UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CIÊNCIA DA COMPUTAÇÃO

IMPLEMENTAÇÃO DE ÁRVORES DE DECISÃO PARA PROPRIEDADES TRIDIMENSIONAIS EM LINGUAGEM PYTHON

TRABALHO DE GRADUAÇÃO

Raphael Giordano do Nascimento e Silva

Santa Maria, RS, Brasil

2015

IMPLEMENTAÇÃO DE ÁRVORES DE DECISÃO PARA PROPRIEDADES TRIDIMENSIONAIS EM LINGUAGEM PYTHON

Raphael Giordano do Nascimento e Silva

Trabalho de Graduação apresentado ao curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientadora: Profª. Dra. Ana T. Winck

Trabalho de Graduação Nº 394
Santa Maria, RS, Brasil

2015

do Nascimento e Silva, Raphael Giordano

Implementação de Árvores de Decisão para Propriedades Tridimensionais em Linguagem Python / por Raphael Giordano do Nascimento e Silva. – 2015.

43 f.: il.; 30 cm.

Orientadora: Ana T. Winck

Monografia (Graduação) - Universidade Federal de Santa Maria, Centro de Tecnologia, curso de Ciência da Computação, RS, 2015.

1. Mineração de Dados. 2. Bioinformática. 3. Desenho Racional de Fármacos. 4. Árvores de Decisão. 5. Árvores de Regressão. I. Winck, Ana T.. II. Título.

E-mail: raphaelgns@gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

IMPLEMENTAÇÃO DE ÁRVORES DE DECISÃO PARA PROPRIEDADES TRIDIMENSIONAIS EM LINGUAGEM PYTHON

elaborado por
Raphael Giordano do Nascimento e Silva

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Ana T. Winck, Dr.
(Presidente/Orientadora)

Giliane Bernardi, Dra. (UFSM)

Giovani R. Librelotto, Dr. (UFSM)

Santa Maria, 13 de Julho de 2015.

Ao meu pai, João Onilso Alves da Silva (in memorian).

AGRADECIMENTOS

Primeiramente, gostaria de agradecer àqueles que não estão mais aqui: meu pai, João Onilso Alves da Silva; meu avô paterno, Benício Alves da Silva; e meu avô materno, Antônio Laurindo Filho. Agradeço também à minha mãe, pelo apoio incondicional ao longo de todo processo, desde a mudança para Santa Maria, ao apoio ao longo do curso. Um agradecimento especial também às minhas avós Socorro e Venina, por terem me apoiado ao longo desses anos. À minha namorada, Luana, por toda força, carinho e amor. Agradeço também aos demais familiares que de alguma forma contribuíram nesse processo. À família da minha namorada, que me apoiou bastante ao longo desses anos.

À minha orientadora, profª Ana Trindade Winck, por ter me aceitado no grupo de pesquisa e por ter acreditado no meu trabalho. Por me proporcionar aprender mais sobre mineração de dados e bioinformática.

Por fim, agradeço aos grupos de pesquisa dos quais faço parte: SIn e GPKosmos - incluindo minhas orientadoras nos mesmos e demais membros. Aos demais que eu não mencionei: obrigado.

“Ideias e somente ideias podem iluminar a escuridão.”
— LUDWIG VON MISES

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

IMPLEMENTAÇÃO DE ÁRVORES DE DECISÃO PARA PROPRIEDADES TRIDIMENSIONAIS EM LINGUAGEM PYTHON

AUTOR: RAPHAEL GIORDANO DO NASCIMENTO E SILVA
ORIENTADORA: ANA T. WINCK
Local da Defesa e Data: Santa Maria, 13 de Julho de 2015.

A bioinformática faz uso de técnicas computacionais para resultados biológicos. O desenho racional de fármacos (RDD - *Rational Drug Design*) é uma importante subdivisão da bioinformática que trata fundamentalmente da interação das macromoléculas, chamadas de receptores, e moléculas menores chamadas de ligantes. O objetivo consiste em investigar o melhor encaixe entre essas duas moléculas para realizar ou inibir funções específicas. Esse encaixe pode ser medido pela energia livre de ligação (FEB - *Free Energy Binding*). A maioria dos algoritmos de docagem molecular considera o receptor como uma estrutura rígida. Porém, um método mais realista, e de melhor resultado, deve considerar não só o ligante como uma estrutura flexível, mas também o receptor. Isso é possível por meio de uma técnica chamada de dinâmica molecular (DM). Após os experimentos *in silico* de RDD, os melhores ligantes de um determinado receptor são testados *in vitro* e um novo fármaco pode surgir. O algoritmo implementado neste trabalho induz uma árvore de decisão para regressão, uma técnica de classificação onde o atributo classe contínuo, que considera as propriedades tridimensionais para se ter um melhor resultado em relação à técnicas de indução convencionais. O algoritmo utiliza as coordenadas para dividir um nodo em duas partes, onde o átomo é avaliado em termos de sua posição em um bloco que melhor represente sua posição no espaço. Um especialista de domínio pode então selecionar conformações promissoras do receptor a partir do modelo induzido. Este trabalho trata da implementação do algoritmo 3D-Tri na linguagem Python. Essa linguagem foi escolhida devido ao seu amplo uso no contexto de mineração de dados, seu grande número de bibliotecas disponíveis e a facilidade de escrita e leitura nessa linguagem. Testes foram realizados com dados do ligante NUNL2, conhecido inibidor da bomba de efluxo AcrB (*Acriflavine resistance protein B*). Esses testes resultaram em um modelo facilmente interpretado por um especialista de domínio, uma característica do algoritmo 3D-Tri. Esta implementação visa acrescentar melhorias ao algoritmo e ampliar o uso do mesmo, para consequentemente contribuir no processo de RDD.

Palavras-chave: Mineração de Dados. Bioinformática. Desenho Racional de Fármacos. Árvores de Decisão. Árvores de Regressão.

ABSTRACT

Undergraduate Final Work
Computer Science
Federal University of Santa Maria

A PYTHON-BASED DECISION TREE IMPLEMENTATION FOR THREEDIMENSIONAL PROPERTIES

AUTHOR: RAPHAEL GIORDANO DO NASCIMENTO E SILVA
ADVISOR: ANA T. WINCK
Defense Place and Date: Santa Maria, July 13st, 2015.

Bioinformatics is a field of research that makes use of computational techniques for bioilogical results. Rational Drug Design is an importante field of research focusing on the interaction between macromolecules, called as receptors, and small molecules, called as ligands. The objective is to investigate the best fit between these molecules to perform or inhibit specific functions, which can be measured by the estimated Free Energy of Binding (FEB). Most studies consider the receptor as a rigid structure, however a more realistic method must consider not only the ligand as a flexible structure, but also the receptor. This flexibility can be simulated by means of a technique called Molecular Dynamics (MD) simulation. After an *in silico* RDD experiments, the most promissing ligands for a particular receptor are tested *in vitro* and a new drug may be created. The algorithm implemented in this work induces a regression-tree considering the three-dimensional properties of the receptor's atoms the leads to a good estimated FEB value. This algorith makes use of the coordinates to split a node into two parts, where the atom is evaluated in terms of its pose in a block - which represents the best position in the space. A domain expert may then select promising conformations of the receptor from the induced model. This work deals with the implementation of the 3D-Tri algorithm in Python. This language was chosen because of its extensive use in data mining context, its large number of available libraries and the facility of reading and writing in that language. Tests were performed with data of NUNL2 ligand, known efflux pump inhibitor AcrB (Acriflavine resistance protein B). These tests resulted in a model easily interpreted by a domain expert, a characteristic of the 3D-Tri algorithm. This implementation aims to add improvements to the algorithm and extend the use of it, to thus contribute to the RDD process.

Keywords: Data Mining. Bioinformatics. Rational Drug Design. Decision Trees. Regression Trees.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

3D	Tridimensional
3D-Tri	*Three-Dimensional Regression Tree Induction Algorithm*
AcrB	*Acriflavine resistance protein B*
API	*Application Programming Interface*
DM	Dinâmica Molecular
FEB	*Free Energy Binding*
MAE	*Mean Absolute Error*
PDB	*Protein Data Bank*
RDD	*Rational Drug*
RDP	Redução de Desvio Padrão
RMSE	*Root Mean Squared Error*
SVM	*Support Vector Machine*

SUMÁRIO

1 INTRODUÇÃO

Mineração de dados é uma parte integral da descoberta de conhecimentos em bancos de dados (KDD - *Knowledge Discovery in Databases*), e consiste no processo de descoberta de informações úteis em grandes volumes de dados. Diversas técnicas de mineração de dados são utilizadas com o intuito de descobrir padrões úteis, que sem o uso das mesmas seriam ignorados (TAN et al., 2006). Esses padrões são identificados por meio de algoritmos de aprendizagem de máquina.

