Análise Comparativa de Algoritmos de Árvore de Decisão para Predição de Encaminhamentos de Processos Administrativos em Órgãos Públicos

Deivid Francisco Gonzaga, Anita Maria da Rocha Fernandes

Pós Graduação em Big Data – UNIVALI – Campus Kobrasol – São José – SC
deividfg@gmail.com, anita.fernandes@univali.br

Abstract. *This article makes a comparison of all WEKA decision tree algorithms, in terms of effectiveness and efficiency, to define the best algorithm to solve the problem of forwarding erroneously forwarded administrative processes in public agencies. Describes all the algorithms, methods and metrics used and displays the results obtained, with J48 as the chosen algorithm in the end.*

Resumo. *Este artigo faz uma comparação de todos os algoritmos de árvore de decisão da ferramenta WEKA, nos quesitos eficácia e eficiência, para definir qual é o melhor algoritmo para resolver o problema de encaminhamento de processos administrativos encaminhados de maneira equivocada em órgãos públicos. Descreve todos os algoritmos, métodos e métricas utilizados e exibe os resultados obtidos, tendo no final o J48 como algoritmo escolhido.*

1. Introdução

Este trabalho tem como objetivo realizar uma análise comparativa de eficiência e eficácia em alguns algoritmos de aprendizagem de máquina na predição de encaminhamentos de processos administrativos em órgãos públicos na tentativa de solucionar o problema de encaminhamentos que não chegam ao setor de destino correto, e posterior utilização do algoritmo escolhido no sistema citado a seguir.

O sistema em questão está implantado há seis anos no órgão do estudo e consiste em um cadastro de processos administrativos físicos e digitais, que automatiza e organiza as rotinas de trabalho do servidor público, auxiliando na eficiência da máquina pública.

Contratação de funcionários, solicitação de férias, pedido de ressarcimento de viagem, pedido de aluguel de imóvel entre outros são exemplos de processos administrativos, podendo estes serem do tipo digital, que são totalmente eletrônicos ou físicos que são encaminhados em papel e eletrônico juntos.

São realizados, em média, aproximadamente 100 (cem) encaminhamentos todos os dias, somente no órgão público que está sendo analisado, e estes encaminhamentos por muitas vezes acabam sendo feitos de maneira equivocada, seja por falta de atenção ou mesmo por falta de conhecimento do servidor público.

Quando um processo é recebido pelo setor, o usuário responsável realiza os trâmites e validações necessários para dar prosseguimento, depois decide para qual setor o processo deve ir. Esta decisão é baseada no tipo de processo, em qual setor ele está, qual setor ele foi aberto e outros parâmetros.

O objetivo é utilizar estes parâmetros e indicar ao usuário qual é o setor ideal

para o encaminhamento de seu processo, ou seja, encaminhar o processo para o setor que realmente deveria ir, e assim evitar todos os transtornos causados por este tipo de problema, e principalmente as horas a mais que são gastas para resolvê-los, que acabam tornando o órgão menos produtivo e menos eficiente.

Com isso em mente, o que este artigo propõe, é o uso de algoritmos de aprendizagem de máquina do tipo árvore de decisão [Breiman et al 1984] e aprendizagem supervisionada [Caruana and Niculescu-Mizil 2006] com a ferramenta Waikato Environment for Knowledge Analysis (WEKA) [Bouckaert et al 2018].

A escolha pela ferramenta WEKA [Bouckaert et al 2018] se dá, pois o sistema deste estudo está escrito em JAVA, e a ferramenta dispõe de uma biblioteca em JAVA bem estável e com uma ótima documentação, ao qual facilitará a futura implementação da solução no sistema. Ela também é open source sobre a licença GNU (General Public License), podendo assim ser usada sem nenhuma limitação.

Quanto a escolha por árvores de decisão [Breiman et al 1984] se dá pelo fato de conter algoritmos de fácil compreensão e ótimos para resolver problemas de classificação. Eles também podem ser aplicados a qualquer tipo de dado, que é o panorama deste estudo.

E são de aprendizagem supervisionada [Caruana and Niculescu-Mizil 2006] pois os dados históricos de encaminhamentos realizados estão com todas as classes definidas.

