Aprendizado de Máquina para a Classificação da Qualidade de Vinhos

RAFAEL CARNEIRO DE CASTRO, Universidade Federal de Minas Gerais

Algoritmos de aprendizado de máquina estão ganhando muita popularidade devido à abrangência de aplicabilidade dos mesmos. Um importante campo de aplicação do aprendizado de máquina é na avaliação da qualidade de processos industriais e produtos. Este trabalho se utiliza de uma base de dados de um vinho português denominado Vinho Verde para se exercitar boas práticas de análise e visualização dos dados, tomada de decisões para a utilização de algoritmos de aprendizado de máquina, e discussão dos resultados obtidos, procurando expor de forma simples e sucinta como se trabalhar com bases de dados e obter resultados satisfatórios na classificação por algoritmos de aprendizado de máquina.

PALAVRAS-CHAVE:

Aprendizado de Máquina, Classificação, Análise de Dados, Correlação, Vinhos

Formato ACM de Referência:

Ben Trovato, G.K.M. Tobin, Lars Thørvӓld, Lawrence P. Leipuner, Sean Fogarty, Charles Palmer, John Smith, and Julius P. Kumquat. 1997. SIG Paper in word Format. *ACM J. Comput. Cult. Herit*. 9, 4, Article 39 (March 2010), 4 pages.

1 INTRODUÇÃO

Com o avanço acelerado da tecnologia e das possiblidades que a mesma abre, o mercado se torna cada vez mais dinâmico e exigente, e os processos industriais precisam cada vez mais buscar otimização e qualidade. Acompanhando as evoluções tecnológicas e o cenário favorável, algoritmos de aprendizado de máquina estão sendo cada vez mais utilizados nas mais diversas áreas, inclusive na otimização de processos industriais e avaliação de qualidade de produtos. Este trabalho propõe a utilização de algoritmos de aprendizado de máquina exercitados em uma base de dados que contém informações de produção de um vinho português denominado Vinho Verde. Os dados possuem características de produção como acidez, potencial Hidrogeniônico, densidade e quantidade de álcool, parâmetros estes que são utilizados para classificar a qualidade do produto. A base de dados está disponível em [1]. Este trabalho tem por objetivo, através da base mencionada, expor uma análise preliminar dos dados, extraindo características que possam ser relevantes para a utilização de algoritmos de aprendizado de máquina. Em seguida, os experimentos serão feitos, considerando diferentes formas de aprendizado e classificação, buscando se chegar a um ponto de classificação da qualidade de forma binária. Por fim, os resultados serão analisados. Também é objetivo deste trabalho chegar a resultados satisfatórios de forma didática e clara, criando uma referência para os primeiros passos na aplicação do conhecimento e teoria de aprendizado de máquina, obtido principalmente pela matéria ministrada no Departamento de Ciência da Computação da UFMG. A linguagem de programação utilizada para a análise e operação dos dados é o Python, por sua abrangência e quantidade de bibliotecas e recursos que facilitam o trabalho com aprendizado de máquina e exibição de dados.

2 METODOLOGIA

2.1 Os Dados

A base de dados utilizada neste trabalho, disponível em [1], possui os dados de produção do vinho português Vinho Verde. Estão disponíveis na referência os dados de produção tanto de vinhos tintos quanto de vinhos brancos. Como a base de vinhos brancos é maior, ela será utilizada aqui. As características de produção contidas nos dados são:

* Acidez Fixa
* Acidez Volátil
* Ácido Cítrico
* Açúcar Residual
* Cloretos
* Dióxido de Enxofre Livre
* Dióxido de Enxofre Total
* Densidade
* pH
* Sulfatos
* Álcool
* Qualidade

Todos os dados são rotulados pelo parâmetro Qualidade, que assume valores entre 0 e 10. Esta característica é importante para a utilização de algoritmos de aprendizado supervisionado para classificação, conforme o objetivo deste trabalho, classificando o vinho produzido como bom ou regular, pelos parâmetros de produção fornecidos. Como o valor dos rótulos dos dados varia entre 0 e 10, por uma sugestão dos fornecedores da base [2], um ajuste será feito, de forma que vinhos com o parâmetro de qualidade igual ou superior a 7 são considerados bons, e regulares caso contrário. Este limiar também é apontado pelos fornecedores da base de dados [2]. Utilizando a biblioteca Pandas para a operação de dados em Python, o código da Figura 1 foi feito com o intuito de categorizar os dados rotulados em vinhos bons ou regulares.