O conhecimento descoberto por meio da mineração de dados pode ser classificado em rótulos de acordo com seus atributos. A classificação, segundo Tan et. al (2006), é a tarefa de aprender um modelo de predição que mapeie cada conjunto de atributos para cada um dos rótulos de classes pré-determinados. Uma das técnicas de classificação muito utilizada é a indução de árvores de decisão. Representações na forma de árvores de decisão podem ser melhor compreendidas por usuários finais que representações do tipo caixa-preta, como redes neurais e SVM (*Support Vector Machine*) (FREITAS; WIESER; APWEILER, 2010). Diversas são as áreas de aplicação que se beneficiam com a mineração de dados, em especial com modelos de classificação. Dentre elas, pode-se citar a bioinformática.

A bioinformática pode ser vista como a ciência que faz uso de técnicas computacionais para obter resultados biológicos mais acurados. Essa ciência possui várias subdivisões, onde uma delas é o desenho racional de fármacos (RDD - *Rational Drug Design*) (KUNTZ, 1992). O desenho racional de fármacos trata fundamentalmente da interação das macromoléculas - chamadas de receptores - e moléculas menores - chamadas de ligantes (LYBRAND, 1995). O objetivo dos experimentos em RDD consiste em investigar o melhor encaixe e conformação entre um ligante e uma cavidade de um receptor. Esse encaixe pode ser medido de acordo com a energia livre de ligação (FEB - *Free Energy Binding*).

O processo de docagem molecular - do ligante no receptor - não é simples. O principal fator que pode influenciar nos resultados é a flexibilidade do receptor. A maioria dos algoritmos de docagem molecular considera somente a flexibilidade do ligante, considerando, portanto o receptor como uma estrutura rígida. Os experimentos de docagem molecular que consideram o receptor flexível, assim como os ligantes, utilizam uma técnica chamada de dinâmica molecular (DM) (LIN et al., 2002). Um dos maiores problemas da utilização de DM é o tempo necessário para a execução dos experimentos e a quantidade de dados gerados.

O algoritmo 3D-Tri (*Three-Dimensional Regression Tree Induction Algorithm*) proposto por Winck (2012) é um algoritmo que trata como propriedades tridimensionais os dados provenientes de resultados de simulação de DM. No algoritmo, cada instância é uma conformação diferente do receptor, tendo como atributos preditivos as coordenadas espaciais dos átomos deste receptor, e o FEB como seu atributo alvo. Esse algoritmo se destaca na forma de indução desses atributos, utilizando uma abordagem de definição de intervalo ideal para cada coordenada dos átomos envolvidos (WINCK, 2012).

Este trabalho tem como objetivo implementar parte do algoritmo 3D-Tri proposto por Winck (2012) na linguagem Python. A escolha da linguagem Python se deu pela a ampla quantidade de bibliotecas desenvolvidas para mineração de dados e aprendizagem de máquina para a mesma. Um exemplo dessas bibliotecas é a Scikit-Learn, que é amplamente utilizada e bem documentada.

O capítulo 2 apresenta uma revisão bibliográfica acerca dos temas abordados neste trabalho, como RDD e árvores de decisão para classificação. No capítulo seguinte, é apresentado a definição do algoritmo 3D-Tri, proposto por Winck (2012), que será implementado neste trabalho. O capítulo 4 apresenta a metodologia aplicada neste trabalho, fornecendo informações a respeito da implementação, dados e parâmetros utilizados, e avaliação do modelo. O capítulo 5 apresenta os resultados obtidos neste trabalho. O capítulo 6 faz considerações a respeito dos resultados obtidos neste trabalho e sua contribuição.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma revisão dos temas abordados neste trabalho. Primeiramente, os conceitos de RDD são abordados, descrevendo a docagem molecular e a dinâmica molecular. Depois, os conceitos e estratégias empregadas na indução de árvores de decisão para classificação. Em seguida, as árvores de decisão para regressão são analisadas, enfatizando particularidades em relação às árvores de decisão para classificação.

2.1 Desenho Racional de Fármacos

A bioinformática foi inicialmente definida como uma área interdisciplinar envolvendo biologia, ciência da computação, matemática e estatística para analisar dados biológicos. Porém, com o advento da era genômica, a bioinformática passou a ser definida em termos de moléculas e a aplicação da computação para entender e organizar as informações associadas aos dados biológicos (LUSCOMBE; GREENBAUM; GERSTEIN, 2001) (MOUNT, 2004).

Uma das características da bioinformática - e também um de seus desafios - é a manipulação e análise em grandes volumes de dados (LESK, 2002). Esses grandes volumes de dados também tendem a crescer a uma taxa bastante elevada. Lesk (2002) aponta algumas áreas de atuação em bioinformática como: genômica, proteômica, alinhamento de árvores filogenéticas, biologia de sistemas e descoberta de fármacos.

Este trabalho tem como foco o contexto de Desenho Racional de Fármacos (RDD - *Rational Drug Design*) (KUNTZ, 1992). O objetivo fundamental do RDD é explorar a interação entre macromoléculas chamadas de receptores e moléculas menores chamadas de ligantes com o objetivo de realizar ou inibir funções específicas (LYBRAND, 1995) (BALAKIN, 2009). No contexto de RDD, um dos maiores desafios é a necessidade de lidar com grande volume de dados. Além da catalogação dos ligantes, há também os dados gerados por meio das simulações de dinâmica molecular (DM) - conformações do receptor - e o resultado dos experimentos de docagem molecular. Para executar tais experimentos é necessário um receptor, um ligante e um software para executar as simulações. Um software de docagem bastante utilizado é o AutoDock (MORRIS et al., 1998).

A indústria farmacêutica tem a necessidade constante de aumentar a taxa com que novos medicamentos são inseridos no mercado (LYBRAND, 1995). O alto custo e o tempo necessário

para que um novo fármaco seja disponibilizado são fatores importantes a se considerar. Portanto, esforços são aplicados com o intuito de reduzir os custos e o tempo, além de aumentar a qualidade dos compostos candidatos a fármacos.

O RDD é um ciclo que consiste em quatro etapas:

1. O primeiro passo consiste em isolar um alvo específico, chamado de receptor. Então, a partir de análise computacionais sobre sua estrutura tridimensional (3D), que por sua vez é armazenada em um banco de dados estrutural, como o PDB (*Protein Data Bank*) (BERMAN et al., 2000), é possível identificar regiões de ligação como, por exemplo, regiões onde uma pequena molécula - ligante - pode se ligar a esse receptor;

2. Baseado na provável região de ligação identificada no primeiro passo, é selecionado um conjunto de ligantes candidatos a se ligarem nesse receptor. As diferentes conformações que um ligante pode assumir dentro do sítio ativo de uma proteína em particular, podem ser simuladas por um software de docagem molecular, como o Autodock (MORRIS et al., 1998);

3. Os ligantes que, teoricamente, obtiverem os melhores resultados em simulações, são experimentalmente sintetizados e testados;

4. De acordo com os resultados experimentais, um novo medicamento pode ser gerado, ou o processo volta ao passo 1.

Na docagem molecular se investiga e avalia o melhor encaixe do ligante na estrutura alvo. Um ligante deve interagir com o receptor para exercer uma função fisiológica vinculada à ligação dessa com outras moléculas. Essas ligações determinam se as funções do receptor serão estimuladas ou inibidas (LYBRAND, 1995). Essas ligações ocorrem em locais específicos, conhecidos por sítios ativos de ligação.

Uma das maneiras de avaliar um resultado de docagem molecular é por meio do valor estimado de energia livre de ligação (FEB - *Free Energy of Binding*). Em outras palavras, o quanto dessa energia é despendida. Portanto, quanto mais negativa, mais forte é a ligação.

A maioria dos algoritmos de docagem molecular considera apenas a flexibilidade do ligante, considerando o receptor rígido. Entretanto, uma simulação mais realista considera a flexibilidade do receptor, e não somente a do ligante. A solução é a utilização de simulação de dinâmica molecular (DM) (LIN et al., 2002). Simulação por DM é uma das técnicas computacionais mais versáteis e amplamente utilizadas para se estudar macromoléculas biológicas

(GUNSTEREN; BERENDSEN, 1990). A partir de simulações pela DM é possível estudar o efeito explícito de ligantes na estrutura e estabilidade das proteínas, os diferentes parâmetros termodinâmicos envolvidos, incluindo energias de interação e entropias.

