UNIVERSIDADE FEDERAL DE PERNAMBUCO

GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CENTRO DE INFORMÁTICA

Relatório do Projeto de Aprendizagem de Máquina

"A Comprehensive Study on Ensemble-Based Imbalanced Data Classification Methods for Bankruptcy Data"

GABRIELA LEAL MAGALHÃES

ELÁDIA CRISTINA CORRÊA

RECIFE

2021

SUMÁRIO

1. Introdução

Dados reais são comumente desbalanceados. Por exemplo, existem muito mais empresas que não declararam falência do que as que declararam. Numa classificação binária como essa, quando há poucos dados de uma classe e muitos dados da outra, o classificador torna-se enviesado para a classe majoritária.

Na indústria financeira, predição de falência continua sendo um tópico em destaque. Realizando a predição, os *stakeholders* podem então analisar melhor o status da empresa e agir de acordo. Geralmente, os métodos que são utilizados para solucionar problemas de desbalanceamento são agrupados em métodos de: amostragem, aprendizagem ativa e sensível a custo.

Em [1], são utilizados métodos de *ensemble* baseados em *bagging* e *boosting*, aplicados em uma base de dados desbalanceada de falência de empresas. Mais especificamente, as abordagens comparadas são *SMOTEBAGGING*, *UNDERBAGGING*, *SMOTEBOOST* e *RUSBOOST*.

Esse relatório tem como objetivo descrever o processo que seguido para replicar o trabalho realizado em [1]: predição de falência de empresas. Quando não foi possível reproduzir similarmente, indicamos o motivo e explicamos o novo procedimento. Na seção 2, aprofundamos os conceitos de bagging, boosting, undersampling, oversampling e descrevemos o funcionamento de SMOTEBAGGING, UNDERBAGGING, SMOTEBOOST e RUSBOOST. Na seção 3, relatamos a metodologia seguida: linguagem, bibliotecas, utilização da base de dados e dos algoritmos. Na seção 4, apresentamos os resultados obtidos. Por fim, a seção 5 conclui este relatório.

2. Abordagens para dados desbalanceados

Para solucionar o problema de dados desbalanceados em um conjunto de treinamento a abordagem mais direta, a partir da perspectiva do tratamento dos dados, é calibrar a quantidade de instâncias por classe de forma que todas as classes passem a ter uma proporção desejada. Isso pode ser alcançado aumentando a quantidade de dados na classe minoritária ou removendo dados da classe predominante, ou inclusive realizando ambos procedimentos.

A primeira estratégia é nomeada *oversampling* e uma das técnicas mais comuns associadas a ela é SMOTE *(Synthetic Minority Oversampling Technique)*. Diferente de técnicas de *oversampling* que duplicam dados, SMOTE gera novos dados artificiais, trabalhando com espaço de feature no lugar de espaço de dados [5]. Sobre a classe minoritária é aplicada a interpolação dos dados de duas ou mais instâncias mapeadas através do algoritmo k-NN [6].

A segunda estratégia é dita *under-sampling* e seu método mais utilizado é o *Random Under Sampling*. Este método realiza a sub-amostragem da classe majoritária eliminando aleatoriamente instâncias [6]. A desvantagem dessa abordagem é a perda de informações, o que pode causar a classificação errada de instâncias e prejudicar a taxa de aprendizagem do modelo. Apesar da sua simplicidade, esse método mostrou-se superior em relação a outros similares, e por este motivo é o mais empregado [7].

Uma outra perspectiva é a nível de algoritmo, em que um método cujo desempenho sobre bases desbalanceadas vem sendo bastante estudado na literatura, o *ensemble*. *Ensemble* combina a decisão de múltiplos classificadores para determinar o rótulo de uma instância. Duas das abordagens mais aplicadas são *bagging* e *boosting*.

Bagging corresponde a abordagem de se treinar múltiplos classificadores em paralelo e retornar como resultado a média das respostas emitidas por eles. Esse método é aplicado em classificadores cujos modelos resultam em alta variância e baixo viés [7]. O objetivo, então, é diminuir a variância mantendo o baixo viés. Para o seu bom desempenho, é importante que os modelos não sejam correlacionados, o que se pode alcançar através do *bootstrap* de amostras do conjunto original de dados [7].

