UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Amanda Pereira Ferraz

Prevendo a aprovação de um participante do Enem no SiSU para o curso de Medicina

São Carlos

2020

Amanda Pereira Ferraz

Prevendo a aprovação de um participante do Enem no SiSU para o curso de Medicina

Trabalho de conclusão de curso apresentado ao Centro de Ciências Matemáticas Aplicadas à Indústria (CeMEAI) e ao Instituto de Ciências Matemáticas e de Computação (ICMC/USP), Universidade de São Paulo, como parte dos requisitos para a obtenção do MBA em Ciências de Dados.

Área de concentração: Ciências de Dados
Opção: MBA em Ciências de Dados

Orientador: Prof. Dr. Carlos Alberto Ribeiro Diniz

Versão original

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

F381p

Ferraz, Amanda Pereira
Prevendo a aprovação de um participante do Enem no SiSU para o curso de Medicina. / Amanda Pereira Ferraz; orientador Carlos Alberto Ribeiro Diniz. -- São Carlos, 2020.
70 p.

Trabalho de conclusão de curso (MBA em Ciência de Dados) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Machine Learning. 2. Random Forest. 3. Boosting. 4. Enem. 5. SiSU. I. Ribeiro Diniz, Carlos Alberto, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

RESUMO

O principal objetivo deste trabalho é aplicar técnicas de Machine Learning, como os algoritmos de Random Forest e Boosting, para prever qual aluno será aprovado ou não para o curso de Medicina na universidade com a menor nota de corte nacional considerando o desempenho obtido por ele no SiSU 2020. Para uma base de dados com mais de 3 milhões de registros, é possível perceber que todos os algoritmos utilizados obtiveram um resultado satisfatório, visto que em todos os casos o Coeficiente de Correlação de Matthews (MCC) foi superior a 0,90.

Palavras-chave: Enem, SiSU, Machine Learning, Random Forest, Boosting, Coeficiente de Correlação de Matthews.

ABSTRACT

The main objective of this work is to apply Machine Learning techniques, such as the Random Forest and Boosting algorithms, to predict which student will be approved or not for Medicine at the university with the lowest national cut grade. considering the performance obtained by him in SiSU 2020. For a database with more than 3 million records, it is possible to notice that all the algorithms used obtained a satisfactory result, since in all cases the Matthews Correlation Coefficient (MCC) was greater than 0.90.

Keywords: Enem, SiSU, Machine Learning, Random Forest, Boosting, Matthews Correlation Coefficient.

LISTA DE FIGURAS

Figura 1 – Atributos que estão mais correlacionados à variável resposta 34
Figura 2 – Importância relativa das variáveis do Random Forest com 101 árvores . 38

LISTA DE TABELAS

Tabela 1 – Vagas para Medicina no SiSU 2020 por Estado 19
Tabela 2 – Vagas para Medicina no SiSU 2020 por Região 20
Tabela 3 – Número total de inscritos no Enem de 2015 a 2019 29
Tabela 4 – Número total de participantes no Enem de 2015 a 2019 que compõem o grupo de interesse 30
Tabela 5 – Desempenho do grupo de interesse no Enem 2015 30
Tabela 6 – Desempenho do grupo de interesse no Enem 2016 31
Tabela 7 – Desempenho do grupo de interesse no Enem 2017 31
Tabela 8 – Desempenho do grupo de interesse no Enem 2018 31
Tabela 9 – Desempenho do grupo de interesse no Enem 2019 32
Tabela 10 – Desempenho médio do grupo de interesse 32
Tabela 11 – Acurácia dos modelos Random Forest variando a quantidade de árvores 35
Tabela 12 – Métricas de avaliação para o modelo Random Forest com 101 árvores . 37
Tabela 13 – Métricas de avaliação para o modelo Árvore de Decisão 39
Tabela 14 – Métricas de avaliação para o modelo de classificação Adaboost com estimador base Random Forest 41
Tabela 15 – Métricas de avaliação para o modelo de classificação Adaboost com estimador base Árvore de Decisão 42

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Justificativa e Importância 13
1.2 Objetivo Geral 15
1.2.1 Objetivos Específicos 15
1.3 Referencial da Pesquisa 15

2 BANCO DE DADOS 17

3 MATERIAL E MÉTODOS 23
3.1 Métodos Ensemble 23
3.1.1 Random Forest (Floresta Aleatória) 24
3.1.2 Boosting 25
3.2 Métricas do Poder Preditivo 27

4 RESULTADOS E DISCUSSÕES 29
4.1 Análise Descritiva dos Dados 29
4.2 Pré-Processamento de Dados 32
4.3 Modelos de Machine Learning 34
4.3.1 Random Forest 34
4.3.2 Árvore de Decisão 38
4.3.3 Boosting 40
4.3.3.1 Adaboost com estimador base Random Forest 40
4.3.3.2 Adaboost com estimador base Árvore de Decisão 42
4.4 Trabalhos Futuros 43

5 CONCLUSÃO 45

REFERÊNCIAS 47

APÊNDICES 49

APÊNDICE A – PACOTES EM PYTHON UTILIZADOS NO TCC 51

APÊNDICE B – PROGRAMAÇÃO EM PYTHON - CÓDIGOS . . 53

ANEXOS 67

ANEXO A – VARIÁVEIS UTILIZADAS NOS MODELOS DE MACHINE LEARNING 69

1 INTRODUÇÃO

1.1 Justificativa e Importância

O Exame Nacional do Ensino Médio (Enem)[1] foi a primeira iniciativa ampla de avaliação do sistema educacional implementado no Brasil e foi utilizado pela primeira vez para avaliar a qualidade da educação brasileira, o teste foi aplicado aos alunos do terceiro ano do ensino médio em todo o país para auxiliar o Ministério da Educação no desenvolvimento de políticas pontuais e estruturais com o objetivo de melhorar o ensino médio brasileiro.

Em 1998, foi realizada a primeira edição do Exame Nacional do Ensino Médio (Enem) que contou com 157.221 inscritos e 115.575 participantes e as notas obtidas puderam ser utilizadas em apenas duas instituições de educação superior.