2. Métodos

Todos os métodos e técnicas utilizados neste artigo estão disponíveis na ferramenta WEKA [Bouckaert et al 2018] em sua versão com interface gráfica ao qual é desenvolvida em java, ou seja, multiplataforma.

2.1. Algoritmos

Os algoritmos escolhidos para este artigo são todos os modelos de árvores de decisão que estão disponíveis nativamente na ferramenta WEKA [Bouckaert et al 2018], e que podem ser utilizados com os tipos de dados utilizados neste artigo, são eles:

- **DecisionStump** [Iba and Langley 1992] é um modelo que consiste em gerar uma árvore de decisão de um único nível utilizando entropia. É um algoritmo classificador de aprendizado fraco, que oferece uma baixa taxa de predição, porém como há somente um nível, ele é muito eficiente. Devido a ele ter somente um nível, ele tende a ficar altamente correlacionado a somente um atributo.
- **HoeffdingTree** [Hulten et al 2001] é um modelo que tem uma característica exclusiva e teoricamente atraente em relação a outros modelos de árvore de decisão, que é o fato de ter garantias sólidas de desempenho. Elas partem do princípio de que, geralmente, uma amostra pequena de dados pode ser suficiente para escolher os atributos de divisão da árvore ideal.
- **C4.5 (no WEKA se chama J48)** [Quinlan 1993] é o modelo mais usado nos dias atuais, ele também usa de entropia da informação para geração do modelo, onde em cada nó o algoritmo escolhe um atributo que subdivide mais eficientemente o conjunto das amostras em subconjuntos homogêneos e caracterizados por sua classe, sendo que o critério para escolha do atributo na

subdivisão é o ganho de informação [Karegowda et al 2010] obtido em cada nó da árvore.

- **LMT** [Landwehr et al 2005] [Sumner et al 2005] quer dizer “logistic model trees” ou modelo de árvores logísticas. São árvores de classificação com funções de regressão logística.
- **RandomForest** [Breiman 2001] RandomForest ou floresta aleatória, é um classificador que consiste em uma coleção de classificadores estruturados em árvore. O algoritmo cria várias árvores de decisão e as combina para obter uma predição mais estável e com maior acurácia. Com isso, ele tente a ser muito pouco eficiente na geração do modelo, mas altamente eficaz.
- **RandomTree** [Waikato 1999-2017] é um modelo que constrói uma árvore onde se considera K atributos escolhidos aleatoriamente em cada nó, e não é realizada a sua poda.
- **REPTree** [Srinivasan 2014] REPTree quer dizer “reduced error pruning tree” ou poda árvore com erro reduzido. É um modelo rápido que constrói uma árvore de decisão [Breiman et al 1984] usando variância ou ganho de informação [Karegowda et al 2010] como critério de divisão, onde a poda a árvore é feita usando a poda de erro reduzido.

2.2. Dados

Para o treinamento dos modelos de árvores de decisão [Breiman et al 1984] e a avaliação dos mesmos, foram utilizados os dados de encaminhamentos do ano de 2018, por conter dados mais recentes.

Quase todos os dados são numéricos, mas devido a natureza deles, podemos classificá-los como categóricos ou nominais, pois não representam valores numéricos.

Os dados estavam em um banco de dados relacional, então foi feita uma consulta em SQL, e posteriormente salvos em um arquivo CSV.

O WEKA [Bouckaert et al 2018] trabalha nativamente com o formato ARFF (Attribute-Relation File Format), porém é possível a importação de arquivos CSV sem muitos problemas.

A vantagem da utilização do formato ARFF [Bouckaert et al 2018] neste caso é que os tipos de dados ficam salvos no arquivo, então não precisamos alterar os tipos de dados todas as vezes que é carregado o arquivo na ferramenta, pois quando o arquivo é importado diretamente de um arquivo CSV, ela reconhece os dados como sendo dados numéricos.

Então foi carregado o arquivo CSV, alterados os tipos de dados dos atributos, e posteriormente o arquivo foi salvo em formato ARFF [Bouckaert et al 2018] para ser utilizado na ferramenta.

Os atributos foram inicialmente selecionados de acordo com o conhecimento da área de negócio, foram escolhidos os atributos que mais podem ter impacto na geração dos modelos.

Segue abaixo a tabela com o dicionário dos dados escolhidos.