Simulações de DM foram utilizadas nos trabalhos desenvolvidos por Winck (2012) e em Winck et al. (2010). Dois problemas típicos da bioinformática são gerados nesses casos: tempo necessário para executar os experimentos e o grande volume dos dados gerados. O algoritmo 3D-Tri, proposto por Winck (2012), trabalha com dados de DM.

2.2 Indução de Árvores de Decisão

A mineração de dados é o processo de descoberta automática de informações úteis em grandes depósitos de dados (TAN et al., 2006). Técnicas de mineração de dados agem sobre grandes bancos de dados com a finalidade de descobrir padrões úteis que, de outra forma, permaneceriam ignorados. Também é possível por meio da mineração de dados que haja uma previsão de um resultado por meio de uma observação futura - tarefa de previsão.

A indução de árvores de decisão é um dos métodos mais utilizados para inferência indutiva. Essa indução é feita a partir de um conjunto de dados rotulados - contendo valores definidos em seu atributo alvo, também conhecido por classe. As árvores de decisão, como toda árvore, apresentam uma estrutura hierárquica. Uma vantagem de uma árvore de decisão é a representação do conhecimento descoberto na forma de grafo, onde a importância dos atributos utilizados na predição é observada através de sua estrutura hierárquica.

De acordo com Alpaydim (2010), uma árvore de decisão é um modelo hierárquico de aprendizagem supervisionada, onde regiões locais são mais identificadas em sequências recursivas de divisões do conjunto de dados. Uma árvore é composta por nodos de decisão internos e por nodos terminais - chamados de folha. Cada nodo *m* implementa uma função de teste $f_m(x)$ com resultados discretos que servem para rotular arestas. A partir de um conjunto de dados de entrada, um teste é aplicado para cada nodo e uma das arestas é percorrida de acordo com o resultado de rótulo. Esse processo inicia no nodo mais ao topo - raiz - e é repetido recursivamente até atingir um nodo folha. Esse nodo é o atributo alvo e descreve a saída.

Uma árvore de decisão é uma estrutura de dados hierárquica implementada a partir de uma estratégia de dividir para conquistar. A maioria dos algoritmos de árvores de decisão faz uso de uma estratégia gulosa. A partição de um nodo é feita de acordo com um parâmetro que identifica o ótimo local para este nodo. O algoritmo de classificação de Hunt é um algoritmo

básico de indução de árvore e serviu como base para construção de outros algoritmos mais aprimorados, como o ID3 (QUINLAN, 1986), o C4.5 (QUINLAN, 1993) e o CART (BREIMAN et al., 1984). A estratégia do algoritmo de Hunt consiste em:

1. Escolha um atributo;
2. Estenda a árvore adicionando um ramo para cada valor do atributo;
3. De acordo com o atributo escolhido, passe os exemplos para as folhas;
4. Para cada folha:
   (a) Se todos os exemplos pertencerem ao mesmo atributo alvo, associe este atributo à folha;
   (b) Caso contrário, repita os passos a partir do passo 1.

Tan et. al (2006) indica duas questões importantes para uma indução de modelos bem acurados:

- Qual a melhor maneira de particionar os atributos? Cada passo recursivo do processo de crescimento da árvore deve selecionar um atributo. Esse atributo é selecionado de acordo com um teste de condição. A partir disso, o conjunto de dados é dividido em subconjuntos. Para isso, o algoritmo deve implementar um método que avalie a qualidade de cada condição de teste, e assim escolha o atributo a ser utilizado no momento.
- Como o procedimento de divisão deve parar? Uma condição de parada deve ser estabelecida para terminar o processo de crescimento da árvore. Uma estratégia possível é continuar expandindo um nodo até que todos os registros pertençam à mesma classe ou tenham valores de atributos iguais. Entretanto, outros critérios podem ser impostos para que o procedimento de crescimento termine antes, visando uma melhor vantagem em relação ao modelo.

2.2.1 Indução de Árvores de Decisão Para Classificação

Classificação é o processo de encontrar um modelo ou função que descreva e diferencie classes de objetos de categoria desconhecida (HAN; KAMBER, 2011). Tan et. al (2006) define classificação como sendo a tarefa de aprender uma função alvo f - também chamada de modelo

de classificação - que mapeie cada conjunto de atributos *x* para um dos rótulos de classe *y* pré-determinados.

Os algoritmos de indução de árvores de decisão para classificação que utilizam uma estratégia gulosa seguem o mesmo princípio do algoritmo de Hunt. Entretanto, são implementadas funções que buscam responder às questões de particionamento e critério de paradas. Essas funções variam de acordo com o algoritmo escolhido.

O melhor particionamento é definido por meio de alguma medida em termos da distribuição das classes em relação aos exemplos antes do particionamento. Ou seja, no caso de todas as instâncias pertencerem a mesma classe, o particionamento é homogêneo, ou puro. Para melhores resultados busca-se encontrar o menor grau de impureza para esse particionamento. Exemplos de medidas de impureza do nodo são a Entropia (QUINLAN, 1986) e Gini (BREIMAN et al., 1984).

Seja *c* o número de classes e *p(i|t)* a fração de exemplos que pertencem à classe *i* para um dado nodo *t*, a Entropia e Gini podem ser definidos como:

$$Entropia(t) = -\sum_{i=0}^{c-1} p(i|t) log_2 p(i|t) \tag{2.1}$$

$$Gini(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2 \tag{2.2}$$

A partir dessas medidas, pode-se calcular quão bem um dado teste ocorreu. Para isso, é calculado o total de impureza após a partição, ou seja, o ganho de informação da partição.

$$GanhoInfo = I(nodo) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) \tag{2.3}$$

Onde *I(.)* é o cálculo da medida de impureza de um determinado nodo, *N* é o número total de registros do nodo pai, *k* é o número de atributos e $N(v_j)$ é o número de registros associados ao nodo filho v_j.

Um algoritmo de indução de árvore de decisão para classificação pode ser implementado conforme o Algoritmo 1 (ALPAYDIN, 2010).

A qualidade dos modelos induzidos pode ser avaliada usando diferentes métricas, dentre as quais pode-se citar: acurácia, medida-F e tamanho da árvore.

A acurácia representa o quão satisfatório foi a classificação segundo o modelo de classificação. Uma maneira de avaliar a acurácia é fazer a indução do modelo a partir de um método

Algoritmo 1: Árvore de decisão para classificação, adaptado de (ALPAYDIN, 2010)

```
1  Função GeraArvore(X)
2      se Impureza(X)<Θ_1 então
3          Cria folha rotulada com a maioria das classes em X
4          retorna
5      fim
6      i ← ParticionaAtributo(X)
7      para cada cada ramo de X faça
8          Encontre X_i seguindo no ramo
9          GeraArvore(X_i)
10     fim
11 Função ParticionaAtributo(X)
12     MinImpureza ← MAX
13     para todo atributo i=1,...,d faça
14         se x_i é discreto com n valores então
15             Particiona X em X_i,...,X_n por x_i
16             e ← GanhoInfo(X_1, ..., X_n)
17             se e < MinImpureza então
18                 MinImpureza ← e
19             fim
20             senão
21                 MelhorParticao ← i
22             fim
23         fim
24         senão
25             para todo partição possível faça
26                 Particiona X em X_i, X_2 sobre x_i
27                 e ← GanhoInfo(X_1, X_2)
28                 se e < MinImpureza então
29                     MinImpureza ← e
30                 fim
31                 senão
32                     MelhorParticao ← i
33                 fim
34             fim
35         fim
36     fim
37     retorna MelhorParticao
```

denominado validação cruzada (WITTEN; FRANK, 2011). Na validação cruzada, o conjunto de dados é dividido em *n* partições, onde $n - 1$ partições são utilizadas para treino e a única restante é utilizada para teste, *n* vezes. Então, são determinadas as instâncias que foram preditas corretamente. Tem-se:

- Número de instâncias classificadas como verdadeiro positivo (VP);

- Número de instâncias classificadas como verdadeiro negativo (VN);
- Número de instâncias classificadas como falso positivo (FP);
- Número de instâncias classificadas como falso negativo (FN).