O Sistema de Seleção Unificada (SiSU)[2] é de responsabilidade do Ministério da Educação (MEC) e é a maneira pela qual muitas instituições públicas de ensino superior atualmente oferecem vagas aos candidatos que participam do Exame Nacional do Ensino Médio (Enem). Isso ocorre porque, desde 2009, o governo implementou essa seleção nacional, o que facilitou a democratização e o acesso às vagas em instituições públicas de ensino superior. O SiSU, juntamente com o Exame Nacional do Ensino Médio (Enem), possibilita ao estudante candidatar-se a vagas em instituições públicas de todo o Brasil.

A partir de 2009, com a criação do SiSU, o Enem mudou de formato e passou a ter 180 questões objetivas, 45 para cada área do conhecimento (Ciências da Natureza, Ciências Humanas, Linguagens e Códigos e, Matemática), e a redação. A aplicação começou a ser feita em dois dias e o exame passou a certificar a conclusão do ensino médio. Nessa edição, 4.138.025 pessoas se inscreveram no Enem.

Além do SiSU, a nota do Enem também é válida como critério para concessão de bolsas para o Programa Universidade para Todos (ProUni), foi utilizada com um dos pré-requisitos para concessão de bolsas do Ciência sem Fronteiras (CsF), e também podia ser aproveitada para a obtenção do certificado de conclusão do ensino médio para maiores de 18 anos, desde que o candidato tivesse feito pelo menos 450 pontos em cada um das provas objetivas e pelo menos 500 pontos na redação. Entretanto, desde 2017, o Enem não certifica o ensino médio, função que retornou ao Exame Nacional para Certificação de Competências de Jovens e Adultos (Encceja).

Em 2010, os resultados do Enem passaram a ser adotados pelo Fies e em 2013, pela primeira vez, quase todas as instituições federais adotaram o Enem como critério

[1] Texto sobre o Enem: https://enem.inep.gov.br/
[2] Texto sobre o SiSU: http://sisu.mec.gov.br/inicial

de seleção e a nota obtida no exame foi utilizada na concessão de bolsas de estudos do programa Ciência sem Fronteiras. Em 2014, as Universidades de Coimbra e Algarve, em Portugal, passaram a aceitar o Enem, marcando o início das parcerias com instituições de ensino superior de Portugal, autorizadas a utilizar as notas do Enem em seus processos seletivos e em 2018, o número de instituições de educação superior portuguesas que usaram as notas do Enem chegou a 35.

No ano de 2015, começou a ser quantificado o número de "treineiros", participantes que fazem o Enem para autoavaliação. Nesse primeiro ano de levantamento, o Inep registrou que 12% dos 7.792.024 inscritos eram "treineiros".

O Enem se consolida a cada ano, visto que o número de participantes aumentou substancialmente, foram quase 158 mil candidatos em 1998 e em 2019, com a mudança de formato do exame, foram mais de 5 milhões de inscritos.

A tendência é que, no futuro, o Enem substitua completamente os vestibulares da maioria das instituições de ensino do país, já que grande parte das universidades públicas e das faculdades privadas já estão adotando o exame como uma forma de ingresso no ensino superior.

Hoje, instituições de todo o Brasil aderiram ao SiSU como uma forma de admissão parcial ou total nos seus cursos de graduação.

O SiSU possui instituições de todos os estados do Brasil. Existem mais de 100 faculdades particulares e universidades públicas oferecendo cursos de graduação por meio do programa.

Para que os alunos do ensino médio obtenham uma vaga em cursos concorridos como Medicina, Direito e Engenharia, eles devem ter uma média geral maior que 700 pontos no total.

A motivação deste trabalho é ser capaz de definir a partir de um conjunto de variáveis - como renda, idade, sexo, raça, estado civil, se tem alguma deficiência, entre outras características - o que contribui para um participante do Enem 2019 ser aprovado ou não em Medicina na universidade com a menor nota de corte nacional considerando o desempenho obtido por ele no SiSU 2020.

O presente trabalho permite uma nova abordagem sobre o desempenho dos alunos no Enem e pode ser considerado muito relevante no que diz respeito ao ingresso dos estudantes no ensino superior brasileiro. Essa análise poderá ser estendida para os demais cursos e, pode ser possível, a partir disso, implementar políticas públicas que visem a expansão das vagas nos cursos de nível superior. A partir desse trabalho, pode ser possível também estruturar políticas públicas voltadas para a melhoria do ensino médio público, visto que o desempenho dos participantes do Enem tende a revelar a situação do ensino público no país.

1.2 Objetivo Geral

O principal objetivo deste estudo é aplicar técnicas de Machine Learning para prever qual aluno será aprovado ou não para o curso de Medicina na universidade com a menor nota de corte nacional considerando o desempenho obtido por ele no SiSU 2020.

1.2.1 Objetivos Específicos

a) Estudar os modelos de classificação Random Forest e Boosting;

b) Comparar a performance dos modelos Random Forest e Boosting, utilizando a linguagem de programação Python, com o uso de medidas de avaliação da capacidade preditiva, como acurácia, precisão, revocação, área sob a curva (Curva ROC), entre outras;

c) Aplicar a metodologia estudada nos dados disponíveis do Enem 2019 e do SiSU 2020.

1.3 Referencial da Pesquisa

O primeiro artigo (SILVA; MORINO; SATO, 2014) utiliza os dados dos questionários socioeconômicos preenchidos pelos inscritos participantes da edição do ENEM 2010 das capitais da região Sudeste do Brasil com o objetivo de determinar padrões. Essas informações, mais especificamente as respostas para quatro perguntas (A quantidade de pessoas que moram com o aluno possui algum tipo de interferência na nota do aluno no ENEM?, O grau de escolaridade da mãe tem relação com o desempenho do aluno na prova objetiva?, O valor da renda familiar mensal contribui para o desempenho do aluno? e O tipo de escola em que o aluno estudou no Ensino Médio afeta o seu desempenho na prova?), estão relacionadas aos resultados de desempenho desses alunos. Os autores utilizaram a técnica de mineração baseada em regras de associação, utilizando diferentes parametrizações do algoritmo A Priori. Nas conclusões (AGRAWAL; SRIKANT, 1994), foram identificados fatores que diminuem o desempenho dos alunos, como a baixa renda familiar, a educação dos pais ser de nível primário e o fato de ser alto o número de pessoas que moram com o estudante.