Figura 1 Código para a categorização dos dados em vinhos bons e regulares

Como se pode ver pela Figura 1, os dados são lidos a partir do arquivo *winequality-white.csv,* disponibilizado pelos fornecedores da base de dados [1]. Em seguida, através do valor da coluna *quality,* traduzida como qualidade, os vinhos são categorizados como bons (valor 1) ou regulares (valor 0). Em seguida a coluna *quality* é excluída do *data set,* já que não será mais utilizada na classificação; apenas a coluna *category* o será.

2.2 Balanceamento das Classes

Um importante ponto de início para a observação dos dados é exibir o balanceamento das classes, ou seja, com que frequência cada uma delas aparece nos dados estudados. No *data set* utilizado neste trabalho tem-se duas classes, uma para vinhos bons e outra para vinhos regulares. Para a observação dos dados neste sentido, o código da Figura 2 foi feito. Ele cria um gráfico através da contagem dos diferentes valores encontrados na coluna *category* na base de dados analisada.



Figura 2 Código para a geração do gráfico de balanceamento dos dados

O gráfico resultante do código da Figura 2 pode ser visto na Figura 3. Como se pode notar, existem 3838 exemplares classificados como 0 (regulares) e 1060 exemplares classificados como 1 (bons), totalizando 4898 vinhos. Como se pode notar, existem consideravelmente mais exemplares classificados como regulares do que como bons. Sendo assim a base de dados é desbalanceada.

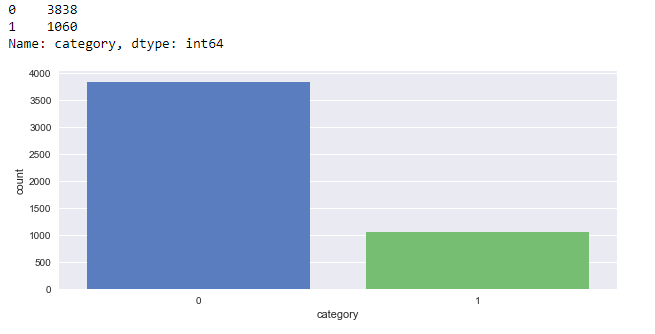


Figura 3 Gráfico que exibe o balanceamento dos dados

Trabalhar com uma base de dados desbalanceada exige atenção, já que esta característica pode trazer alguns problemas. Se uma base que é usada para treinar um modelo possui, por exemplo, apenas 5% de amostras na classe 1, e o modelo treinado acaba classificando todos os exemplares como 0, ele terá uma acurácia de cerca de 95%, o que pode parecer um resultado bom, quando na verdade não é, já que o modelo erraria a classificação de qualquer exemplar da classe 1. Outras métricas podem ser usadas para avaliar a qualidade do modelo, além da acurácia, como a precisão ao se classificar amostras em cada classe, isoladamente. Estas precisões podem ser tomadas utilizando-se a quantidade de falsos positivos e falsos negativos, ou seja, amostras classificadas erroneamente como positivas ou negativas [3].

Existem na literatura diversas formas de se tratar o problema do desbalanceamento das classes. Muitas partem da ideia primordial de se gerar novas amostras a partir da cópia de outras amostras da classe que possui menos representação, processo este que pode é chamado, em algumas referências, de *bootstraping* ou *oversampling* [4]. Outra estratégia, denominada *undersampling*, consiste na redução da quantidade de amostras da classe predominante, com o intuito de equilibrar as classes [4]. Uma técnica que é utilizada comumente para se contornar o problema do desbalanceamento é chamada *SMOTE (Syntetic Minority Oversampling Techinique)*. Ela cria novas amostras da classe minoritária, não apenas copiando amostras existentes, mas usando técnicas para gerar outras com base nas já existentes, como pode ser visto em [4]. O *SMOTE* será experimentado neste trabalho para se tentar chegar a melhores resultados na classificação dos vinhos do *data set* explorado aqui. Isto será exposto na sessão 3, com o auxílio da biblioteca *Sklearn* do Python, amplamente utilizada em problemas de aprendizado de máquina.