A acurácia é calculada a partir da divisão das predições corretas sobre o total de predições, conforme:

$$\textit{Acurácia} = \frac{\textit{Predições Corretas}}{\textit{Total Predições}} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.4}$$

Uma outra maneira de avaliar modelos de classificação é por meio da medida-F. A medida-F é uma métrica que faz uso de duas outras medidas chamadas de precisão e revocação, e apresenta valores entre 0 e 1. De acordo com Han & Kamber (2011), essas medidas são calculadas da seguinte forma:

$$\textit{Precisão} = \frac{|Relevantes \cap Recuperadas|}{|Recuperadas|} \tag{2.5}$$

$$\textit{Revocação} = \frac{|Relevantes \cap Recuperadas|}{|Relevantes|} \tag{2.6}$$

Onde:

- $|Relevantes \cap Recuperadas|$: as instâncias que foram recuperadas corretamente (VP);
- $|Recuperadas|$: as instâncias que foram recuperadas, corretas ou incorretas (VP + FP);
- $|Relevantes|$: todas as instâncias que foram recuperadas corretamente, mais as instâncias que não foram recuperadas, mas que deveriam ter sido (VP + FN).

Portanto, a medida-F é dada por:

$$\textit{Medida-F} = \frac{\textit{Precisão} \times \textit{Revocação}}{(\textit{Precisão} + \textit{Revocação})/2} \tag{2.7}$$

Uma terceira métrica para avaliar não a qualidade do modelo, mas a compreensibilidade do mesmo, é o tamanho da árvore. Essa métrica diz respeito a profundidade da árvore.

2.2.2 Indução de Árvores de Decisão Para Regressão

Regressão é um modelo de predição que se diferencia pelo fato de seu atributo alvo ser contínuo. Portanto, é a tarefa de aprender uma função alvo f que mapeia cada conjunto de atributos *X* para uma saída de valores contínuos *y*. São chamadas de árvores de decisão para regressão aquelas cuja função f mapeada para cada nodo folha contém a média dos valores de *y* para os exemplos que compõe uma folha. Além de árvores de regressão existem também árvores que rotulam cada nodo como um modelo de regressão linear. Essa técnica é chamada de árvores modelo.

Segundo Witten et al (2011) e Alpaydim (2010), tanto árvores de regressão quanto árvores modelo seguem o mesmo princípio de indução de árvores para classificação, mudando o foco do método de particionamento e critério de parada. Um dos algoritmos mais utilizados para árvores de regressão e árvores modelo é o M5P (QUINLAN et al., 1992) (WANG; WITTEN, 1997). O maior objetivo do M5P é maximizar a redução do desvio padrão (RDP), considerando o desvio padrão dos exemplos no conjunto de dados $dp(X)$.

$$RDP = dp(X) - \sum_{i} \frac{|X_i|}{|X|} \times dp(X_i) \tag{2.8}$$

O algoritmo particiona os atributos de acordo com o RDP: para cada ciclo recursivo da indução, é escolhido o atributo que apresenta o maior valor de RDP. O critério de parada considera limiares do número de exemplos em *X* e um limiar em relação a uma taxa do $dp(X)$. No Algoritmo 2 é descrito o pseudocódigo adaptado do M5P.

A regressão tem como objetivo induzir um modelo que sua função f minimize o erro. As típicas funções de erro para tarefas de regressão são Erro Médio Absoluto (MAE - Mean Absolute Error) e Erro Médio Quadrático (RMSE - Root Mean Squared Error). Ambas retornam valores entre 0 e 1, sendo o menor valor, o melhor resultado. Sendo p o valor predito e a o valor real:

$$MAE = \frac{|(p_1 - a_1) + \cdots + (p_n - a_n)|}{n} \tag{2.9}$$

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}} \tag{2.10}$$

É possível medir a correlação estatística entre a e p. Esses valores variam entre 0 e 1, onde 1 apresenta correlação perfeita e 0 ausência de correlação. O valor -1 também é válido,

significando uma correlação inversa perfeita.

$$\textit{Correlação} = \frac{S_{PA}}{\sqrt{S_p S_A}} \tag{2.11}$$

$$\textit{onde} \quad S_{PA} = \frac{\sum_i (p_i - \overline{p})(a_i - \overline{a})}{n-1}, \quad S_P = \frac{\sum_i (p_i - \overline{p})}{n-1}, \quad e \quad S_A = \frac{\sum_i (a_i - \overline{a})}{n-1}$$

O algoritmo 3D-Tri proposto por Winck (2012) trata de regressão e também utiliza o RDP como critério para a partição dos atributos.

Algoritmo 2: Algoritmo de Árvore de decisão para regressão, adaptado de (ALPAYDIN, 2010) e (WANG; WITTEN, 1997)

```
1  Função GeraArvore(X)
2      se DP não foi calculado então
3          DP ← dp(X)
4      fim
5      se Número de exemplos em X < 4 ou dp(X) < 0.05 × DP então
6          Cria folha rotulada com a média dos valores de y em X
7          retorna
8      fim
9      i ← ParticionaAtributo(X)
10     para cada cada ramo de X faça
11         Encontre X_i seguindo no ramo
12         GeraArvore(X_i)
13     fim
14 Função ParticionaAtributo(X)
15     para todo atributo i=1,...,d faça
16         Calcula RDP
17         se x_i é discreto com n valores então
18             Particiona X em X_i,...,X_n por x_i
19             MelhorParticao ← i com maior valor de RDP
20         fim
21         senão
22             para todo partição possível faça
23                 Particiona X em X_i, X_2 sobre x_i
24                 MelhorParticao ← i com maior valor de RDP
25             fim
26         fim
27     fim
28     retorna MelhorParticao
```

2.3 Considerações

Os temas abordados neste capítulo descreveram conceitos importantes para compreensão deste trabalho. No contexto da bioinformática, foram apresentados as questões que envolvem o RDD. No ponto de vista computacional, os conceitos de indução de árvores de decisão, em especial para regressão, foram discutidos neste capítulo. A indução de árvores de decisão é uma técnica bastante utilizada para classificação de dados provenientes de docagem molecular. Essa relação pode ser observada ao longo deste trabalho.

No próximo capítulo será abordado o algoritmo 3D-Tri - um algoritmo de indução de árvores de decisão para regressão que interpreta as propriedades tridimensionais dos átomos em dados provenientes de dinâmica molecular.

3 MATERIAIS E MÉTODOS

Este capítulo apresenta o algoritmo 3D-Tri proposto em Winck (2012) e implementado neste trabalho.

3.1 Algoritmo 3D-Tri

Esta seção trata do algoritmo 3D-Tri proposto em Winck (2012). A subseção 3.1.1 define o algoritmo e a etapa de pré-processamento. Na subseção 3.1.2 é descrito o processo de definição do bloco. E, por fim, o processo de indução da árvore é discutido na subseção 3.1.3.

3.1.1 Definição e pré-processamento

O algoritmo 3D-Tri, proposto por Winck (2012), é um algoritmo de indução de árvore de decisão para regressão capaz de interpretar propriedades tridimensionais no formato x, y, z, e induzir uma árvore que representa essas propriedades, predizendo um valor de *FEB*. A estratégia consiste em minerar dados de simulações por DM, considerando as propriedades tridimensionais (3D) de cada conformação do receptor. Essa estratégia é diferente da convencional, que faz uso da distância entre os átomos dos resíduos do receptor e os átomos do ligante sendo considerado. O algoritmo assume como atributos preditivos as coordenadas espaciais no espaço euclidiano de cada átomo dos resíduos do receptor, em cada uma de suas conformações. Em tal estratégia, os valores de *FEB* para cada conformação ainda são considerados como atributo alvo (WINCK, 2012).

A primeira etapa consiste no pré-processamento de dados provenientes de simulações por DM para geração de um conjunto de dados apropriados. Na Tabela 3.1 pode-se observar um exemplo de um conjunto de dados gerado para um ligante. Esse conjunto de dados gerados após a primeira etapa não pode ser corretamente interpretado por um algoritmo convencional de indução de árvores de decisão. Isso porque um algoritmo convencional consideraria cada atributo individualmente, em vez de considerar a relação entre cada três atributos, os quais representam uma propriedade tridimensional. Como resultado, o modelo induzido poderia não ser preciso, e sua interpretação poderia apresentar distorções. O algoritmo 3D-Tri é capaz de ler esse conjunto de dados e interpretar suas propriedades tridimensionais, e a partir dele, induzir uma árvore com base nessas propriedades.