Outro estudo (STEARNS et al., 2014) avaliou dois métodos de regressão baseados em Boosting de árvores de decisão, para prever a nota de um estudante no ENEM 2014 baseando-se somente no questionário socioeconômico. Os modelos utilizados foram métodos de regressão baseados em árvore de decisão. Este trabalho comparou duas técnicas de boosting para agregar Árvores de Decisão: Gradient Boosting e AdaBoost. Embora esses agregadores melhorem a capacidade da regressão, o número de hiper-parâmetros fica maior, o que torna mais difícil encontrar um conjunto que defina um modelo ótimo, tanto para as árvores quanto para as técnicas de Boosting. Os melhores resultados foram obtidos com o Gradient Boosting. Os autores concluíram que os indicadores socioeconômicos poderiam

explicar um viés nas pontuações dos participantes e que seria possível prever a nota do inscrito no exame com valores de métricas adequados. Outro aspecto que vale ressaltar é que as informações mais importantes para o modelo foram, respectivamente, a longitude, a latitude, a idade, o motivo de realizar o Enem para ingressar no ensino privado, o ano de conclusão do ensino médio, a renda mensal familiar, o motivo de realizar o Enem para obter o financiamento do FIES, a quantidade de pessoas que moram na mesma residência, o motivo de realizar o Enem para aumentar a possibilidade de conseguir um emprego e a idade em que começou a exercer uma atividade remunerada.

A árvore de decisão é uma estrutura hierárquica para a decisão sequencial na forma de uma árvore invertida da raiz às folhas (SAFAVIAN; LANDGREBE, 1991) que particiona o espaço de entrada para maximizar o ganho de informações em relação ao objetivo. As árvores de decisão explicitam o conhecimento incorporado nos dados de uma maneira compreensível pelas regras se-então. No topo da árvore estão os atributos mais importantes.

Outro artigo (MARTINS; FARIA, 2010) expõem alguns pontos que podem ser melhorados a partir do conhecimento dos perfis das instituições de ensino de Santa Catarina sob a ótica dos inscritos no Enem 2008. Por meio da análise do questionário socioeconômico do Enem foi definido o perfil dos candidatos e foram considerados alguns pontos que podem ser melhorados no ensino médio do estado de Santa Catarina. Do ponto de vista dos candidatos, a avaliação geral do ensino médio foi positiva. 83,91% desses candidatos atribuíram nota acima da média sete para a sua formação. Entretanto, essa avaliação contradisse a realidade demonstrada pelo desempenho no exame, pois 95% e 74,77% respectivamente, das notas da prova objetiva e redação, ficaram abaixo da média sete. Com isso, foi possível concluir que há alguns pontos que podem ser melhorados na rede pública de ensino, pois apesar dos alunos acreditarem que se encontravam preparados, a maioria deles não atingiu a nota mínima esperada para aprovação no exame.

Outro estudo (MARCIANO; DAVIM, 2017) cria modelos preditivos que baseados em diversos fatores sobre o inscrito no Enem 2015 são capazes de prever a nota de redação obtida por ele. Os autores desse trabalho esperam que a análise realizada por eles não seja apenas um exercício computacional, mas também uma fonte de informação para posteriores análises sociológicas, pois acreditam que esse trabalho possa ter um impacto positivo na discussão do modelo ideal de acesso ao ensino superior brasileiro. Nele, os autores utilizaram 5 modelos de predição, entre eles a árvore de decisão, com o objetivo de comparar o desempenho de tais técnicas por meio das métricas de validação previamente discutidas, a fim de investigar qual opção se ajusta melhor ao dataset Enem 2015.

2 BANCO DE DADOS

Para identificar os fatores que mais influenciam no fato de um participante do Enem 2019 ser aprovado em Medicina no SiSU 2020 na universidade com a menor nota de corte nacional serão utilizadas apenas as variáveis das categorias: dados do participante, dados dos pedidos de atendimento especializado, dados da prova objetiva, dados da redação e dados do questionário socioeconômico.

As variáveis disponíveis em cada uma dessas categorias são as seguintes:

- Dados do participante: Município de residência, Unidade da Federação de residência, Idade, Sexo, Estado civil, Cor/raça, Nacionalidade, Situação de conclusão do Ensino Médio, Ano de conclusão do Ensino Médio, Tipo de escola do Ensino Médio.

- Dados dos pedidos de atendimento especializado: Indicador de baixa visão, Indicador de cegueira, Indicador de surdez, Indicador de deficiência auditiva, Indicador de surdo-cegueira, Indicador de deficiência física, Indicador de deficiência mental, Indicador de déficit de atenção, Indicador de dislexia, Indicador de discalculia, Indicador de autismo, Indicador de visão monocular, Indicador de outra deficiência ou condição especial. Esse grupo de variáveis tem a função de indicar se o inscrito tem alguma deficiência, por esse motivo, foi criada a partir de todas essas, uma única variável genérica indicativa de deficiência, que atende esse objetivo.

- Dados da prova objetiva: Nota da prova de Ciências da Natureza, Nota da prova de Ciências Humanas, Nota da prova de Linguagens e Códigos, Nota da prova de Matemática, Língua Estrangeira.

- Dados da redação: Nota da prova de redação.

- Dados do questionário socioeconômico: Nível de escolaridade do pai do inscrito, Nível de escolaridade da mãe do inscrito, Ocupação do pai do inscrito, Ocupação da mãe do inscrito, Quantidade de pessoas que moram na residência do inscrito, Renda mensal familiar do inscrito, Se na residência do inscrito tem empregado(a) doméstico(a), Se na residência do inscrito tem banheiro, Quantidade de quartos para dormir que tem na residência do inscrito, Se na residência do inscrito tem carro, Se na residência do inscrito tem motocicleta, Se na residência do inscrito tem geladeira, Se na residência do inscrito tem freezer, Se na residência do inscrito tem máquina de lavar roupa, Se na residência do inscrito tem máquina de secar roupa, Se na residência do inscrito tem micro-ondas, Se na residência do inscrito tem máquina de lavar louça, Se na residência do inscrito tem aspirador de pó, Se na residência do

inscrito tem televisão em cores, Se na residência do inscrito tem aparelho de DVD, Se na residência do inscrito tem TV por assinatura, Se na residência do inscrito tem telefone celular, Se na residência do inscrito tem telefone fixo, Se na residência do inscrito tem computador, Se na residência do inscrito tem acesso à Internet.