2.3 Correlação de Variáveis

Uma análise importante a se fazer das variáveis do problema tratado é a de correlação das variáveis. Os coeficientes de correlação permite mensurar qual forte é a relação entre duas variáveis [5]. Coeficientes de correlação, segundo a correlação de Pearson, a mais comumente usada [5], assumem valores entre -1 e 1. Quanto maior for o módulo da correlação entre duas variáveis, mais elas estão correlacionadas. Um coeficiente de correlação igual a 1 significa que cada incremento positivo em uma variável provoca incremento positivo de proporção fixa na outra variável. Por outro lado, um coeficiente de correlação igual a -1 significa que cada incremento positivo em uma variável provoca um decremento negativo de proporção fixa na outra variável. É importante deixar claro que a correlação não implica em causalidade [6]. Contudo, continua sendo uma análise pertinente para se entender a força das relações entre as variáveis num problema de aprendizado de máquina. Se uma variável possui alto coeficiente de correlação com a classe das amostras, este é um bom indicativo de que aquela variável pode ser mais relevante para a classificação. Se uma variável possui alto coeficiente de correlação com outra variável, tem-se um indicativo de que talvez estas variáveis podem possuir algum nível de redundância, sendo inclusive um indicativo para a redução de dimensionalidade do problema.

Para a análise de correlação entre as variáveis das amostras de vinhos tratadas aqui, o código da Figura 4 foi desenvolvido para a geração de uma matriz de correlação das variáveis do problema. A matriz resultante pode ser vista na Figura 5.



Figura 4 Código para a geração da matriz de correlação das variáveis do problema

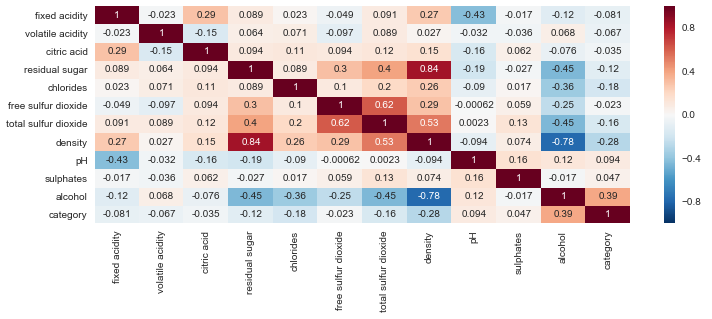


Figura 5 Matriz de correlação das variáveis do problema

Como se pode notar, a matriz de correlação tem a diagonal igual a 1. Isso é esperado porque o coeficiente de correlação entre uma variável e ela mesma é sempre igual a 1. A matriz é espelhada pela diagonal porque a correlação da variável A com a variável B é igual à correlação da variável B com a variável A, o que se reflete nos respectivos coeficientes de correlação. Pela Figura 5 percebe-se que a quantidade de álcool do vinho é o parâmetro que está mais positivamente correlacionado à sua qualidade (categoria), se comparado à correlação dos outros parâmetros medidos. Já a densidade é o parâmetro que possui a maior correlação negativa. Nota-se também que a densidade do vinho está muito positivamente correlacionada com a quantidade de açúcar residual, e que a quantidade de álcool está negativamente correlacionada à densidade, o que é de se esperar, já que a primeira está positivamente correlacionada à categoria, e a segunda está negativamente correlacionada à categoria.



Figura 6 Gráfico da tendência do crescimento da qualidade com o crescimento da quantidade de álcool

Tendo em vista os dados de correção obtidos, a Figura 6 mostra a tendência do crescimento da qualidade do vinho junto à tendência do crescimento da quantidade de álcool, dentro das amostras obtidas. Pode-se ver que de fato a qualidade tende a subir com o aumento do álcool, conforme o coeficiente de correlação indica.