Já o método de *boosting* combina múltiplos classificadores de maneira sequencial. Ao final da execução de cada classificador, são aumentados os pesos das instâncias classificadas erradas e diminuídos os pesos das instâncias corretamente classificadas [7]. São também atribuídos pesos a cada classificador de acordo com o seu desempenho. O sucesso desse método está relacionado à diversidade dos classificadores ao receberem distribuições diferentes dos dados [7].

Contudo, ambas abordagens não são suficientemente eficientes sobre dados desbalanceados. Para este cenário é preciso incrementar os algoritmos de *bagging* e *boosting* com algoritmos de amostragem como o SMOTE e o *Random Under Sampling*. Nas subseções a seguir estão descritos os algoritmos que combinam SMOTE e *Random Under Sampling* com *bagging* e *boosting*, são eles SMOTEBAGGING, UNDERBAGGING, SMOTEBOOST e RUSBOOST.

2.1 SMOTEBAGGING

De forma sucinta, o algoritmo SMOTEBAGGING [9] é uma combinação de SMOTE com a técnica de *bagging*. Apesar desta técnica obter melhores resultados do que um único classificador sobre *datasets* desbalanceados, o problema da predominância das classes majoritárias permanece. É inevitável que na amostragem de dados para a construção dos modelos no *ensemble* alguns dos conjuntos de treinamento possuam percentuais desproporcionais de instâncias de cada classe ou até não venha a receber dados de uma classe minoritária.

Por isso, no SMOTEBAGGING é performado *oversampling* a cada iteração do algoritmo de *bagging* através da aplicação de SMOTE, visando corrigir o desbalanceamento dos dados recebidos por cada classificador presente no *ensemble*. O algoritmo de SMOTE é responsável por gerar novos dados sintéticos durante a construção de cada subconjunto. A quantidade de dados a serem selecionados de cada classe para a construção do subconjunto é ajustada através de um percentual *b*, o qual auxilia também na definição de quantos novos dados devem ser gerados para cada classe minoritária através do SMOTE.

O algoritmo de SMOTEBAGGING define também que cada classificador no *ensemble* deve possuir um valor distinto para *b* de modo a garantir a diversidade dos dados na construção de cada modelo. Além disso, o uso do percentual *b* permite que o SMOTEBAGGING opere tanto em casos de *datasets* desbalanceados binários como multiclasse.

2.2 UNDERBAGGING

O UNDERBAGGING [9] incorpora também a técnica de *bagging* assim como no SMOTEBAGGING, porém com a diferença de que é usada a técnica de *Random Under Sampling* ao invés de SMOTE, focando na sub-amostragem da classe majoritária.

Neste algoritmo, para cada iteração de *bagging*, ou seja, para a construção de cada modelo do ensemble, é dado como entrada um subconjunto de dados advindo da sub-amostragem aleatória da classe predominante e contendo também as demais classes.

Para a realização da sub-amostragem, é introduzido um coeficiente α que corresponde ao quanto da classe alvo deve ser replicado. Para as outras classes a quantidade de instâncias a serem replicadas será a proporção desta classe em relação a classe alvo, isso multiplicado pelo coeficiente α. Então cada classificador é executado e a classe mais votada é a selecionada para o rótulo da instância em questão. Este procedimento vale tanto para a etapa de treinamento quanto para a de teste.

Através dessa implementação o algoritmo, assim como SMOTEBAGGING, é capaz de balancear não apenas *datasets* binários como também problemas multiclasse. Uma outra vantagem deste método é a facilidade no controle da diversidade de dados através do coeficiente α.

2.3 SMOTEBOOST

Assim como nos algoritmos anteriores, o algoritmo SMOTEBOOST é uma combinação de SMOTE desta vez com o conceito de *boosting*. O objetivo dessa implementação é melhorar a acurácia do modelo de uma forma global ao focar na classe minoritária enquanto evita causar grandes danos à acurácia da predição da classe majoritária. Como consequência, o propósito deste algoritmo diz respeito ao aumento de instâncias verdadeiramente positivas classificadas corretamente.