Os inscritos no Enem 2019 apresentam as seguintes características: [1]

- As mulheres seguem sendo maioria entre os inscritos: 59,5% dos confirmados são do sexo feminino, contra 40,5% do sexo masculino.
- Em relação à faixa etária, 26,7% tem de 21 a 30 anos, 17,8% tem 17 anos; 15,9% tem 18 anos.
- Ao todo, 46,4% se autodeclararam pardos; 36%, brancos; e 12,7%, pretos.
- Em relação à situação escolar, 28,8% são concluintes do ensino médio este ano. Os egressos, aqueles que já se formaram em anos anteriores, representam 58,7%. E os treineiros, participantes que não vão concluir a educação básica este ano, são 12,1% dos inscritos.

O arquivo de notas de corte do SiSU 2020 para o curso de Medicina apresenta 4 variáveis, são elas: Universidade, Quantidade de Vagas, Campus e Nota de corte.[2]

A nota de corte para o curso de Medicina no SiSU 2020 considerada para verificar se um participante do ENEM 2019 seria aprovado em Medicina na universidade com a menor nota de corte nacional levou em consideração a nota da modalidade da ampla concorrência. A maior nota de corte em Medicina no Sisu 2020 foi 928,13 na Universidade Federal do Maranhão - UFMA - Campus de São Luís, já a menor nota de corte em Medicina no SiSU 2020 foi 736,04 na Universidade Federal Rural Do Semi Árido - UFERSA - Unidade Sede. A nota de corte média no SiSU 2020 foi 788,18, considerando os 88 cursos de Medicina que fazem parte do SiSU em todo Brasil.

O SiSU 2020 contou com 4.457 vagas para o curso de Medicina em todo o Brasil e a distribuição dessas vagas por Estado pode ser vista na Tabela 1.

[1] Texto sobre o perfil dos participantes do Enem 2019: http://portal.mec.gov.br/busca-geral/140-programas-e-acoes-1921564125/enem-exame-nacional-do-ensino-medio-1766000317/76581-mais-de-5-milhoes-de-participantes-estao-confirmados-para-a-edicao-do-exame-em-2019s

[2] Informações sobre a note de corte para o curso de Medicina no SiSU 2020: https://www.blogdovestibular.com/sisu-2020/notas-de-corte-medicina-sisu-2020.html

Tabela 1: Vagas para Medicina no SiSU 2020 por Estado

Estado	Quantidade de Vagas
Minas Gerais (MG)	796
Rio Grande do Sul (RS)	408
Bahia (BA)	301
Rio de Janeiro (RJ)	300
Ceará (CE)	280
Rio Grande do Norte (RN)	240
Mato Grosso (MT)	228
Paraíba (PB)	195
Pernambuco (PE)	185
Goiás (GO)	170
Alagoas (AL)	160
Sergipe (SE)	160
Maranhão (MA)	130
Piauí (PI)	130
São Paulo (SP)	128
Mato Grosso do Sul (MS)	126
Paraná (PR)	108
Distrito Federal (DF)	80
Espírito Santo (ES)	80
Tocantins (TO)	80
Amazonas (AM)	56
Acre (AC)	40
Amapá (AP)	30
Santa Catarina (SC)	30
Roraima (RR)	16

Fonte: Elaborada pela autora.

Os estados do Pará e de Rondônia não possuem nenhum curso de Medicina que faça parte do SiSU 2020.

A Tabela 2 apresenta o total de vagas para Medicina no SiSU 2020 por região:

A nota de corte que será utilizada para verificar se um determinado participante

Tabela 2: Vagas para Medicina no SiSU 2020 por Região

Região	Quantidade de Vagas
Nordeste	1.781
Sudeste	1.304
Centro-Oeste	604
Sul	546
Norte	222

Fonte: Elaborada pela autora.

do Enem seria aprovado em Medicina na universidade com a menor nota de corte nacional será a nota de corte da modalidade da ampla concorrência na Universidade Federal Rural Do Semi Árido - UFERSA - Unidade Sede.

A nota que será comparada com essa nota de corte será a média simples das cinco notas obtidas por cada um dos candidatos que fez o Enem (Ciências da Natureza, Ciências Humanas, Linguagens e Códigos, Matemática e redação), pois como cada universidade tem a possibilidade de atribuir pesos distintos para cada uma das provas, e o objetivo é verificar se um determinado participante do Enem seria aprovado em Medicina na universidade com a menor nota de corte nacional, o cálculo da nota será realizado de maneira genérica.[3]

As variáveis listadas abaixo, apesar de estarem disponíveis no arquivo de microdados do Enem 2019, não serão utilizadas como covariáveis na construção dos modelos, pois a variável do primeiro grupo está em branco para mais de 65,3% dos 26.169 participantes do Enem 2019 que seriam aprovados em Medicina no SiSU 2020 na universidade com a menor nota de corte nacional, já as variáveis do segundo grupo estão sem informação para mais de 75,8% considerando esses mesmos participantes.

- Dados do participante: Tipo de instituição que concluiu ou concluirá o Ensino Médio.

- Dados da escola: Município da escola, Unidade da Federação da escola, Dependência administrativa (Escola), Localização (Escola), Situação de funcionamento (Escola).

As variáveis Município de nascimento e Unidade da Federação de nascimento da categoria dados do participante, apesar de estarem preenchidas para grande parte dos registros, encontram-se em branco para alguns deles e por essa razão também não serão utilizadas como covariáveis na construção dos modelos de Machine Learning.

[3] Texto sobre o cálculo da nota obtida no SiSU: https://blog.mesalva.com/de-tudo-um-pouco/sisu-como-calcular-sua-media/

Como apenas 26.169 participantes do grupo de interesse, que é composto por 3.089.609 de inscritos, seriam aprovados no curso de Medicina, os modelos de Machine Learning serão construídos levando em consideração que as classes são desbalanceadas. [4]

[4] Aplicação no Kaggle para detecção de fraudes com Random Forest: https://www.kaggle.com/pileatedperch/detecting-fraud-with-random-forest-mcc-0-869

3 MATERIAL E MÉTODOS

O referencial teórico deste trabalho abordará os seguintes tópicos:

- Random Forest;
- Boosting.