Tabela 1. Dicionário de dados

Atributo	*Característica*
flTipoProcesso	Indica o tipo de processo, se é protocolo ou processo físico.
nuTramite	Número sequencial do encaminhamento.
cdAssunto	Assunto do qual se trata o processo (intimação, férias, reclamação, concurso, declaração, elogio, indenização, etc.).
cdTipoParecer	Tipo do parecer (despacho, encaminhamento, decisão ou arquivamento).
cdSetorResp	Código do setor responsável pelo processo.
cdSetorAbertura	Código do setor de abertura do processo.
cdSetorAutuacao	Código do setor de autuação do processo.
cdTipoProcesso	Tipo do processo (administrativo, plano geral de autuação, sindicância, disciplinar ou jurídico).
cdUsuario	Código do usuário que realizou o encaminhamento.
cdSetorOrigem	Código do setor onde o processo estava antes de ser encaminhado.
cdSetorDestino	Código do setor onde o processo foi encaminhado (classe).

Como a quantidade de atributos é proporcional ao tempo de treinamento do modelo, ou seja, quanto mais atributos mais tempo é necessário para treinar o modelo, e como alguns atributos podem ser pouco impactantes ou mesmo irrelevantes para o problema, foi então utilizado o método de seleção de atributos conhecido como ganho de informação (Gain ratio) [Karegowda et al 2010], que faz avaliação do valor de um atributo medindo a sua taxa de ganho em relação a cada classe, lhe atribuindo uma nota (mérito) para cada atributo.

Foram então ranqueados os atributos na seguinte ordem: cdSetorOrigem, cdSetorAbertura, cdSetorResp, cdUsuario, cdAssunto, cdSetorAutuacao, cdTipoParecer, flTipoProcesso, nuTramite, cdTipoProcesso.

Para avaliar quantos atributos deveriam ser retirados para a construção dos modelos, foi removido cada atributo na ordem inversa, gerando os modelos de cada algoritmo, e avaliando os resultados para que não perdessem sua eficácia e melhorassem a sua eficiência.

Abaixo segue os dados compilados em gráficos para uma melhor visualização da eficácia e eficiência dos modelos escolhidos.

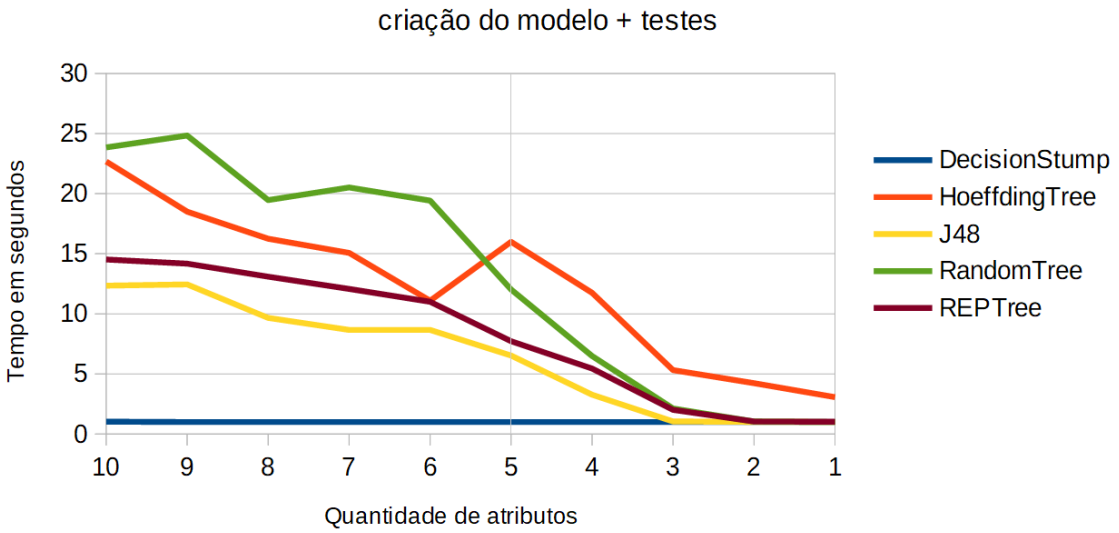

Imagem 1. Gráfico de eficiência do modelo.