Apesar dos algoritmos de *boosting* serem eficientes em reduzir a variância e o viés entre as etapas de treinamento e teste, como eles selecionam amostras a partir do conjunto total de dados, é inevitável que a classe majoritária prevaleça dentro dessas amostras. Desta forma, esses algoritmos são capazes de modelar corretamente a classe predominante a partir dos ajustes de pesos, porém, o mesmo não acontece com a classe minoritária.

Portanto, a abordagem do SMOTEBOOST construída em [8] incrementa o algoritmo de *boosting* Adaboost com SMOTE. A cada iteração do Adaboost é executado SMOTE de modo a se gerar novos dados sintéticos apenas para a classe minoritária. Os pesos do Adaboost são ajustados a partir de uma amostragem sobre a inclusão dos novos dados à base original. Ao fim da iteração os dados sintéticos são descartados e em seguida o procedimento se repetirá até que os pesos convergem. Assim, a inserção de SMOTE ao algoritmo de boosting garante o balanceamento das classes, solucionando o problema anterior e consequentemente trazendo melhora para a acurácia do modelo de ensemble.

2.4 RUSBOOST

O algoritmo RUSBOOST [4] possui a mesma proposta que o SMOTEBOOST: melhorar a acurácia global do modelo de ensemble para bases de dados desbalanceadas ao focar, neste caso, no tratamento da classe majoritária. Diferente do SMOTEBOOST, o RUSBOOST agrega a um algoritmo de *boosting* a técnica de *Random Under Sampling*.

Essa abordagem visa também atenuar a complexidade de tempo associada ao uso de SMOTE, a qual costuma ser mais elevada devido ao aumento na quantidade de dados de treinamento. Como *Random Under Sampling* remove aleatoriamente dados da classe majoritária, o tempo para construção do modelo tende a diminuir pois a execução é dada sobre uma base de treinamento mais concisa. A remoção de dados implica também na perda de informações, o que vem a ser uma desvantagem em usar *Random Under Sampling*. Todavia, no caso do RUSBOOST, não há um impacto tão negativo por tratar-se de um algoritmo que implementa *boosting*, isto é, caso um dado tenha sido removido na construção de um modelo, ele estará presente no treinamento de outro modelo.

O algoritmo RUSBOOST é implementado de acordo com o seguinte: para cada iteração do algoritmo Adaboost, é criado um novo *dataset* onde são removidos aleatoriamente da classe majoritária tantos quanto forem os dados necessários para que ela tenha o mesmo tamanho da classe minoritária; então, é executado um classificador pouco complexo e os pesos do Adaboost são atualizados.

3. Metodologia

Como mencionado anteriormente, este trabalho seguiu a metodologia adotada em [1] a fim de replicar seus resultados. Na figura 1 é mostrado o fluxograma da implementação deste trabalho.