Tabela 3.1 – Exemplo de um conjunto de dados gerado para um ligante, adaptado de (WINCK, 2012)

x1 ALA1_N	y1 ALA1_N	z1 ALA1_N	...	x4008 LEU268_OXT	y4008 LEU268_OXT	z4008 LEU268_OXT	FEB
15,838	-20,060	8,807	...	-20,647	-17,858	-3,495	-11,22
15,665	-19,974	7,918	...	-20,600	-17,957	-3,176	-11,21
...	...	...	...	...	...	...	...
14,959	-14,885	-18,370	...	21,498	16,662	-11,545	-1,00

Algoritmos de indução de árvore são basicamente implementados obedecendo a uma estratégia de dividir para conquistar, onde um conjunto de dados *X* é dividido em regiões locais a fim de predizer o atributo alvo. No algoritmo 3D-Tri, o objetivo principal é fazer com que essa divisão represente regiões em um espaço euclidiano para um dado atributo preditivo, em função de um valor de *FEB* (WINCK, 2012).

No contexto dos dados já processados, um átomo é um atributo preditivo. Para cada nodo da árvore executa-se duas divisões, onde um nodo é um átomo e as arestas testam se os objetos sendo avaliados fazem parte de um dado intervalo $[(x_i, x_f), (y_i, y_f), (z_i, z_f)]$, onde i indica a posição inicial de uma coordenada e f representa sua posição final (WINCK, 2012). A Figura 3.1 ilustra a ideia dessa divisão, onde o nodo é um dado átomo e o teste identifica se as instâncias estão *dentro* ou *fora* do intervalo.

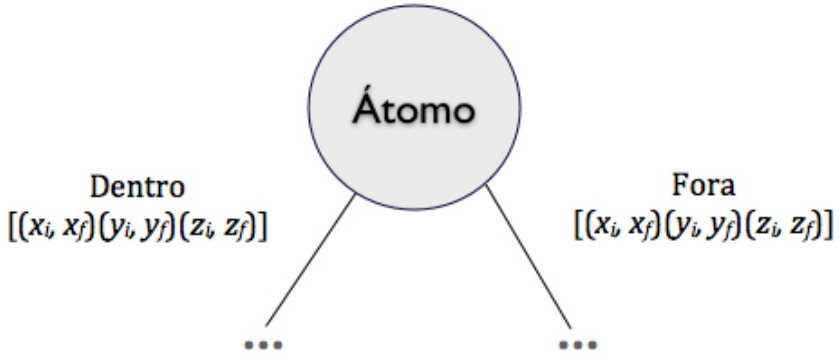


Figura 3.1 – Divisão de um nodo pelo pelo algoritmo 3D-Tri (WINCK, 2012)

O algoritmo 3D-Tri apresenta dois módulos principais:

- O primeiro diz respeito à definição do melhor intervalo, ou bloco, para cada átomo, sendo possível induzir uma árvore binária a partir desse intervalo;
- O segundo refere-se à indução recursiva da árvore a partir das árvores binárias induzidas no módulo anterior.

3.1.2 Definição do bloco

Para gerar as árvores binárias, deve-se primeiro identificar o intervalo que um dado átomo deve estar considerando as instâncias (ou conformações) envolvidas, para que haja um bom valor de *FEB* (WINCK, 2012). Para isso, cada átomo do *Dataset* é submetido a uma estratégia de agrupamento, como o K-means (HARTIGAN; WONG, 1979). Em trabalhos posteriores a Winck (2012) foi constatado o K-means como sendo o melhor algoritmo de agrupamento para os tipos de dados utilizados no algoritmo 3D-Tri (MACHADO, 2014).

O algoritmo K-means assume um valor *k* como parâmetro e particiona um dado conjunto de dados *X* em *k* grupos, apresentando uma alta similaridade entre objetos em um mesmo grupo, e baixa similaridade entre objetos de grupos distintos. O algoritmo seleciona *k* objetos, de maneira aleatória, onde cada objeto representa, inicialmente, uma média ou centro de m dos *k* grupos. Então, os objetos restantes são distribuídos ao grupo de maior similaridade, e uma nova média do grupo é calculada, até que haja uma convergência entre os objetos (HAN; KAMBER, 2011).

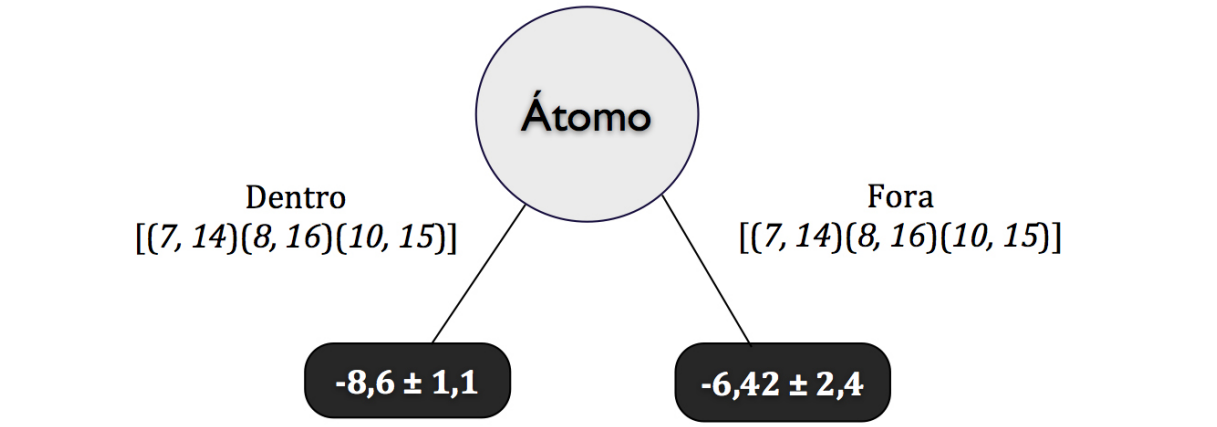
Após a execução do K-means, cada um dos átomos está dividido em *k* grupos. Para as instâncias de cada *k* grupo será retornado:

- O valor médio de cada coordenada espacial, obtendo $\overline{x}, \overline{y}, \overline{z}$;
- O valor médio de *FEB*, definido por $\overline{FEB}$.

Após a execução do K-means, para cada átomo é necessário eleger *k* grupos a ser considerado (GE). Como resultado, tem-se as coordenadas e FEB centroide. A partir do centroide, é feito o cálculo da distância euclidiana para cada instância. Então, obtêm-se o valor mínimo e máximo dessas distâncias. Após aplicar um critério de parada, é possível definir o intervalo $[(x_i, x_f), (y_i, y_f), (z_i, z_f)]$.

A Figura 3.2 mostra a árvore binária gerada pelo conjunto de dados que tem o bloco definido pelas coordenadas $[(7, 14)(8, 16)(10, 15)]$. O nodo folha contém o $\overline{FEB}$ para todas as instâncias t de um *Dataset* que pertencem a cada uma das arestas da árvore. Para ilustrar melhor, abaixo dos nodos folha são apresentadas as instâncias do *Dataset* com suas respectivas coordenadas e valores de FEB.

Esse processo é repetido para cada átomo do *Dataset*, de modo que existe uma árvore binária para cada um dos átomos. O próximo passo, é induzir a árvore. Para isso, é necessário



SS	x	y	z	FEB
10	14	16	13	-7
13	11	12	13	-9
14	7	14	12	-8
16	13	8	15	-10
18	13	13	10	-9

SS	x	y	z	FEB
1	2	2	3	-7
2	3	7	9	-5
3	5	8	2	-7
4	7	5	4	-8
5	9	6	7	-8
6	8	3	5	-7
7	2	4	3	-4
8	6	7	4	-7
9	4	3	8	-3
11	10	17	14	-9
12	8	10	8	-11
15	15	9	16	-4
17	14	7	12	-3
19	15	14	7	-8

Figura 3.2 – Árvore binária induzida (WINCK, 2012)

escolher um desses átomos. Para isso, calcula-se o RDP de cada bloco, de modo a retornar o bloco com o maior valor de RDP.

3.1.3 Indução da árvore

Os critérios de construção do bloco foram definidos. O próximo passo é estabelecer o critério para indução da árvore. Para isso, uma técnica clássica de indução de árvore é utilizada, construída recursivamente a partir de uma abordagem top-down. Como o *Dataset* contém como atributo alvo o valor estimado de FEB e o mesmo é um atributo número, trata-se de uma indução de árvore de regressão.