No gráfico de eficiência, fica claro que basicamente, quanto menos atributos, mais eficientes são os modelos, com exceção do HoeffdingTree [Hulten et al 2001] que teve um aumento considerável com 5 atributos.

Podemos ter uma influência externa nestes resultados, pois executando os algoritmos mais de uma vez, teremos resultados diferentes, portanto temos que olhar pra este gráfico de forma geral. Mas, o importante é notar que a menor quantidade de atributos esta relacionada a maior eficiência.

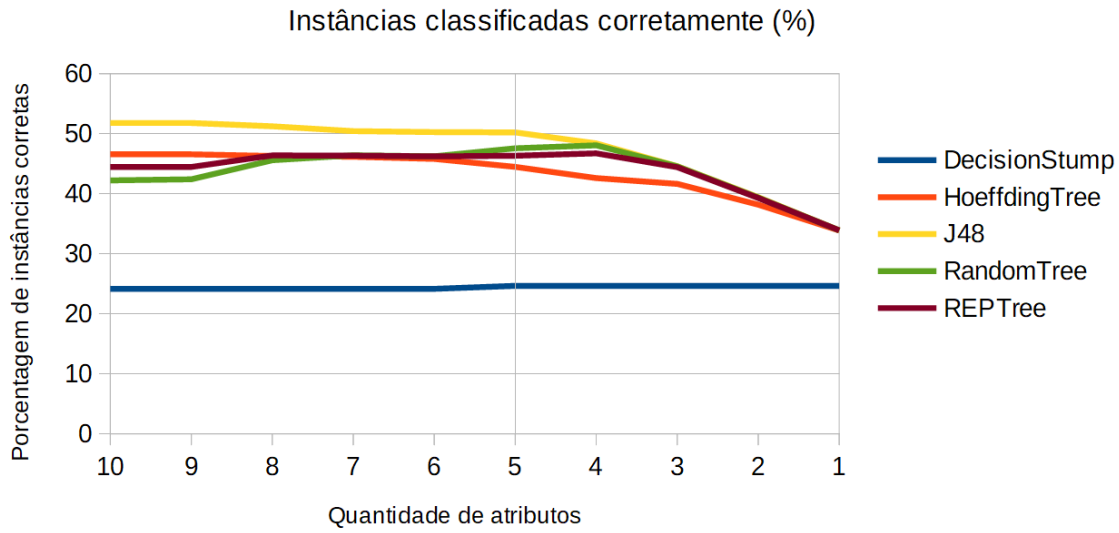

Imagem 2. Gráfico de eficácia do modelos.

Até a linha de 5 atributos nota-se que a eficácia permanece praticamente constante, e após isso, com 4 atributos uma leve queda, seguida a partir de 3 atributos por uma queda mais acentuada da eficácia dos modelos.

Interessante notar, que os algorítimos RandomTree [Waikato 1999-2017] e REPTree [Srinivasan 2014], tiveram um leve aumento da eficácia, mesmo com uma menor quantidade de atributos.

O DecisionStump [Iba and Langley 1992] praticamente ficou estabilizado, pois

como diz na descrição do algoritmo, ele cria um modelo de um único nível, assim podemos dizer que ele é altamente correlacionado ao atributo cdSetorOrigem, que foi o atributo melhor ranqueado.

Como a eficácia é mais importante neste caso, foi definido que a linha de corte dos atributos seja de 5 atributos.

2.3. Testes

Para a realização dos testes e obtenção dos resultados, foi utilizado o método de testes chamado de cross-validation (validação cruzada) [Kohavi 2008] com o método k-fold. Este é um método estatístico onde os dados são subdivididos aleatoriamente em partes iguais (ou aproximadamente iguais). Com este método, é possível inferir de forma quantitativa a capacidade de generalização do modelo.

Para este estudo foi escolhido um número de 10 k-folds, que é o número padrão na ferramenta WEKA[Bouckaert et al 2018] e o mais comumente usado. Sendo assim, os dados são divididos em 10 partes, onde enquanto uma parte é usada para testes as outras 9 serão usadas para treinamento, posteriormente é usada outra parte para testes e assim sucessivamente até todas as partes sejam usadas como testes.

Este método obtêm resultados mais homogêneos e precisos, pois todos os registros acabam sendo testados.