2.4 Dados para Treino e Teste

Quando se lida com problemas de aprendizado supervisionado para classificação, é preciso separar parte do dado para treino do modelo, sendo que a outra parte é utilizada para testar o modelo treinado. Contudo, a escolha é feita levando-se em consideração que, quanto menor for a quantidade de dados para o treino, mais variância existirá na estimação dos parâmetros, menor acurácia terá o modelo. Por outro lado, quanto menor for a quantidade de dados para teste, as estatísticas de desempenho terão maior variância, já que os dados de treino passam a ter menor representatividade do dado real. Para o problema tratado aqui, com uma base de dados com 4898 amostras, optou-se por treinar os modelos com 80% dos dados, com os outros 20% sendo usados para testes. O código apresentado na Figura 7 faz a separação dos dados em treino e teste, com o auxilio da biblioteca *SKLearn* do Python.

Uma técnica que pode ser usada também, considerando o tamanho da base disponível, é a de validação cruzada, ou *cross-validation*. Utilizando o *K-Fold Cross-Validation*, o dado é divido em partes iguais, denominadas *folds*, e o modelo é treinado iterativamente nos *folds*, sendo que a cada iteração um deles é utilizado para testes, e os outros são utilizados para treino [7]. Sendo assim, com 5 *folds* é preciso fazer 5 iterações, para que cada um deles seja usado como teste. A acurácia do modelo pode ser tomada pela média das acurácias de cada iteração. Neste trabalho a validação cruzada será utilizada em alguns casos para verificar se a acurácia final será afetada por tal procedimento. Como o problema tratado aqui possui uma base desbalanceada, é importante fazer com que os *folds* sejam estratificados, de forma que cada um deles possua aproximadamente a mesma porcentagem de amostras em cada uma das classes. Para facilitar este procedimento, a biblioteca *SKLearn*, do Python, será utilizada. Serão utilizados 5 *folds*, valor este amplamente utilizado em diversas aplicações de aprendizado de máquina [7].

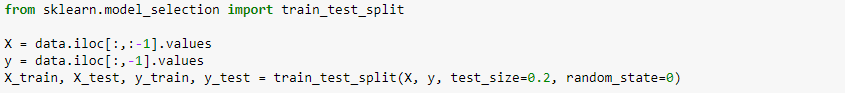


Figura 7 Código para a separação do dado em treino e teste

2.5 Modelos

Para o tratamento de problemas de classificação como este, é interessante pensar em quais modelos serão utilizados para se experimentar os dados. Neste trabalho, a Regressão Logística será utilizada como um modelo de *baseline,* devido à sua simplicidade. Considerando a experimentação e exploração, o algoritmo do *Random Forest* também será usado. Será possível visualizar a importância que o *Random Forest* dará às variáveis consideradas no problema. Conforme estudado em sala de aula, para bases de dados estruturadas, com a aqui tratada, modelos baseados em *boosting* são amplamente utilizados. Aqui serão experimentados o *AdaBoost* e o *GradientBoost*. Todos os modelos serão implementados com a ajuda do *SKLearn*.

2.6 Padronização do Conjunto de Dados

A padronização dos dados é um requisito comum de muitos estimadores de aprendizado de máquina implementados com o *SKLearn* [8]*.* Estes estimadores podem se comportar estranhamente ou ter um desempenho pior caso as características dos dados não sejam mais ou menos distribuídas segunda a normal padrão. Se as escalas das características forem muito diferentes, o modelo pode ter sua habilidade de aprendizado reduzida, dependendo do método utilizado, já que uma pequena variação no valor absoluto de um dado pode provocar um impacto considerável na classificação do mesmo, e o modelo pode não enxergar isso, se as características não tiverem escala adequada [8]. Neste trabalho será experimentada a diferença da padronização dos dados, com o auxílio de funções do próprio *SKLearn*.