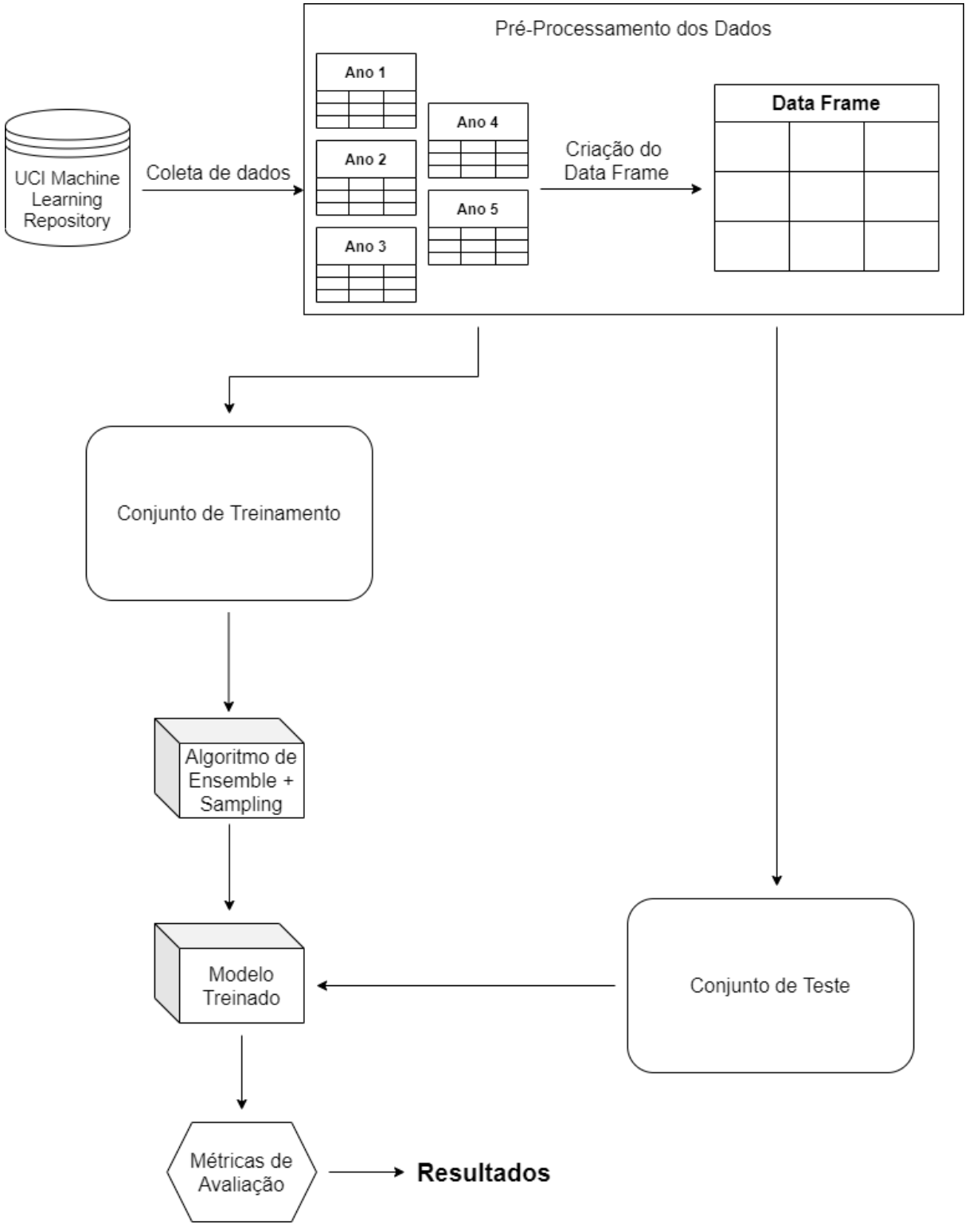

Figura 1. Fluxograma dos experimentos.

Apesar de em [1] os experimentos terem sido implementados em R, fazendo uso da biblioteca EBMC, este trabalho foi implementado em *Python*, no *Google Colab*. O pré-processamento dos dados foi realizado com o auxílio da biblioteca *Pandas* e para a análise dos algoritmos *SMOTEBAGGING*, *UNDERBAGGING* e *RUSBOOST* e das métricas de avaliação *recall*, precisão, acurácia, *f-measure* e ROC foram utilizadas as implementações presentes nas bibliotecas *imbalanced-learn* e *scikit-learn* respectivamente. Contudo, o algoritmo *SMOTEBOOST* não encontra-se ainda disponível nessas bibliotecas. Sendo assim, foi importada a implementação deste algoritmo desenvolvida pelo laboratório Data, Inference, Analytics, and Learning (DIAL) da Universidade de Notre Dame, disponível em [3]. Foram necessários alguns ajustes de importação de dependências no código original dele para compatibilizar com a versão mais atual do *scikit-learn*, que é a utilizada pelo *imbalanced-learn*.

3.1 Base de Dados

Em [1], é informado que a base de dados utilizada para a realização dos experimentos foi obtida no repositório *Kaggle*. No acesso a este, descobriu-se que esses dados foram coletados para a tarefa de prever especificamente a falência de empresas polonesas. Utilizando essa informação sobre o país de origem dos dados, foi possível encontrar uma visão mais completa sobre os mesmos no repositório de aprendizagem de máquina UCI [2].