Esses serão os assuntos abordados, visto que o objetivo deste trabalho é desenvolver modelos de classificação que sejam capazes de prever qual aluno será aprovado ou não para o curso de Medicina na universidade com a menor nota de corte nacional considerando o desempenho obtido por ele no SiSU 2020 e, para isso, serão aplicadas técnicas de Machine Learning para definir os modelos de classificação Random Forest e Boosting.

3.1 Métodos Ensemble

Métodos Ensemble[1] são uma categoria de algoritmos de Machine Learning, que podem ser usados tanto em aprendizagem supervisionada quanto não supervisionada. Esses algoritmos têm uma técnica eficaz e poderosa para alcançar alta precisão em soluções supervisionadas e não supervisionadas.

Os métodos ensemble permitem combinar modelos concorrentes para formar uma espécie de comitê, e tem havido muita pesquisa nesta área com um bom grau de sucesso.

Um princípio amplamente discutido em Machine Learning, é que a acurácia e a simplicidade do modelo levam ao estado da arte em análise preditiva. Mas combinar esses dois aspectos normalmente é muito difícil, o que leva a um trade-off: um modelo mais complexo é mais flexível, mas consequentemente mais suscetível ao overfitting e provavelmente não vai generalizar bem em novos conjuntos de dados. Já os modelos mais simples podem não conseguir realizar o aprendizado de forma efetiva, embora reduzam o tempo total de treinamento.

As técnicas de regularização tem sido utilizadas como forma de se atingir o estado da arte em Machine Learning, aplicando uma função de erro que penaliza a complexidade do modelo. A regularização é apontada como uma das principais razões da performance muito superior de modelos de métodos ensemble.

Construir um modelo ensemble consiste em dois passos:

- Construir diversos modelos;

[1] Informações sobre Métodos Ensemble: https://didatica.tech/metodos-ensemble/

- Combinar suas estimativas.

Pode-se gerar modelos diferentes, variando, por exemplo, os pesos das observações, os valores de dados, os parâmetros, os subconjuntos de variáveis ou aplicando as diferentes técnicas de pré-processamento. A combinação pode ser feita por meio de votação, mas normalmente a combinação é feita por meio de pesos das estimativas dos modelos. Os métodos Ensemble permitem aumentar consideravelmente o nível de precisão nas suas previsões.

O Random Forest é um método ensemble que combina diversas árvores de decisão e adiciona um componente estocástico para criar mais diversidade entre as árvores de decisão.

O Adaboost constrói diversos modelos de forma iterativa, variando o peso nos exemplos no dataset de treino, penalizando exemplos com muitos erros e reduzindo o peso daqueles com estimativa incorreta ou menos precisa.

O Gradient Boosting é uma extensão do AdaBoost com uma variedade de funções de erro para classificação.

3.1.1 Random Forest (Floresta Aleatória)

Floresta Aleatória é um algoritmo de aprendizagem de máquina flexível e fácil de usar que produz excelentes resultados a maioria das vezes, mesmo sem ajuste de hiperparâmetros. É também um dos algoritmos mais utilizados, devido à sua simplicidade e o fato de que pode ser utilizado para tarefas de classificação e também de regressão. (BREIMAN, 2001)

Floresta Aleatória é um algoritmo de aprendizagem supervisionada. Esse algoritmo cria uma floresta de um modo aleatório. A "floresta" que ele cria é uma combinação (ensemble) de árvores de decisão.

As florestas aleatórias assumem que os modelos básicos são árvores de classificação ou de regressão. A idéia é adicionar aleatoriedade ao construir cada árvore, com o objetivo de reduzir ainda mais a correlação entre os modelos de base. A princípio, essa ideia pode parecer estranha: incluir aleatoriedade ao treinamento de um modelo deve comprometer/piorar o seu desempenho.

Uma grande vantagem do algoritmo de florestas aleatórias é que ele pode ser utilizado tanto para tarefas de classificação quanto para regressão.

O algoritmo de floresta aleatória adiciona aleatoriedade extra ao modelo, quando está criando as árvores. Ao invés de procurar pela melhor característica ao fazer a partição de nós, ele busca a melhor característica em um subconjunto aleatório das características.

Este processo cria uma grande diversidade, o que geralmente leva a geração de modelos melhores.

Assim, quando uma árvore é criada no modelo floresta aleatória, apenas um subconjunto aleatório das características é considerado na partição de um nó. É possível fazer as árvores ficarem mais aleatórias utilizando limiares (thresholds) aleatórios para cada característica, ao invés de procurar pelo melhor limiar (como uma árvore de decisão geralmente faz).

3.1.2 Boosting

Boosting é diferente das florestas aleatórias, pois seus modelos de base são treinados sequencialmente, um após o outro, e cada modelo tenta corrigir os erros cometidos pelos modelos anteriores. O principal efeito do boosting é a redução do viés em comparação com o modelo base. Assim, o boosting tem a capacidade de transformar um conjunto de modelos de base fracos em um modelo de conjunto forte. (SCHAPIRE; FREUND, 2012)

A técnica de Boosting é utilizada em Machine Learning com o objetivo de melhorar a acurácia dos algoritmos de classificação. Essa técnica tem sido usada nos problemas de regressão e consiste em efetuar diversas execuções de um algoritmo de aprendizagem básico, modificando a distribuição de pesos no conjunto de treinamento e combinando linearmente os preditores obtidos em um único preditor mais eficiente.

A primeira proposta do algoritmo foi apresentada por Schapire em 1990 para resolver problemas de classificação com duas classes. Em 1996, Freund e Schapire propuseram uma versão do algoritmo chamada de AdaBoost, em que o algoritmo ajusta um modelo logístico aditivo em que o número de iterações é o número de funções usadas na representação aditiva que será utilizada para aproximar a função estimada. O algoritmo gera a cada passo uma distribuição em relação às observações da amostra, atribuindo um peso maior às observações classificadas incorretamente no passo anterior.