3 EXPERIMENTOS E RESULTADOS

3.1 Modelo de Baseline

Para começar a experimentação dos dados, um modelo será utilizado como *baseline*, como um ponto de partida para se avaliar uma acurácia mínima desejada para a classificação dos dados. Neste trabalho o modelo utilizado como ponto de partida é o da Regressão Logística (*Logistic Regression*). Modelos de regressão logística se baseiam na análise da regressão utilizada quando a variável dependente é categórica, normalmente assumindo valores binários, e como o interesse aqui é obter a probabilidade de ocorrência de uma determinada classe, dadas as variáveis de entrada, os valores de saída estarão entre 0 e 1, utilizando a função logística, com formato sigmoidal [9]. A classificação é feita com base na maior probabilidade.

Através da biblioteca *SKLearn*, o código mostrado na Figura 8 faz o treino do modelo por regressão linear e a predição para os dados de teste. Utilizando a função apropriada, a precisão do modelo é exibida, e os valores podem ser visto na Tabela 1. Vale ressaltar que a precisão para a classe 0 é a razão entre a quantidade das amostras corretamente classificadas como 0 pela quantidade de todas as amostras classificadas como 0. O mesmo vale para a classe 1. O total é a razão da quantidade de amostras corretamente classificadas pela quantidade total de amostras. O modelo alcançou precisão total de 76%. Contudo, por causa da característica de desbalanceamento dos dados, a classificação como 1 possui precisão de apenas 58%, ou seja, o modelo acertou apenas 58% do vinhos classificados como bons. Os resultados apresentados são para taxa de aprendizado igual a 1, por ser a que apresentou melhor desempenho por experimentação.

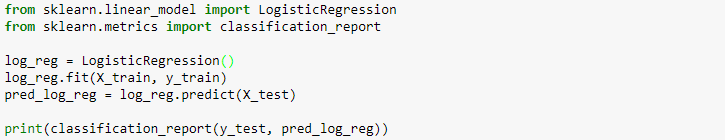


Figura 8 Código para o treino do modelo de regressão logística

Tabela 1 Precisões medidas para o modelo de regressão logística

|  |  |  |
| --- | --- | --- |
| **Classe 0** | **Classe 1** | **Total** |
| 0,81 | 0,58 | 0,76 |

3.2 Modelo Random Forest

O modelo do *Random Forest* utiliza um conjunto de árvores de decisão que subdivide o espaço de entrada em regiões menores. Para problemas de classificação, este modelo toma a estimativa de cada árvore do conjunto para obter a classificação com base na maioria dos votos [9]. O código mostrado na Figura 9 é o utilizado para treinar o modelo. O parâmetro *random\_state* é a semente do gerador de números aleatórios do modelo. Qualquer número pode ser passado por parâmetro. É interessante definir este parâmetro para que novas execuções do modelo com os mesmos parâmetros e dados não variem no resultado final. O parâmetro *n\_estimators* define a quantidade de árvores do algoritmo do *Random Forest*. Por experimentação, o valor 200 obteve melhor compromisso entre precisão final e tempo de execução. A Tabela 2 apresenta os resultados deste modelo. Como se pode ver, já se tem grande evolução na precisão de classificação, alcançando 87% de acertos. Este modelo lidou de forma bem melhor com o desbalanceamento dos dados, o que também pode ser observado pela precisão das classes na Tabela 2.

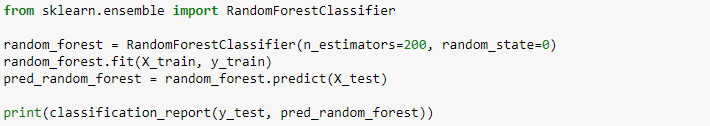


Figura 9 Código para o treino do modelo de *Random Forest*

Tabela 2 Precisões medidas para o modelo de *Random Forest*

|  |  |  |
| --- | --- | --- |
| **Classe 0** | **Classe 1** | **Total** |
| 0,89 | 0,82 | 0,87 |

Uma informação interessante que se pode obter do modelo de *Random Forest* treinado é a importância que ele deu a cada uma das características das amostras. O *SKLearn* facilita a obtenção desta informação. O código para a geração do gráfico de importância pode ser visto na Figura 10, e o gráfico resultante pode ser visto na Figura 11. Como se pode notar, pela analise feita em 2.3 da correlação das variáveis, o algoritmo do *Random Forest* deu mais importância ás características que possuem maior correlação com a variável de qualidade na base de dados, o que prova como a análise de correlação pode trazer um esboço antecipado das importâncias das características do dado.

