 366 registros e a técnica de validação *Cross-Validations* com 10 partições.

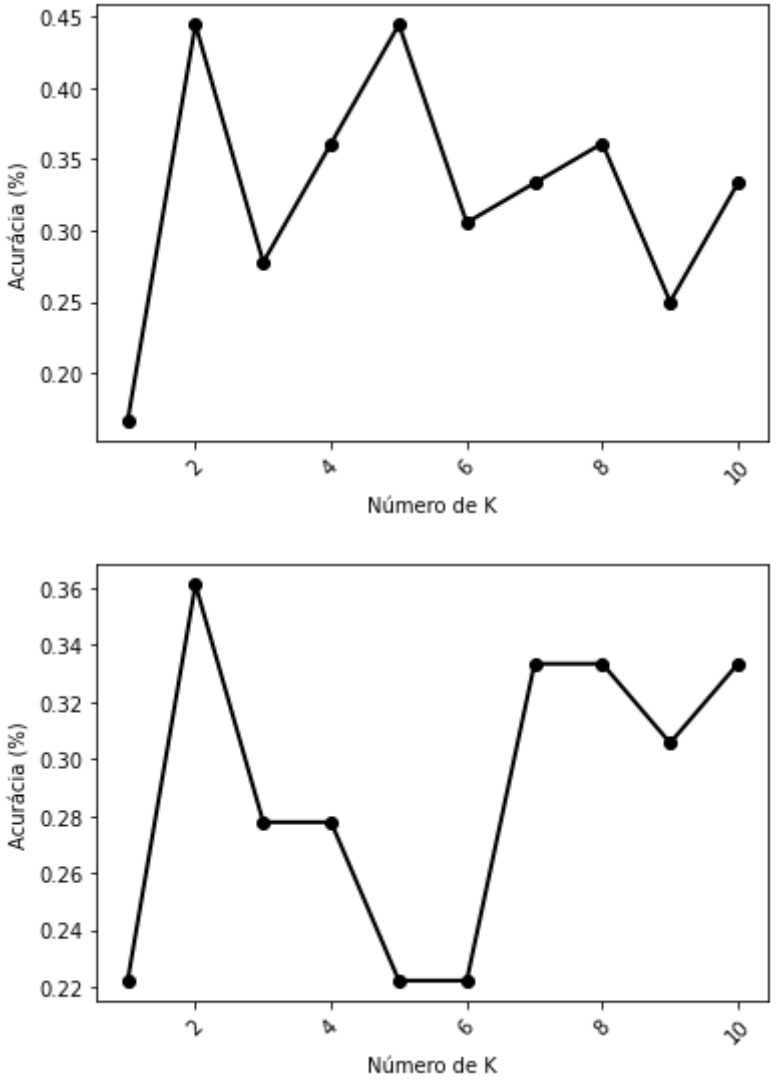


Figura 14. Resultados Dissimilar Bagging Model.

5 Conclusões

- Para a execução das tarefas propostas no presente trabalho, foi selecionado o dataset *Dermatology Database*, possui 366 instâncias e 34 atributos. Trinta e três dos atributos são de valores lineares e um nominal. As informações refletem os resultados do diagnóstico de doenças eritemato escamosas, um problema real na dermatologia. Tais doenças representam as classes deste grupo.
- Seguido da seleção e importação do dataset, foi realizado o pré-processamento dos dados, tratando os dados faltantes usando duas técnicas: primeiro, pela remoção dos itens faltantes, e segunda pela substituição dos valores faltantes pela média do atributo idade. Por último, para o tratamento do desbalanceamento entre a quantidade de classes foi aplicado o algoritmo de ADASYN para gerar amostras próximas às originais. Aplicou-se os algoritmos nos três datasets e não observamos diferenças significativas entre as acurácias obtidas, no caso do nosso dataset.
- Como Baselines foram escolhidos dois métodos: um método baseado em ensemble o *Random Forest,* método este que apresenta duas grandes vantagens, aumentam a acurácia dos resultados diminuindo a correlação de cada árvore e, além disso, fornecem estimativas dos erros de generalização das árvores de forma contínua. E um outro método não-paramétrico, o KNN, umas das vantagens deste método é que consegue um rápido treinamento e não perde/desperdiça informação, no entanto é mais lento em comparação com o *Random Forest.* Comparando os valores de acurácia entre ambos os métodos é possível observar uma maior precisão do *Random Forest* em relação ao *KNN*.
- Foram implementados dois algoritmos que atendessem aos critérios para a definição de partições de exemplos. Estes são: o K_means para obter partições que contenham exemplos similares, e o get_dissimilar_partitions para obter partições que contenham exemplos menos similares. Consequentemente foi testada a corretude de ambos os algoritmos com uma nova base de test criada. O dataset conta apenas com 29 instâncias e três atributos, sendo dois as coordenadas em um plano, e o outro a classe correspondente às coordenadas. Com isso é possível visualizar através de um gráfico o funcionamento correto dos algoritmos, o teste foi realizado , uma vez que, é difícil plotar a visualização de data sets maiores e com muitos atributos.
- Para a construção do algoritmo baseado em bagging foram feitas duas propostas: A primeira, utilizou-se o *BaggingClassifier* da biblioteca Sklearn e na segunda foi criado um bagging próprio chamado *BaggingModel.* Em ambas propostas foram utilizadas árvores de decisão e a função DecisionTreeClassifier(). Assim como as funções K_means, utilizada para