3. Resultados

Os resultados foram obtidos, executando todos os algoritmos da ferramenta, com os mesmos conjuntos de dados, com os parâmetros padrões e tudo sendo executado no mesmo computador, para que não houvesse interferência externa e os resultados fossem consistentes.

Os resultados obtidos que são retornados da ferramenta, além das estatísticas mostradas na tabela 2, são:

- **Kappa statistic (Estatística kappa)** [Kumar and Sahoo 2012] pode ser definida como grau de concordância entre dois conjuntos de dados. Neste caso o grau é medido entre os avaliadores e a classificação esperada. O resultado do Kappa varia entre 0 e 1, e quanto maior for o valor de Kappa, mais forte é a concordância.
- **Mean absolute error (Erro absoluto médio)** [Srivastava 2014] é a média de diferença entre as predições e a realidade ou a soma de erros absolutos divididos pelo número de predições. Neste caso, quanto menor for o valor, melhor é a acurácia do modelo.
- **Root mean squared error (Erro quadrático médio da raiz)** [Kumar and Sahoo 2012] é o desvio padrão dos resíduos ou erros de previsão. Esses resíduos são a medida da distância entre os pontos de dados e da linha de regressão. Portanto um valor menor, define que os pontos estão mais pertos desta linha, então, quanto menor for o valor, melhor é a acurácia do modelo.
- **Relative absolute error (Erro absoluto relativo)** [Srivastava 2014] como o próprio nome diz, é o erro total relativo. Valores menores indicam uma melhor acurácia do modelo.
- **Root relative squared error (Erro quadrático relativo à raiz) [Srivastava**

2014] é o quadrado do erro total absoluto. Valores menores indicam uma melhor acurácia do modelo.

Há também outros parâmetros disponíveis, como TP rate (taxa de verdadeiros positivos), FP rate (taxa de falsos positivos), precision (precisão) etc. Porém tais parâmetros são relacionados a classe, e como há muitas classes nestes dados, 116 ao todo, fica inviável exibir estes resultados aqui e analisá-los.

Tabela 2. Comparação de algoritmos para seleção de atributos.

Algoritmo	***Tempo de criação do modelo (em segundos)***		***Tempo de testes (em segundos)***		***Instâncias classificadas corretamente (%)***	
	10 atributos	***5 atributos***	***10 atributos***	***5 atributos***	***10 atributos***	***5 atributos***
DecisionStump	0,03	0,00	1	1	24,1454	24,6392
HoeffdingTree	1,67	0,99	21	15	46,5600	44,4601
J48	1,34	0,53	11	6	51,7743	50,2116
LMT	-	-	-	-	-	-
RandomForest	-	-	-	-	-	-
RandomTree	1,84	1,02	22	11	42,2084	47,5583
REPTree	1,52	0,72	13	7	44,4547	46,3158

Percebe-se que os algoritmos LMT [Landwehr et al 2005] [Sumner et al 2005] e RandomForest [Breiman 2001] não obtiveram resultados, pois a memória do computador utilizado (16 gb) não suportou estes modelos, mesmo com a quantidade de atributos reduzida, portanto eles serão desconsiderdos para esta análise porque os tempos de criação do modelo e de testes são muito importantes neste caso.

Alguns modelos de árvore de decisão [Breiman et al 1984] não se adaptam muito bem com conjuntos de dados muito grandes, com um número alto de atributos e nem com uma alta quantidade de classes [Katuwal and Suganthan 2018].

Sendo assim, como o conjunto de dados contêm muitas classes, 116 ao todo, o motivo de os algoritmos LMT e RandomForest não apresentarem resultados é devido à alta quantidade de classes.

Tabela 3. Resultados.

Algoritmo	***Kappa statistic***	***Mean absolute error***	***Root mean squared error***	***Relative absolute error***	***Root relative squared error***
DecisionStump	0,0999	0,0153	0,0875	96,4354	98,2285
HoeffdingTree	0,3779	0,0109	0,0852	68,7122	95,6230
J48	0,4506	0,0106	0,0759	66,7952	85,1898
LMT	-	-	-	-	-
RandomForest	-	-	-	-	-
RandomTree	0,4233	0,0101	0,0801	63,5345	89,8399

REPTree	0,4073	0,0110	0,0783	69,4814	87,8232

Conforme a explicação dada sobre os resultados e as tabelas 2 e 3, verifica-se que o algoritmo J48 [Quinlan 1993] é o melhor em quase todos os quesitos, só perde por uma margem pequena para o RandomTree [Waikato 1999-2017] nos parâmetros mean absolute error [Srivastava 2014] e no relative absolute error [Srivastava 2014].