O algoritmo AdaBoost utiliza como entrada um conjunto de treinamento (x_1, y_1), ..., (x_m, y_m) em que cada x_i é um vetor de características, pertencente a algum espaço X e y_i pertence ao conjunto Y = $\{-1, +1\}$, ou seja, o algoritmo é utilizado para problemas de classificação binários. O Adaboost considera um classificador base, repetidamente, num conjunto de T execuções, em que t = 1, ..., T. Uma das principais ideias do algoritmo é modificar a distribuição, ou conjunto de pesos, sobre o conjunto de treinamento. O peso dessa distribuição no treinamento i na execução t é denotado por $D_t(i)$. Inicialmente os pesos são todos iguais, mas a cada execução, os pesos das execuções classificadas incorretamente são incrementados de forma que o classificador base atue com maior intensidade sobre essas execuções no conjunto de treinamento, a taxa de erro é calculada

contando o número de classificações incorretas, conforme abaixo:

$$e_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \tag{3.1}$$

É possível verificar que o erro é calculado de acordo com a distribuição D_t sobre a qual o classificador fraco foi treinado.

Uma vez que a hipótese h_t tenha sido recebida (obtida na iteração t do algoritmo base), o AdaBoost escolhe um parâmetro α_t pertencente ao conjunto dos números reais, o qual mede a importância que é dada à hipótese h_t. Para formular a hipótese final H, cada hipótese h_t contribui com uma certa confiança dada por α_t.

Algoritmo Adaboost:

Seja

$$S = \{(x_1, y_1), ..., (x_m, y_m); x_i \in X, y_i \in \{-1, +1\}\} \tag{3.2}$$

Inicialize

$$D_1(i) := \frac{1}{m} \forall (x_i, y_i) \in S \tag{3.3}$$

Para t = 1, ..., T treine o classificador base usando a distribuição D_t e obtenha a hipótese fraca $h_t : X \to \{-1, +1\}$. Calcule

$$\alpha_t : \alpha_t = \frac{1}{2} ln(\frac{1 - e_t}{e_t}) \tag{3.4}$$

onde e_t é a taxa de erro do classificador h_t.

Atualize a distribuição:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} = \begin{cases} e^{-\alpha_t}, \text{se } h_t(x_i) = y_i \\ e^{\alpha_t}, \text{se } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t} \tag{3.5}$$

em que Z_t é o fator de normalização.

Saída: Hipótese Final:

$$H(x) = sign(\sum_{1,...,T} \alpha_t * h_t(x)) \tag{3.6}$$

Outras variações do AdaBoost são:

- Real AdaBoost, em que o algoritmo utiliza a probabilidade de classe estimada para construir previsões com valor real;
- LogitBoost, em que o algoritmo utiliza as etapas de Newton para ajustar um modelo logístico aditivo simétrico pela máxima verossimilhança;
- Gentle AdaBoost, uma versão modificada do Real AdaBoost, em que o algoritmo utiliza as etapas de Newton no lugar das otimizações exatas a cada passo.

3.2 Métricas do Poder Preditivo

As fórmulas para cada uma das métricas[2] serão apresentadas a partir da matriz de confusão, em que TP é o número de verdadeiros positivos, TN é o número de verdadeiros negativos, FP é o número de falsos positivos e FN é o número de falsos negativos.

- Coeficiente de Correlação de Matthews (MCC): Leva em consideração verdadeiros e falsos positivos e negativos e geralmente é considerada uma medida equilibrada que pode ser usada mesmo se as classes forem de tamanhos muito diferentes. O MCC é basicamente um valor de coeficiente de correlação entre -1 e +1. Um coeficiente de +1 representa uma predição perfeita, 0 uma predição aleatória média e -1 uma predição inversa.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \tag{3.7}$$

  Se qualquer uma das somas do denominador for igual a zero, o denominador é definido como igual a 1 e isso resulta em um Coeficiente de Correlação de Matthews (MCC) igual a 0.

- Acurácia Balanceada: A acurácia balanceada em problemas de classificação é capaz de lidar com conjuntos de dados desequilibrados e é definida como a média de evocação obtida em cada classe. No caso binário, a acurácia balanceada é igual à média aritmética da sensibilidade (verdadeiros positivos) e da especificidade (verdadeiros negativos), ou a área sob a curva ROC.

$$AcuráciaBalanceada = \frac{1}{2}(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}) \tag{3.8}$$

- Precisão: É a capacidade do classificador de não rotular como positiva uma amostra negativa.

$$Precisão = \frac{TP}{TP + FP} \tag{3.9}$$

[2] Informações sobre Métricas do Poder Preditivo: https://scikit-learn.org/stable/modules/model$_e valuation.html$

- Revocação: É a capacidade do classificador de encontrar todas as amostras positivas.

$$Revocação = \frac{TP}{TP + FN} \tag{3.10}$$

- Medida F1: A medida F1 pode ser interpretada como uma média ponderada da precisão e da revocação, em que uma pontuação F1 atinge seu melhor valor em 1 e seu pior resultado em 0. A contribuição relativa da precisão e da recuperação para a medida F1 são iguais.

$$MedidaF1 = 2 * \frac{Precisão * Revocação}{Precisão + Revocação} \tag{3.11}$$

- Pontuação de Precisão Média: Calcula a precisão média das pontuações de predição. O valor está entre 0 e 1 e quanto maior melhor.

$$PontuaçãodePrecisãoMédia = \sum_n (Revocação_n - Revocação_{n-1}) * Precisão_n \tag{3.12}$$

em que a precisão e a revocação estão no enésimo limite. Com previsões aleatórias, a Pontuação de Precisão Média é a fração das amostras positivas.

- Curva ROC[3]: Área sob a curva das características operacionais do receptor. É um gráfico que ilustra a capacidade de diagnóstico de um sistema classificador binário conforme o limite de discriminação varia. A curva ROC mostra a relação entre a taxa de verdadeiros positivos (revocação) e a taxa de falsos positivos (taxa de casos em que a categoria 0 foi incorretamente classificada como sendo da categoria alvo).

- Cohen's kappa[4]: Calcula a estatística Cohen's kappa. A pontuação kappa é um número entre -1 e 1 e expressa o nível de concordância entre dois anotadores em um problema de classificação. Pontuações acima de 0,8 são geralmente consideradas de boa concordância; zero ou menos significa nenhum acordo (rótulos praticamente aleatórios).

[3] Informações sobre Curva ROC: https://medium.com/bio-data-blog/entenda-o-que-é-auc-e-roc-nos-modelos-de-machine-learning-8191fb4df772

[4] Informações sobre Cohen's kappa: https://thenewstack.io/cohens-kappa-what-it-is-when-to-use-it-and-how-to-avoid-its-pitfalls/

4 RESULTADOS E DISCUSSÕES

4.1 Análise Descritiva dos Dados

Como somente a partir do Enem 2015 foi possível distinguir os participantes que fazem o Enem apenas para autoavaliação dos que fazem o exame com o objetivo de ingressar em um curso de nível superior, o panorama histórico será feito desde 2015.