A base de dados utilizada neste trabalho é composta por cinco arquivos cada um referente a um ano de dados coletados. Foi feita, então, uma concatenação dos arquivos CSVs, os quais foram importados como um *Pandas DataFrame*. As linhas que possuíam dados faltantes foram removidas, para em seguida ser feita uma normalização dos dados para o intervalo (-1, 1) e um *shuffle* das linhas com *random_state=42*. Removeu-se também linhas duplicadas. Após esses procedimentos, foi obtido um *DataFrame* composto por 19449 registros, sendo 19027 de “não falência”, 422 de “falência” e a taxa de desbalanceamento dessa base de dados aproximadamente 0.022179008777.

3.2 Experimentos

Foram realizados neste trabalho quatro experimentos com abordagens híbridas de ensemble para o problema de datasets desbalanceados. Os algoritmos utilizados foram o SMOTEBAGGING, UNDERBAGGING, SMOTEBOOST e RUSBOOST.

Seguindo a metodologia de [1], aplicou-se sobre a base de dados a técnica de *hold-out* para separar os conjuntos de treinamento e teste, sendo o primeiro composto por 70% da base e o segundo constituído dos 30% restantes de dados.

- SMOTEBAGGING: Para a realização deste experimento foi utilizado o classificador *ensemble* “BalancedBaggingClassifier” disponibilizado pela biblioteca *imbalanced-learn* passando como hiperparâmetro “sampler” uma chamada à implementação do SMOTE disponível na mesma biblioteca.
- UNDERBAGGING: Neste experimento, assim como no anterior, foi utilizado o classificador *ensemble* “BalancedBaggingClassifier” implementado pela biblioteca *imbalanced-learn*, porém, neste caso foi passado ao hiperparâmetro “sampler” a chamada ao *Random Under Sampler*, implementado também por essa biblioteca.
- SMOTEBOOST: Como mencionado anteriormente, para a execução do experimento com SMOTEBOOST foi utilizada a implementação do algoritmo em [3] em sua forma *default*.
- RUSBOOST: Para o experimento com o algoritmo RUSBOOST foi utilizado o classificador “RUSBoostClassifier” disponibilizado pela biblioteca *imbalanced-learn* em sua forma *default*.

3.3 Avaliação

Após a execução dos algoritmos acima, foram computadas as métricas de avaliação *recall*, precisão, acurácia, *f-measure* e ROC. Com esta última foi possível gerar o gráfico com a curva ROC.

As métricas de *recall* e precisão são calculadas com base na quantidade de *True Positives, False Positives, True Negatives* e *False Negatives.* Na tabela 1 abaixo é possível visualizar o que cada um desses valores representa. Já o f-measure consiste na média harmônica dessas duas métricas.

	Instâncias Positivas	Instâncias Negativas
Dados Classificados Positivos	True Positive	False Positive
Dados Classificados Negativos	False Negative	True Negative

Tabela 1. Relação entre rótulos reais e preditos.

- Recall = True Positive / (True Positive + False Negative)
- Precisão = True Positive / (True Positive + False Positive)

A pontuação ROC também é calculada sobre os valores mencionados na tabela acima e com ela é possível gerar uma curva que mapeia a correlação entre a taxa de *True Positives* e *False Positives*.

A acurácia é a métrica mais popularizada para a avaliação de máquinas de aprendizagem. Com ela é calculado o percentual de dados classificados corretamente.

Apesar de ter sido realizado o balanceamento das classes no conjunto de treinamento, o conjunto de teste não foi alterado e permaneceu desbalanceado para que representasse a distribuição real dos dados. Por este motivo, todas as métricas com a exceção da acurácia tiveram suas médias calculadas através da função "*weighted"*.

4. Resultados

Nesta seção são apresentados os resultados dos experimentos implementados neste trabalho. Abaixo na tabela 2 estão dispostos os valores associados às métricas de avaliação de desempenho de cada algoritmo.

Algoritmo	*Recall*	Acurácia	Precisão	*F-Measure*
SMOTEBAGGING	0.973	0.973	0.968	0.970
UNDERBAGGING	0.865	0.865	0.973	0.909
SMOTEBOOST	0.972	0.972	0.968	0.970
RUSBOOST	0.829	0.829	0.971	0.888

Tabela 2. Resultados das métricas de avaliação.

Primeiramente é preciso considerar que na tarefa de classificar dados desbalanceados, devido à predominância da classe majoritária, o modelo tende a aprender melhor a disposição dos dados dessa classe. Logo, o desafio é conseguir classificar corretamente o maior número de instâncias da classe minoritária, o que pode ser medido através das métricas de *recall* e precisão.