O procedimento principal da indução da árvore recebe como parâmetro o conjunto de dados *Dataset*. Na primeira execução desse procedimento é calculado o valor do desvio padrão do valor de FEB de todas as instâncias do *Dataset*.

$$DP = dp(FEB_{Dataset}) \tag{3.1}$$

O valor de DP é utilizado para verificação do critério de parada (*CritParada*) de indução. Esse critério de parada avalia o tamanho do *Dataset* ($Tam_{Dataset}$) em relação a uma população mínima definida por ($PopMinima$) de instâncias que devem fazer parte de uma ramificação, e o desvio padrão das instâncias de *Dataset* sendo avaliado em relação a uma taxa do valor de DP ($TaxaDP$), sendo:

$$CritParada = \begin{cases} V & \text{se} \quad (Tam_{Dataset} < PopMinima) \\ & \text{ou} \quad (dp(Dataset) < TaxaDP * DP) \\ F & \text{caso contrário} \end{cases} \tag{3.2}$$

O processo de indução é parado obedecendo o $CritParada$, então é calculado o valor médio de FEB do *Dataset* ($\overline{FEB}_{Dataset}$), criando-se um nodo folha para esta aresta da árvore, rotulando tal nodo como $\overline{FEB}_{Dataset}$:

$$Nodo \leftarrow Folha(\overline{FEB}_{Dataset}) \tag{3.3}$$

Caso o critério de parada não for obedecido, então é chamado o procedimento de definição de bloco ($DefineBloco$), passando como parâmetro $Dataset$ e DP, retornando para a estrutura de dados $BlocoNodo$ o átomo e o intervalo de coordenadas correspondente. Assim, um novo nodo é criado na árvore:

$$Nodo \leftarrow BlocoNodo \tag{3.4}$$

Para cada aresta ar deste nodo, cujo rótulo é $Bloco$, testa-se as instâncias que estão $dentro$ ou $fora$ deste intervalo. Atualiza-se o *Dataset* com essas instâncias para cada aresta ($Dataset_{ar}$), e faz-se uma chamada recursiva do procedimento $InduzArvore$, passando como parâmetro $Dataset_{ar}$. O Algoritmo 3 apresenta o pseudocódigo para o processo recursivo de indução de árvore para o conjunto de dados *Dataset*.

Para avaliar-se a árvore induzida pelo Algoritmo 3 pode-se usar os mesmos critérios

tradicionais de indução de árvore de regressão, utilizando métricas de erro MAE e $RMSE$.

Algoritmo 3: Indução da Árvore, adaptado de (WINCK, 2012)

Seja $Dataset_{a\times 3+1}$ uma matriz bidimensional de t linhas, chamadas instâncias, e $a \times 3 + 1$ colunas, geradas a partir do pré-processamento
Seja $Dataset_n$ uma submatriz de $Dataset$ contendo um subconjunto n de instâncias
Seja a um átomo em $Dataset$
Seja DP o desvio padrão calculado para as instâncias sendo computadas
Seja $PopMinima$ um parâmetro que indica qual a população mínima de instâncias em cada divisão dos nodos
Seja $TaxaDP$ um parâmetro de erro para a divisão dos nodos
Seja $BlocoNodo$ uma estrutura de dados que contém $(a, [(x_i, x_f)(y_i, y_f)(z_i, z_f)])$ para a divisão dos nodos
Função `InduzArvore(`$Dataset$`)`
    se DP *não foi Computado* **então**
        Computa DP
    fim
    se $CritParada$ **então**
        Computa $\overline{FEB}_{Dataset}$
        $Nodo \leftarrow Folha(\overline{FEB}_{Dataset})$
    fim
    senão
        $BlocoNodo \leftarrow DefineBloco(Dataset, SD)$
        $Nodo \leftarrow BlocoNodo$
        para cada *aresta* ar *de* $Nodo$ **faça**
            $Dataset_{ar} \leftarrow$ instâncias que fazem parte do teste da aresta
            InduzArvore($Dataset_{ar}$)
        fim
    fim

3.2 Considerações

O algoritmo 3D-Tri é capaz de interpretar dados tridimensionais como atributos preditivos relacionados a posição de cada átomo no espaço euclidiano. O modelo gerado a partir do algoritmo 3D-tri pode ser melhor interpretado por um especialista de domínio, facilitando a seleção de conformações para futuros experimentos de docagem molecular (WINCK, 2012).

Este trabalho implementa a indução de árvores de decisão para regressão segundo o algoritmo 3D-Tri, conforme o Algoritmo 3. O próximo capítulo descreve a metodologia utilizada neste trabalho.

4 METODOLOGIA

Diferentes trabalhos foram desenvolvidos com objetivo de minerar dados de docagem molecular e dinâmica molecular para tentar encontrar as conformações de um receptor que sejam mais promissoras para um determinado ligante. Este trabalho consiste na implementação da indução da árvore de regressão conforme o algoritmo 3D-Tri. Para tanto, contribuições anteriores foram necessárias. No trabalho desenvolvido por Machado (2014), foi desenvolvido o módulo de agrupamento de dados para o *Dataset* tridimensional na linguagem Python. Além disso, foi constatado que o K-means é o algoritmo de agrupamento de dados mais promissor para dados de *DM*. A tarefa de geração do melhor intervalo foi desenvolvida por alunos do grupo de pesquisa.

A partir de um *Dataset* com dados de DM pré-processado, o algoritmo de indução da árvore de decisão gera, para cada iteração, um bloco com um intervalo onde há melhor valor de FEB para um determinado átomo escolhido pelo critério de divisão da árvore. Esse átomo, com seu respectivo bloco, é então um nodo não-terminal da árvore. Para cada nodo não-terminal, as instâncias são atribuídas para as arestas definidas por *dentro* e *fora*, de acordo com a posição do átomo dessa instância em relação ao bloco.

A escolha do bloco para a divisão é feita por meio da chamada da função *DefineBloco*, implementada por alunos do grupo de pesquisa e utilizada neste trabalho. O valor de RDP é utilizado como critério de escolha do bloco, escolhendo-se então o bloco de maior RDP. O objetivo então é obter divisões onde haja um menor desvio padrão, e, consequentemente, divisões onde as instâncias tenham valor de atributo alvo, *FEB*, mais próximos uns dos outros.

O critério de parada do algoritmo foi proposto por Winck (2012), e utiliza como um dos critérios o valor do desvio padrão do subconjunto do Dataset em relação a uma taxa mínima de desvio padrão em relação ao desvio padrão do *Dataset* original. Outro critério avaliado é o tamanho do subconjunto de *Dataset* em relação a um parâmetro de população mínima. O critério de parada é definido na equação 3.2.

A linguagem de programação Python foi escolhida devido a ampla quantidade de bibliotecas desenvolvidas para mineração de dados e aprendizagem de máquina disponíveis para a mesma. Umas das bibliotecas mais utilizadas é a Scipy e a Scikit-Learn. A Scikit-Learn é um exemplo de uma biblioteca amplamente utilizada, bem documentada, e com diversos recursos para aprendizagem de máquina. Além desses critérios, a linguagem Python vem sendo bastante

utilizada para mineração de dados, sendo também uma linguagem fácil tanto para escrita como para leitura.

4.1 Implementação

Buscou-se, neste trabalho, implementar o algoritmo 3D-Tri seguindo os passos indicados em Winck (2012) e descritos no capítulo 3. Para tanto, a implementação contém uma função principal, chamada de *InduzArvore* conforme o Algoritmo 3. Adicionalmente, foi criada uma classe chamada *Arvore* e outra classe chamada *Nodo*, no intuito de se representar o modelo de árvore e seus nodos, respectivamente.

Alguns pacotes disponíveis para linguagem Python foram utilizados, como o Numpy. Esse pacote suporta vetores e matrizes multidimensionais, possuindo uma vasta coleção de funções matemáticas para trabalhar com estas estruturas. A função auxiliar *DefineBloco* foi implementada anteriormente em trabalhos no grupo de pesquisa. Além dessa, outras funções auxiliares foram utilizadas, como *computaDP*, para computar o desvio padrão, e *computaFEB-Dataset*, para calcular o *FEB* médio do *Dataset*.

A função principal *InduzArvore* inicia-se computando o desvio padrão e verificando os critérios de parada. Caso o critério de parada seja atendido, é criado um nodo folha com o valor de FEB, e adicionalmente nesta implementação, a informação do desvio padrão para as instâncias deste nodo. Caso o critério de parada não seja atendido, a função *DefineBloco* é chamada, e a partir da escolha do átomo e do bloco, é criado um nodo não-terminal tendo como informação o nome do átomo e seu respectivo intervalo de bloco. As instâncias são divididas em instâncias que estão dentro e fora do intervalo, criando dois novos subconjuntos. É verificado então, se um dos dois novos subconjuntos é vazio, e, caso positivo, o nodo é transformado em nodo terminal com as informações de um nodo terminal: FEB e desvio padrão. Caso contrário, a função *InduzArvore* é chamada recursivamente para cada uma das arestas do nodo.