O número total de inscritos no Enem de 2015 a 2019 pode ser visto na Tabela 3:

Tabela 3: Número total de inscritos no Enem de 2015 a 2019

Ano do Enem	Quantidade de Inscritos
2015	7.746.427
2016	8.627.367
2017	6.731.341
2018	5.513.747
2019	5.095.270

Fonte: Elaborada pela autora.

O arquivo de microdados do Enem 2019 é composto por 136 variáveis divididas em 9 categorias de informações, são elas: dados do participante, dados da escola, dados dos pedidos de atendimento especializado, dados dos pedidos de atendimento específico, dados dos pedidos de recursos especializados e específicos para realização das provas, dados do local de aplicação da prova, dados da prova objetiva, dados da redação e dados do questionário socioeconômico.

As variáveis utilizadas para definir o grupo de interesse foram as seguintes:

- Dados do participante: Indica se o inscrito fez a prova com intuito de apenas treinar seus conhecimentos.

- Dados da prova objetiva: Indica se o inscrito estava presente na prova objetiva de Ciências da Natureza, Indica se o inscrito estava presente na prova objetiva de Ciências Humanas, Indica se o inscrito estava presente na prova objetiva de Linguagens e Códigos, Indica se o inscrito estava presente na prova objetiva de Matemática.

- Dados da redação: Indica a situação da redação do participante.

O grupo de interesse foi composto pelos inscritos que não eram treineiros, que estavam presentes em todas as provas e cuja redação foi considerada sem problemas. Considerando esses critérios, o número total de participantes em cada um dos anos pode ser visto na Tabela 4:

Tabela 4: Número total de participantes no Enem de 2015 a 2019 que compõem o grupo de interesse

Ano do Enem	Quantidade de Participantes
2015	4.768.463
2016	4.868.906
2017	3.755.174
2018	3.364.746
2019	3.089.609

Fonte: Elaborada pela autora.

Na Tabela 5 é possível observar o desempenho do grupo de interesse no Enem 2015(INEP, a):

Tabela 5: Desempenho do grupo de interesse no Enem 2015

Prova	Média	1° Quartil	Mediana	3° Quartil	Nota Mínima	Nota Máxima
Ciências da Natureza	480,4	427,8	472,9	523,9	0,0	875,2
Ciências Humanas	560,7	516,5	566,1	608,8	0,0	850,6
Linguagens e Códigos	506,4	459,9	509,6	556,0	0,0	825,8
Matemática	467,2	388,6	444,5	520,1	0,0	1008,3
Redação	542,7	460,0	540,0	600,0	0,0	1000,0

Fonte: Elaborada pela autora.

Na Tabela 6 é possível observar o desempenho do grupo de interesse no Enem 2016(INEP, b):

Tabela 6: Desempenho do grupo de interesse no Enem 2016

Prova	Média	1° Quartil	Mediana	3° Quartil	Nota Mínima	Nota Máxima
Ciências da Natureza	479,4	423,2	464,6	524,4	0,0	871,3
Ciências Humanas	537,5	488,3	540,0	588,4	0,0	859,1
Linguagens e Códigos	522,2	476,2	526,2	569,9	0,0	802,6
Matemática	491,1	413,6	469,0	549,9	0,0	991,5
Redação	542,3	460,0	540,0	620,0	0,0	1000,0

Fonte: Elaborada pela autora.

Na Tabela 7 é possível observar o desempenho do grupo de interesse no Enem 2017(INEP, c):

Tabela 7: Desempenho do grupo de interesse no Enem 2017

Prova	Média	1° Quartil	Mediana	3° Quartil	Nota Mínima	Nota Máxima
Ciências da Natureza	512,3	456,0	507,6	563,0	0,0	885,6
Ciências Humanas	524,0	462,2	526,9	584,6	0,0	868,3
Linguagens e Códigos	514,6	474,1	518,6	559,4	0,0	788,8
Matemática	520,2	437,2	504,3	586,5	0,0	993,9
Redação	560,3	480,0	560,0	640,0	40,0	1000,0

Fonte: Elaborada pela autora.

Na Tabela 8 é possível observar o desempenho do grupo de interesse no Enem 2018(INEP, d):

Tabela 8: Desempenho do grupo de interesse no Enem 2018

Prova	Média	1° Quartil	Mediana	3° Quartil	Nota Mínima	Nota Máxima
Ciências da Natureza	494,2	436,1	483,6	542,5	0,0	869,6
Ciências Humanas	571,5	514,4	585,2	629,8	0,0	850,4
Linguagens e Códigos	528,4	478,0	532,8	580,6	0,0	816,9
Matemática	434,4	454,6	515,3	598,5	0,0	996,1
Redação	524,7	360,0	520,0	640,0	40,0	1000,0

Fonte: Elaborada pela autora.

Na Tabela 9 é possível observar o desempenho do grupo de interesse no Enem 2019(INEP, e):

Tabela 9: Desempenho do grupo de interesse no Enem 2019

Prova	Média	1° Quartil	Mediana	3° Quartil	Nota Mínima	Nota Máxima
Ciências da Natureza	478,5	418,7	470,7	532,9	0,0	860,9
Ciências Humanas	511,5	452,9	514,7	569,4	0,0	835,1
Linguagens e Códigos	523,7	487,2	528,2	566,9	0,0	801,7
Matemática	522,7	434,6	500,0	596,7	0,0	985,5
Redação	596,1	500,0	580,0	680,0	40,0	1000,0

Fonte: Elaborada pela autora.

Na Tabela 10 é possível observar o desempenho médio do grupo de interesse no Enem de 2015 a 2019, ou seja, o resultado obtido calculando-se a média aritmética simples das provas de Ciências da Natureza, de Ciências Humanas, de Linguagens e Códigos, de Matemática e da redação:

Tabela 10: Desempenho médio do grupo de interesse

Enem	Média	1° Quartil	Mediana	3° Quartil	Nota Mínima	Nota Máxima
2015	511,5	461,7	502,2	551,6	52,0	866,5
2016	514,5	463,5	503,8	554,9	36,0	857,6
2017	526,3	473,8	517,7	569,9	40,0	855,2
2018	530,6	469,2	520,2	580,6	44,0	858,2
2019	526,5	467,5	516,0	576,9	44,0	850,8

Fonte: Elaborada pela autora.