De uma forma geral é possível notar que a técnica SMOTE, independente do método de *ensemble* ao qual foi acoplada, resultou nos melhores valores das métricas avaliadas. Ambos o SMOTEBAGGING e o SMOTEBOOST obtiveram altos valores de *recall* e precisão, o que significa que foram extremamente eficientes em classificar corretamente os dados positivos.

Em contrapartida, a aplicação de *Random Under Sampling* aos métodos de *ensemble* mostrou resultados inferiores nas métricas de *recall*, acurácia e *f-measure*, quando comparada a aplicação de SMOTE aos mesmo métodos. Apenas na métrica de precisão os algoritmos com essa abordagem superaram os algoritmos que implementam SMOTE. O alto valor da precisão combinado a valores inferiores das outras métricas, aponta que o UNDERBAGGING e o RUSBOOST classificaram como positivas apenas instâncias verdadeiramente positivas, entretanto, das instâncias verdadeiramente positivas, muitas foram classificadas erroneamente como negativas.

Comparando os métodos de *ensemble bagging* e *boosting*, não há um intervalo suficientemente grande entre os valores das métricas associados a eles, podendo-se então concluir que para esta base de dados eles têm desempenhos praticamente equivalentes.

Apenas no caso da implementação desses métodos com a sub-amostragem do *Random Under Sampling*, o método de *bagging* mostrou-se levemente superior ao método de *boosting*, visto que o UNDERBAGGING obteve valores de até 0.04 pontos acima do RUSBOOST.

É interessante também analisar as curvas ROC (mostradas nas figuras abaixo) construídas de acordo com a classificação positiva das instâncias em cada um dos algoritmos executados. Mais uma vez a maior diferença se encontra na comparação das técnicas de amostragem e pouco se nota de diferente ao comparar as curvas considerando os algoritmos de *ensemble*.

A partir das curvas ROC é possível visualizar que os algoritmos que implementam SMOTE acertaram mais instâncias verdadeiramente positivas ao mesmo tempo que poucas instâncias negativas foram falsamente classificadas por eles como positivas. Para os algoritmos com *Random Under Sampling*, apesar do número mais elevado de instâncias positivas classificadas corretamente, isso também levou esses algoritmos a classificarem equivocadamente instâncias negativas como positivas.

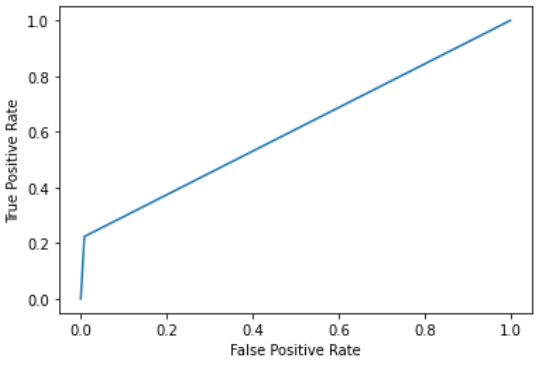

Figura 2. Curva ROC SMOTEBAGGING.

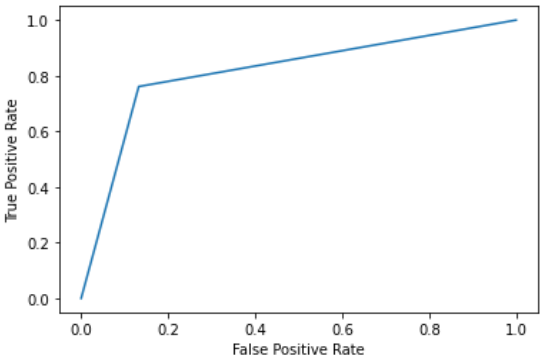

Figura 3. Curva ROC UNDERBAGGING.