Para avaliar o modelo da árvore, também foi criado as funções *calculaMAE* e *calculaRMSE*, para calcular o *MAE* e *RMSE* respectivamente.

4.2 Dados utilizados

O teste do algoritmo implementado neste trabalho utilizou dados do ligante NUNL2, conhecido como um inibidor da bomba de efluxo AcrB *(Acriflavine resistance protein B)*. As

bombas de efluxo podem ser entendidas como uma forma das bactérias se defenderem da presença de antibióticos, como por exemplo, a penicilina (PIDDOCK, 2006). O receptor, AcrB, é uma proteína relacionada ao mecanismo de efluxo (PRATES, 2014). Os dados utilizados neste trabalho foram gerados por simulações de DM por Prates (2014).

No *Dataset* pré-processado original havia 9.661 átomos, 1.029 resíduos e 1.001 conformações, totalizando 28.984 colunas e 1.001 linhas. Entretanto, tal dimensionalidade torna a execução muito dispendiosa. Então, a fim de reduzir a dimensionalidade para fins de validação do algoritmo desenvolvido, cinco resíduos foram escolhidos de forma aleatória: *ARG_418*, *ARG_650*, *ASP_784*, *GLN_228*, *PHE_386*. Os átomos de hidrogênio (H) também foram desconsiderados. Após a redução de dimensionalidade, o *Dataset* passou a ter 107 átomos, totalizando 322 colunas.

A Tabela 4.1 ilustra as coordenadas *x*, *y*, *z* das três primeiras e três últimas conformações do *Dataset* utilizado, onde fazem parte os resíduos SER_134 e LEU_843, respectivamente. As siglas dos átomos N e O são acompanhadas do número sequencial com que os mesmos aparecem no arquivo PDB da proteína AcrB.

Tabela 4.1 – Exemplo de coordenadas para o *Dataset* utilizado

SER_134_ N_1173_x	SER_134_ N_1173_y	SER_134_ N_1173_z	... ...	LEU_843_ O_7922_x	LEU_843_ O_7922_y	LEU_843_ O_7922_z	FEB
81.710	78.490	69.450	...	70.210	77.050	97.680	-7.6
78.350	80.900	76.060	...	73.630	110.870	76.410	-8.2
80.300	75.890	72.300	...	74.320	89.360	98.180	-8.1
...	...	...	...	...	...	...	...
80.900	78.780	70.300	...	74.490	91.750	97.880	-8.6
80.500	79.040	70.780	...	75.960	89.200	98.640	-8.5
78.460	75.970	72.270	...	72.880	93.260	97.540	-7.1

4.3 Parâmetros utilizados

Os parâmetros utilizados neste trabalho são definidos segundo o Algoritmo 3D-Tri, com os seguintes valores:

- **Número de grupos** - Foram definidos dois grupos (k = 2) para a identificação do centroide na geração do bloco no algoritmo *DefineBloco*;
- **Taxa de erro na expansão do bloco** - O limite da taxa de erro utilizado como critério de parada da expansão do bloco foi definido como 0,05;

- **Taxa de população na expansão do bloco** - Segundo os critérios de paradas do algoritmo, o bloco é expandido até atingir o limite de erro do item anterior, ou enquanto o número de exemplos que fazem parte do bloco for inferior a taxa de população mínima. Essa taxa foi definida em 0,05 em relação ao número de exemplos sendo computados;
- **Taxa do desvio padrão para a indução** - Foi definido uma taxa de 0,1 para o desvio padrão dos exemplos sendo computados, para o critério de parada da indução.

4.4 Avaliação do modelo

O modelo induzido pelo algoritmo implementado neste trabalho será observado as seguintes métricas:

- **Erros** - São calculados os erro médio absoluto (MAE) e erro médio quadrático (RMSE) para as instâncias do *Dataset*;
- **Número de nodos** - São observados quantos nodos internos e nodos folha compõem o modelo induzido;
- **Profundidade** - É avaliada qual a profundidade máxima da árvore, considerando os nodos folha;
- **Tempo de execução** - É avaliado o tempo de execução do algoritmo para o *Dataset*;
- **Desvio padrão dos nodos** - São observados o desvio padrão dos nodos folha.

5 RESULTADOS

Neste capítulo é apresentada a árvore resultante do modelo induzido para o algoritmo implementado neste trabalho. A Figura 5.1 ilustra o modelo de árvore para o *Dataset* utilizado.

O conjunto de dados, representados conforme exemplo da Tabela 4.1, é utilizado para induzir a árvore da Figura 5.1. Os nodos indicam o átomo sendo testados, indicado pelo nome do átomo incluindo o nome do resíduo e a posição em relação ao *PDB* original. As arestas representam o teste do intervalo das coordenadas x, y, z para o átomo. As arestas à esquerda correspondem às instâncias que fazem parte do intervalo, e as arestas à direita às instâncias que não fazem parte do intervalo. Os nodos folha contém o valor médio de *FEB* e *DP* de suas instâncias.

Para avaliar os resultados, as métricas de erro, número de nodos e profundidade da árvore estão detalhadas na Tabela 5.1.

Tabela 5.1 – Métricas de avaliação do modelo induzido

Métrica	Valor
MAE	2.2537
RMSE	0.7210
Nodos Internos	15
Nodos Folha	16
Profundidade	8
Tempo de execução	1min 51s

A redução do *Dataset* para 107 átomos reduziu significativamente o tempo de execução. Em outros testes verificou-se que para *Dataset* maiores o tempo aumenta consideravelmente. Portanto, um pré-processamento visando a redução do *Dataset* é desejável na maioria dos casos.

Durante os testes, tornou-se necessário a criação de mais um critério de parada. O algoritmo original proposto por Winck (2012) não previa casos em que após a escolha do nodo resulta em um conjunto vazio. Ou seja, todas as instâncias presentes dentro, ou todas as instâncias fora do intervalo. Nesses casos, para evitar uma recursão sem parada, o nodo torna-se folha. Esse processo, ao contrário dos demais critérios de parada, não ocorre no início da função, mas sim no final. Isso acontece porque o conhecimento de que todas as instâncias estão dentro ou fora do bloco só acontece após a definição do bloco, ou seja, no meio do algoritmo.

O modelo induzido tem em seus nodos referência à posição de coordenadas espaciais, até atingir o nodo folha. Essa é uma característica positiva do algoritmo 3D-Tri, que permite ao

especialista de domínio uma melhor interpretação, facilitando na seleção de conformações para futuros experimentos de docagem (WINCK, 2012). Tal característica não ocorre em algoritmos de indução de árvore de decisão convencionais, como o M5P. Isso ressalta a importância do algoritmo para indução de dados de docagem molecular, e consequentemente a importância deste trabalho no contexto da bioinformática.