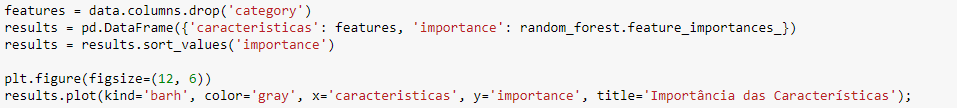


Figura 10 Código para a geração do gráfico de importâncias do modelo de *Random Forest*

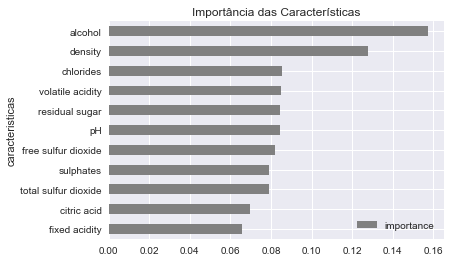


Figura 11 Importância das características para o *Random Forest*

3.3 Modelo de *Boosting*

Modelos baseados em *Boosting* seguem a ideia de se utilizar a combinação de modelos mais simples para se criar um modelo final mais eficiente. A final, é menos custoso se construir diversos classificadores simples do que se construir um único classificador complexo [10]. Classificadores simples também possuem a vantagem de terem menor tendência a cair em *overfit*, decorando o dado de treino, e consequentemente chegando a um pior desempenho real. Estes algoritmos também trabalham com a ponderação de diferentes amostras dos dados. A diferença entre os algoritmos de *boosting* está principalmente em dois pontos: a forma de atualização da ponderação do conjunto de dados e a forma de combinação dos classificadores parciais [10]. Uma forma de atualizar os pesos é aumentando os pesos dos exemplos classificados incorretamente em classificações parciais e uma forma simples de combinar os classificadores parciais é através da votação, onde cada classificador recebe um voto e a classificação final é decidida pelo voto da maioria. Conforme visto em sala de aula, algoritmos baseados em *Boosting* costumam ter bons desempenhos em bases de dados estruturadas, como é o caso da base de dados de qualidade de vinhos tratada neste trabalho. Os algoritmos que serão utilizados aqui são o *AdaBoost* e o *GradientBoost*.

Utilizando como auxilio a biblioteca *SKLearn* do Python mais uma vez, o código mostrado na Figura 12 foi feito para treinar o modelo com o *AdaBoost*. O parâmetro *n\_estimators* corresponde à quantidade de classificadores simplificados que deve ser usada. O tipo do classificador simplificado usado, por padrão, é a árvore de decisão de altura um (*stump*). O parâmetro *learning\_rate* é a taxa de aprendizado e *random\_state* é a semente do gerador de números aleatórios do modelo. Estes parâmetros foram ajustados por experimentação, utilizando a combinação que obteve o melhor compromisso entre precisão final e tempo de execução. Mesmo obtendo uma boa precisão para a classe menos recorrente, este algoritmo obteve uma precisão final de 79%, pior do que o algoritmo de *Random Forest*. Seguindo a mesma metodologia do *AdaBoost*, a Figura 12 mostra o código para o treino do modelo *GradientBoost*, e segundo a Tabela 3, a precisão deste foi de 80%. Ele possui um desempenho total melhor que o *AdaBoost*, mas teve precisão um pouco menor para classificar a classe menos recorrente.