particionar o dataset em subconjuntos de elementos similares e get_dissimilar_partitions para particionar o dataset em subconjuntos de elementos dissimilares/distintos. Comparando ambas propostas o *BaggingClassifier* conseguiu um melhor desempenho, analisando a acurácia, para os dados similares. Em contrapartida, para dados dissimilares os melhores resultados de acurácia obtidos foram pelo modelo *BaggingModel.*

- Comparando todos os algoritmos, *Random Forest, KNN, BaggingClassifier (usando* K_means), *BaggingClassifier (usando* get_dissimilar_partitions), *BaggingModel (usando* K_means) e *BaggingModel (usando* get_dissimilar_partitions). O algoritmo *Random Forest* foi aquele que mostrou o melhor desempenho e resultados. O pior desempenho foi observado no algoritmo *BaggingModel*, usando partições com elementos menos similares, implementado no método *get_dissimilar_partitions*. Uma hipótese é que isso acontece por uma dimininuição da densidade dos pontos em cada classe, efeito que pode ser observado na Figura 10. Nesse caso, as classes com os menores números de observações podem sofrer e inclusive ficar sem observações de uma determinada classe. O uso de datasets com milhões ou dezenas de milhões de observações poderiam diminuir este efeito e mostrar se este algoritmo realmente pode se comportar de maneira eficiente. Por outra parte, entende-se que os algoritmos da biblioteca do Sklearn são melhores e robustos que o nosso algoritmo. O algoritmo proposto requer melhorias e otimizações.

6 Atribuição de tarefas

Dentre as responsabilidades de cada aluna, temos: a aluna Camila Paiva ficou responsável pela importação da base, pré-processamento, retirada de dados faltantes, substituição dos dados faltantes, plotagem dos gráficos, criação do método baseado em ensemble e os procedimentos adequados para validação cruzada e cálculos de funções de avaliação, construção do ensemble baseado em bagging (o procedimento que utiliza o Bagging Classifier) e procedimentos adequados para validação cruzada e cálculos de funções de avaliação, e a escrita do relatório; e, a aluna Isela Mendonza responsável pela criação do método não-paramétrico e os procedimentos adequados para validação cruzada e cálculos de funções de avaliação, construção dos métodos de similaridade e dissimilaridade (função K Means e função get dissimilar partitions), construção do ensemble baseado em bagging (o procedimento próprio que utiliza o Bagging Model) e procedimentos adequados para validação cruzada e cálculos de funções de avaliação, e a escrita do relatório.

Referências

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Bramer, M. (2016). Principles of data mining. 3rd edition. 2016 ed. New York, NY: Springer London.

Russell, S. J., Norvig, P. and Davis, E. (2010). Artificial intelligence: a modern approach. 3rd ed ed. Upper Saddle River: Prentice Hall.

Simon, P. (2015). Too Big to Ignore: The Business Case for Big Data. John Wiley & Sons.

Winston, P. H. (1992). Artificial intelligence. 3rd ed ed. Reading, Mass: Addison-Wesley Pub. Co.