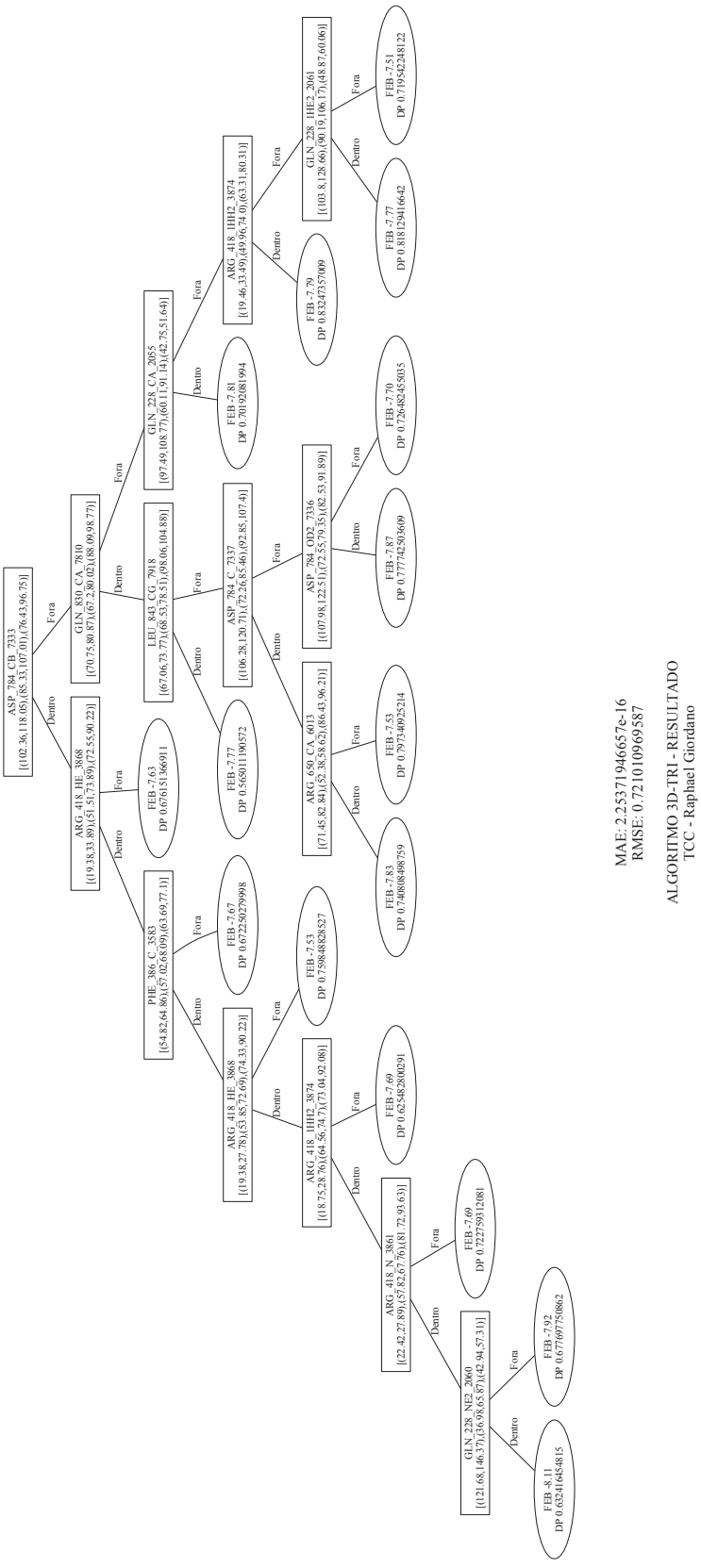


Figura 5.1 – Modelo de árvore induzido

6 CONCLUSÃO

Este trabalho teve como objetivo a implementação do algoritmo 3D-Tri proposto por Winck (2012) na linguagem Python. Está inserido no contexto de RDD, onde o principal objetivo é minerar dados de docagem molecular com o objetivo de selecionar conformações promissoras do receptor para um determinado ligante e, assim, reduzir o tempo de execução em novos experimentos de docagem com o objetivo de auxiliar no processo de surgimento de um novo fármaco.

Os dados utilizados neste trabalho foram resultado de simulação por DM para a proteína AcrB, uma proteína relacionada ao mecanismo de efluxo. Para tanto foi feito um pré-processamento dos dados a fim de reduzir a dimensionalidade do *Dataset*. Após essa redução, o número de átomos foi reduzido de 9.661 para 107, possibilitando uma execução mais rápida do algoritmo.

Durante os testes, tornou-se necessária a criação de mais um critério de parada, além dos propostos originalmente no algoritmo. Esse critério avalia, após a criação do nodo, se todos as instâncias pertencem a mesma aresta do teste. Caso positivo, o nodo torna-se folha.

Foram utilizadas métricas de avaliação do modelo induzido de acordo com o proposto por Winck (2012). Adicionalmente, foi adicionado o desvio padrão para cada nodo.

Este trabalho atingiu o objetivo de dar continuidade ao trabalho desenvolvido por Winck (2012), implementando o algoritmo em uma linguagem amplamente utilizada no contexto de mineração de dados. A partir deste trabalho, é possível também uma maior integração dos algoritmos implementados no grupo de pesquisa e assim criar uma API (*Application Programming Interface*) para o algoritmo 3D-Tri, podendo assim aumentar significativamente a facilidade e rapidez no uso do algoritmo. A existência de uma API cria também a oportunidade de melhoramentos para o algoritmo. Sendo assim, a partir deste trabalho, novas aplicações do algoritmo 3D-Tri poderão surgir, e como resultado, os melhores ligantes poderão ser testados *in vitro* e então novos fármacos poderão surgir.

REFERÊNCIAS

ALPAYDIN, E. **Introduction to machine learning**. Boston: MIT press, 2010.

BALAKIN, K. **Pharmaceutical data mining**: approaches and applications for drug discovery. New York: John Wiley Sons, 2009.

BERMAN, H. M. et al. PDB - Protein Data Bank. **Nucleic Acids Research**, [S.l.], v.28, p.235–242, 2000.

BREIMAN, L. et al. **Classification and Regression Trees**. Belmont: Wadsworth International Group, 1984.

FREITAS, A. A.; WIESER, D. C.; APWEILER, R. On the importance of comprehensible classification models for protein function prediction. **IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)**, [S.l.], v.99, n.1, p.172–182, 2010.

GUNSTEREN, W. F. van; BERENDSEN, H. J. Computer simulation of molecular dynamics: methodology, applications, and perspectives in chemistry. **Angewandte Chemie International Edition in English**, [S.l.], v.29, n.9, p.992–1023, 1990.

HAN, J.; KAMBER, M. **Data Mining**: concepts and techniques. 3rd.ed. New York: Morgan Kaufmann, 2011.

HARTIGAN, J. A.; WONG, M. A. A k-means clustering algorithm. **Applied Statistics**, [S.l.], p.100–108, 1979.

KUNTZ, I. D. Structure-based strategies for drug design and discovery. **Science**, [S.l.], v.257, n.5073, p.1078–1082, 1992.

LESK, A. **Introduction to bioinformatics**. [S.l.]: Oxford University Press, 2002.

LIN, J. et al. Computational drug design accommodating receptor flexibility: the relaxed complex scheme. **Journal of the American Chemical Society**, [S.l.], v.124, p.5632–5633, 2002.

LUSCOMBE, N. M.; GREENBAUM, D.; GERSTEIN, M. **What is bioinformatics? a proposed definition and overview of the field**. [S.l.]: Methods Information in Medicine, 2001. 346–358p. v.40.

LYBRAND, T. P. Ligand—protein docking and rational drug design. **Current Opinion in Structural Biology**, [S.l.], v.5, n.2, p.224–228, 1995.

MACHADO, O. Um estudo comparativo de algoritmos de agrupamento de dados para dados de docagem molecular. , [S.l.], 2014.

MORRIS, G. M. et al. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. **Journal of computational chemistry**, [S.l.], v.19, n.14, p.1639–1662, 1998.

MOUNT, D. W. Sequence and genome analysis. **Bioinformatics: Cold Spring Harbour Laboratory Press: Cold Spring Harbour**, [S.l.], v.2, 2004.

PIDDOCK, L. J. Multidrug-resistance efflux pumps? not just for resistance. **Nature Reviews Microbiology**, [S.l.], v.4, n.8, p.629–636, 2006.

PRATES, N. S. **Simulações por Dinâmica Molecular e Agrupamento de Estruturas da Proteína da Bomba de Efluxo AcrB**. 2014.

QUINLAN, J. R. Induction of decision trees. **Machine learning**, [S.l.], v.1, n.1, p.81–106, 1986.

QUINLAN, J. R. **C4.5**: programs for machine learning. [S.l.]: Morgan Kaufmann, 1993.

QUINLAN, J. R. et al. Learning with continuous classes. In: AUSTRALIAN JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 5. **Anais...** [S.l.: s.n.], 1992. v.92, p.343–348.

TAN, P.-N. et al. **Introduction to data mining**. [S.l.]: Pearson Addison Wesley Boston, 2006. v.1.

WANG, Y.; WITTEN, I. Inducing model trees for continuous classes. In: EUROPEAN CONFERENCE ON MACHINE LEARNING. **Anais...** [S.l.: s.n.], 1997. p.128–137.

WINCK, A. et al. Processo de KDD aplicado à bioinformática. **Tópicos em sistemas colaborativos, multimídia, web e banco de dados. Sociedade Brasileira de Computação**, [S.l.], v.1, p.159–180, 2010.

WINCK, A. T. **3D-Tri**: um algoritmo de indução de árvore de regressão para propriedades tridimensionais-um estudo sobre dados de docagem molecular considerando a flexibilidade do receptor. 2012. Tese (Doutorado em Ciência da Computação) — .

WITTEN, I. H.; FRANK, E. **Data Mining**: practical machine learning tools and techniques. [S.l.]: Morgan Kaufmann, 2011.