Como, neste caso, os parâmetros mais importantes são as instâncias classificadas corretamente que se define como a eficácia do modelo, e o tempo de criação do modelo e testes que se define como eficiência do modelo, temos o J48 [Quinlan 1993] em vantagem, perdendo somente para o DecisionStump [Iba and Langley 1992] no tempo, entretanto o DecisionStump [Iba and Langley 1992] tem uma eficácia muito baixa de apenas 24,6392% contra 50,2116% J48 [Quinlan 1993].

4. Conclusões

Como o intuito deste artigo era definir o algoritmo mais eficiente e eficaz para o problema de encaminhamento de processos, tem-se o algoritmo J48 [Quinlan 1993] como a melhor escolha.

Embora os resultados não se mostrem satisfatórios no quesito eficácia, pois com aproximadamente 50% de instâncias classificadas corretamente é um resultado muito aquém do esperado, tem-se uma eficiência muito boa e pode-se dizer que, com um trabalho melhor na etapa de pré-processamento, como, o tratamento do desbalanceamento das classes, dos campos em branco, dentre outras, pode-se obter um resultado muito melhor do que os exibidos neste artigo.

5. Referências

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) “Classification and regression trees” Wadsworth International Group, Belmont, CA.

Caruana, R. and Niculescu-Mizil, A. (2006) “An empirical comparison of supervised learning algorithms”, In: Proceedings of the twenty-third international conference on machine learning (ICML’06), Edited by William Cohen & Aandrew Moore, ACM New York, NY, USA, p. 161-168.

Karegowda, A.G., Manjunath, A.S. and Jayaram, M.A. (2010) “Comparative study of attribute selection using gain ratio and correlation based feature selection”, In: International Journal of Information Technology and Knowledge Management (IJITKM), Volume 2, No. 2, p. 271-277.

Bouckaert, R. R., Frank, E., Hall M., Kirkby R., Reutemann P., Seewald, A. and Scuse D. (2018) “WEKA Manual for Version 3-8-3”, In: The University of Waikato, September.

Iba, W. and Langley, P. (1992) “Induction of One – Level Decision Trees” In: Proceedings of the Ninth International Conference on Machine Learning.

Quinlan, J. R. (1993) “C4.5: Programs for machine learning”, Morgan Kaufmann Publishers Inc, San Mateo, CA, USA.

Hulten G., Spencer L. and Domingos P. (2001) “Mining time-changing data streams.” In: ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, p. 97-106.

The University of Waikato, (1999-2017) “WEKA Documentation”,

http://weka.sourceforge.net/doc.dev/

Srinivasan, B. and Mekala, P. (2014) “Mining Social Networking Data for Classification Using REPTree”, International Journal of Advance Research in Computer Science and Management Studies, Volume 2, p. 155-160.

Landwehr, N., Hall, M. and Frank, E. (2005) “Logistic Model Trees. Machine Learning”, p. 161-205.

Sumner, M., Frank, E. and Hall, M. (2005) “Speeding up Logistic Model Tree Induction”, In: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, p. 675-683.

Breiman, L. (2001) “Random Forests. Machine Learning.” p5-32.

Kohavi, R. (2008) “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection” In: International Joint Conference on Artificial Intelligence (IJCAI), Stanford University, Stanford, CA, USA.

Kumar, Y. and Sahoo, G. (2012) “Analysis of Parametric & Non Parametric Classifiers for Classification Technique using WEKA”, In: I.J. Information Technology and Computer Science, p. 43-49.

Srivastava, S. (2014) “Weka: A Tool for Data preprocessing, Classification, Ensemble, Clustering and Association Rule Mining”, In: International Journal of Computer Applications, Volume 88, Nº 10.

Katuwal, R. and Suganthan, P.N. (2018) “Enhancing Multi-Class Classification of RandomForest using Random Vector Functional NeuralNetwork and Oblique Decision Surfaces”, School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.