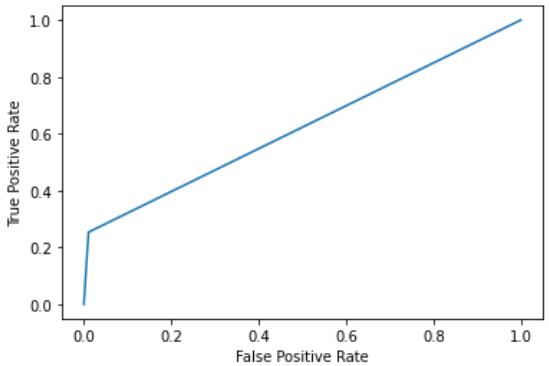

Figura 4. Curva ROC SMOTEBOOST.

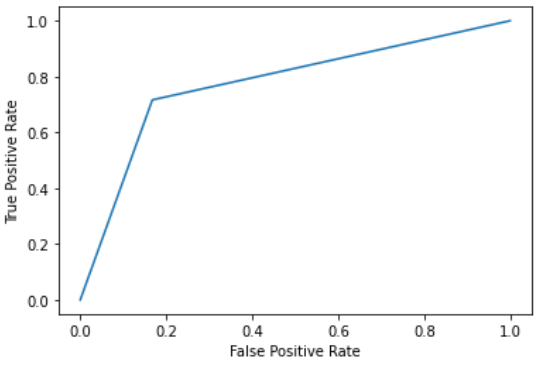

Figura 5. Curva ROC RUSBOOST.

5. Conclusão

Utilizar dados desbalanceados é uma tarefa desafiadora. Em [1], é afirmado que a taxa de desbalanceamento é de 0.02072063. Esse valor é muito próximo ao que obtivemos. Após aplicarmos *SMOTEBAGGING*, *UNDERBAGGING*, *SMOTEBOOST* e *RUSBOOST*, obtivemos resultados similares aos demonstrados em [1].

Recall, Precisão e *F-measure* são métricas importantes para dados desbalanceados, pois possuem um enfoque na taxa de instâncias verdadeiramente positivas classificadas corretamente. Para estas métricas, obtivemos valor alto com *SMOTEBOOST* e *SMOTEBAGGING*. Desta maneira, evidenciamos que *SMOTE* supera *Random Under Sampling*.

6. Referências Bibliográficas

[1] K. UlagaPriya and S. Pushpa, "A Comprehensive Study on Ensemble-Based Imbalanced Data Classification Methods for Bankruptcy Data," 2021 6th International Conference on Inventive Computation Technologies (ICICT), 2021, pp. 800-804, doi: 10.1109/ICICT50816.2021.9358744.

[2] UCI Machine Learning Repository: Polish companies bankruptcy data Data Set. Disponível em https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data. Acesso em: 27 de abril, 2021.

[3] Python-based implementations of algorithms for learning on imbalanced data. Disponível em https://github.com/dialnd/imbalanced-algorithms. Acesso em: 27 de abril, 2021.

[4] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse and A. Napolitano, "RUSBoost: Improving classification performance when training data is skewed," 2008 19th International Conference on Pattern Recognition, 2008, pp. 1-4, doi: 10.1109/ICPR.2008.4761297.

[5] A. Amin et al., "Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study," in IEEE Access, vol. 4, pp. 7940-7957, 2016, doi: 10.1109/ACCESS.2016.2619719.

[6] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 4, pp. 463-484, July 2012, doi: 10.1109/TSMCC.2011.2161285.

[7] S. Ahmed, A. Mahbub, F. Rayhan, R. Jani, S. Shatabda and D. M. Farid, "Hybrid Methods for Class Imbalance Learning Employing Bagging with Sampling Techniques," 2017 2nd International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), 2017, pp. 1-5, doi: 10.1109/CSITSS.2017.8447799.

[8] Chawla N.V., Lazarevic A., Hall L.O., Bowyer K.W, “SMOTEBoost: Improving Prediction of the Minority Class in Boosting”, n: Lavrač N., Gamberger D., Todorovski L., Blockeel H. (eds) Knowledge Discovery in Databases: PKDD 2003. PKDD 2003. Lecture Notes in

Computer Science, vol 2838. Springer, Berlin, Heidelberg, 2003, doi: 10.1007/978-3-540-39804-2_12.

[9] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," 2009 IEEE Symposium on Computational Intelligence and Data Mining, 2009, pp. 324-331, doi: 10.1109/CIDM.2009.4938667.