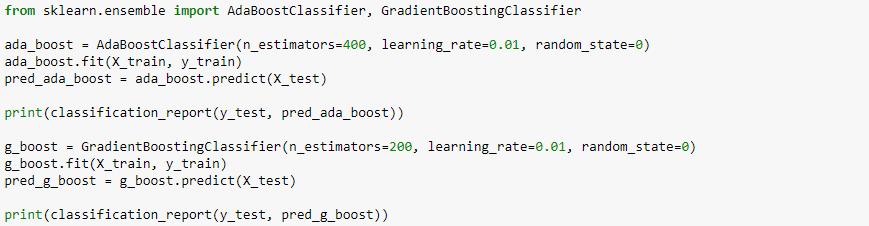


Figura 12 Código para o treino dos modelos *AdaBoost* e *GradientBoost*

Tabela 3 Precisões medidas para o modelo *AdaBoost* e *GradientBoost*

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Classe 0** | **Classe 1** | **Total** |
| **AdaBoost** | 0,79 | 0,82 | 0,79 |
| **GradientBoost** | 0,81 | 0,75 | 0,80 |

3.4 Ajustando a Base de Dados

O código mostrado na Figura 13 faz algumas mudanças na base de dados, mudanças estas discutidas na Sessão 2. Com o auxilio do *SKLearn* e de outra biblioteca chamada *IMBLearn*, o desbalanceamento de classes no *data set* foi tratado, com a técnica do *SMOTE*, e a base de dados teve a escala padronizada. Os dois melhores modelos treinados até aqui, *GradientBoost* e *Random Forest*, foram treinados novamente com estas mudanças na base. Os resultados obtidos podem ser vistos na Tabela 4. Como se pode notar, o algoritmo do *GradientBoost* teve uma precisão final um pouco reduzida, mas teve melhor desempenho ao classificar amostras da classe que anteriormente era menos presente. Já o *Random Forest* obteve melhor desempenho em todas as três precisões, se mostrando o melhor algoritmo para a base de classificação de vinhos em questão. Obteve precisão total de 92%.

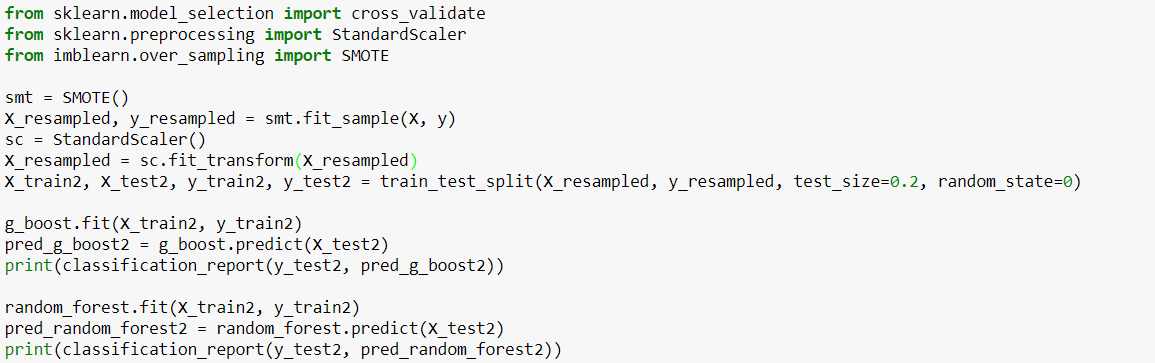


Figura 13 Código para tratamento de desbalanceamento e acerto de escala

Tabela 4 Tabela de resultado dos tratamentos

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Classe 0** | **Classe 1** | **Total** |
| **GradientBoost** | 0,81 | 0,78 | 0,79 |
| **Random Forest** | 0,93 | 0,91 | 0,92 |

Pelo código mostrado na Figura 14, estes dois últimos algoritmos passaram pelo processo de validação cruzada, para que todas as amostras pudessem ser usadas tanto em teste quanto em treino, conforme visto em 2.4. Foi utilizada a validação cruzada com 5 *folds* estratificada, para que todos os *folds* tenham a mesma proporção de amostras nas duas classes. O algoritmo do *GradientBoost* obteve acurácia de 77% após a *cross validation,* enquanto que o *Random Forest* obteve precisão de 87%.

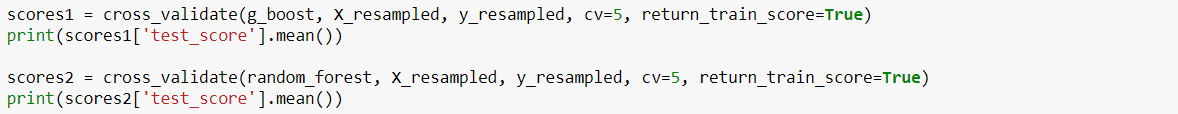


Figura 14 Código para o *cross validation* nos modelos do *GradientBoost* e *Random Forest*

4 CONCLUSÃO

Algoritmos de aprendizado de máquina estão sendo utilizados nos mais diversos campos e aplicabilidades, e isso inclui a classificação de qualidade de produtos. Pelos experimentos apresentados neste trabalho, conclui-se que o objetivo de apresentar os primeiros passos para se manipular dados com a teoria do aprendizado de máquina foi alcançado com sucesso. Os procedimentos para a classificação de uma base de dados foram apresentados de forma clara, sucinta de didática, e o resultado final, considerando o tamanho e as características do *data set* de vinhos produzidos pela Vinho Verde, foi satisfatório. Como próximos passos e experimentações, aconselha-se a utilização de bases de dados com mais amostras e características, para que técnicas como o preenchimento de características faltantes e transformação de variáveis que estão em formato de texto possam ser feitas. É possível também trabalhar com o redimensionamento dos dados. Todas estas são técnicas que podem melhorar o desempenho de algoritmos de aprendizado de máquina. Pode-se também experimentar outros modelos, como os baseados em *Deep Learning*, ou outros comumente usados pelo mercado. A conclusão final é que o resultado do trabalho foi satisfatório.

REFERÊNCIAS

|  |  |
| --- | --- |
| [1] | **Cortez, Paulo.** Wine Quality Data Set [Online] // Machine Learning Data Set. 2009. https://archive.ics.uci.edu/ml/datasets/wine+quality. Acesso em: 28/03/2018. |
| [2] | **Kaggle.** Red Wine Quality [Online] // Kaggle. 03/12/2017. <https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009>. Acesso em: 02/04/2018. |
| [3] | **Chioka.** Class Imbalance Problem [Online] // Kaggle. 30/08/2013. http://www.chioka.in/class-imbalance-problem/. Acesso em: 05/04/2018. |
| [4] | **Fawcett, Tom.** Learning from Imbalanced Classes [Online] // Silicon Valley Data Science. 25/08/2016. https://www.svds.com/learning-imbalanced-classes/. Acesso em: 29/03/2018. |
| [5] | **Statistics How To.** Correlation Coefficient [Online] // Statistics How To. 03/04/2017. http://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/. Acesso em: 07/04/2018. |
| [6] | **Kelleher, Adam.** If Correlation Doesn’t Imply Causation, Then What Does? [Online] // Medium. 27/06/2016. https://medium.com/causal-data-science/if-correlation-doesnt-imply-causation-then-what-does-c74f20d26438. Acesso em: 07/04/2018. |
| [7] | **Gupta, Prashant.** Cross-Validation in Machine Learning [Online] // Towards Data Science. 05/06/2017. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f. Acesso em: 19/04/2018. |
| [8] | **Scikit Learn.** Preprocessing data [Online] // SKLearn. 2017. http://scikit-learn.org/stable/modules/preprocessing.html. Acesso em: 27/04/2018. |
| [9] | **Fred, Luís.** O Machine Learning e seus principais algoritmos de aprendizagem supervisionada [Online] // LinkedIn. 20/11/2017. https://pt.linkedin.com/pulse/o-machine-learning-e-seus-principais-algoritmos-de-lu%C3%ADs-fred. Acesso em: 27/04/2018. |
| [10] | **Duarte, Julio Cesar.** O Algoritmo Boosting at Start e suas Aplicações [Tese] // PUC Rio. 08/09/2009. |
|  |  |
|  |  |