A partir dessas informações da média das notas obtidas ano a ano, é possível observar que o desempenho dos participantes do Enem melhorou de 2015 a 2018, já no Enem 2019, a nota dos participantes baixou um pouco em relação ao desempenho obtido no exame do ano anterior, é importante ressaltar que esses dados levam em consideração apenas os inscritos que não eram treineiros, que estavam presentes em todas as provas e cuja redação foi considerada sem problemas.

4.2 Pré-Processamento de Dados

A base de dados inicial apresenta 48 variáveis e 3.089.609 registros. As variáveis categóricas foram transformadas em variáveis numéricas por meio do label encoding, em que cada categoria da variável é renomeada por um número. Após realizar essa transformação nas variáveis, as que eram originalmente do tipo caractere foram excluídas da base juntamente com a variável que identifica cada um dos participantes do Enem, o que resultou em uma base de dados com 45 atributos.

A base de dados foi dividida em treino e teste, na proporção 80%/20%, ou seja, a base de treinamento tem 2.471.687 registros e a base de teste tem 617.922 registros. Como os dados são desbalanceados, a divisão dos dados em treino e teste foi feita de forma estratificada, ou seja, conservando a mesma proporção de candidatos que seriam aprovados em Medicina da base original.

Como a variável resposta é construída a partir das variáveis da nota média do participante nas provas objetivas e de Redação e da menor nota de corte no curso de Medicina, esses atributos também não serão utilizados na construção dos modelos, visto que estão diretamente ligados à variável resposta.

Observando o mapa de calor da Figura 1, é possível perceber que os atributos que estão mais correlacionados à variável resposta são as notas obtidas pelos participantes do Enem.

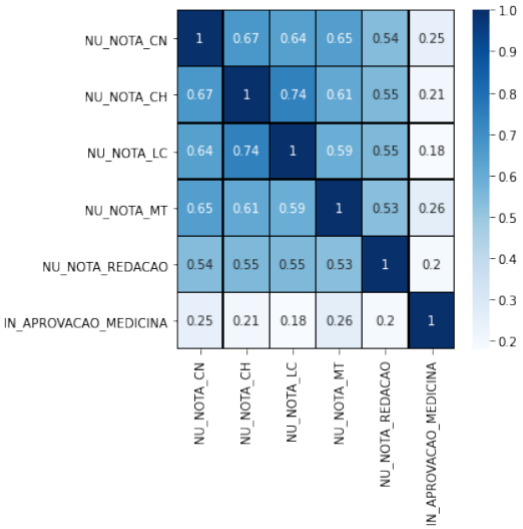

Figura 1: Atributos que estão mais correlacionados à variável resposta

Fonte: Elaborada pela autora.

4.3 Modelos de Machine Learning

4.3.1 Random Forest

Foi construído o modelo Random Forest[1] utilizando o critério Gini, ou seja, a função que mede a qualidade de uma divisão utiliza a impureza de Gini, variando o número de árvores de 1 a 500 de 50 em 50. A acurácia obtida por cada um desses modelos no conjunto de teste pode ser observada na Tabela 11:

[1] Informações sobre Random Forest: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.Random

Tabela 11: Acurácia dos modelos Random Forest variando a quantidade de árvores

Quantidade de Estimadores	Acurácia
1 árvore	0,99649
51 árvores	0,99876
101 árvores	0,99888
151 árvores	0,99882
201 árvores	0,99883
251 árvores	0,99884
301 árvores	0,99888
351 árvores	0,99884
401 árvores	0,99888
451 árvores	0,99883

Fonte: Elaborada pela autora.

A melhor acurácia foi obtida quando o número de árvores foi igual a 101. Esse modelo foi construído a partir dos seguintes parâmetros, alguns deles tem como objetivo aumentar o poder preditivo do modelo:

- Número de estimadores: A quantidade de árvores no modelo foi 101, esse parâmetro indica o número de árvores construídas pelo algoritmo antes de tomar uma votação. Em geral, uma quantidade elevada de árvores aumenta a performance e torna as predições mais estáveis, mas também torna a computação mais lenta;

- Técnica de poda: A profundidade máxima das árvores não foi definida. Assim, os nós foram expandidos até que todas as folhas fossem puras ou até que todas as folhas contivessem menos de 2 amostras;

- Técnica de partição: O número mínimo de amostras necessárias para dividir um nó interno foi de 2 amostras. O algoritmo de floresta aleatória adiciona aleatoriedade extra ao modelo quando está criando as árvores. No lugar de procurar pela melhor característica ao fazer a partição dos nós, ele busca a melhor característica em um subconjunto aleatório das características. Este processo cria uma grande diversidade, o que geralmente leva a geração de modelos melhores;

- Número de atributos na construção da árvore: Indica o número máximo de características a serem utilizadas pelo Random Forest na construção de uma dada árvore ao procurar a melhor divisão. No caso, o número máximo de atributos foi de 7

(arredondando, a base apresenta 42 variáveis). É importante ressaltar que a busca por uma divisão não para até que pelo menos uma partição válida das amostras de nó seja encontrada, mesmo que seja necessário inspecionar efetivamente mais que o número máximo de características consideradas.

- Número de amostras no nó folha: Indica o número mínimo de amostras necessárias para estar em um nó folha. Um ponto de divisão em qualquer profundidade só será considerado se deixar pelo menos uma amostra de treinamento em cada um dos ramos, esquerdo e direito.

- Poda de custo-complexidade mínima: A poda de custo-complexidade mínima é um algoritmo usado para podar uma árvore para evitar o sobreajuste. A subárvore com a maior complexidade de custo menor do que 0 será escolhida. Por padrão, nenhuma poda é executada.

- Regra de classificação: A classe prevista de uma amostra de entrada foi o voto das árvores na floresta ponderado por suas estimativas de probabilidade. Ou seja, a classe prevista foi aquela com estimativa de probabilidade média mais alta entre as árvores.

O Random Forest é um algoritmo muito acessível, pois seus hiperparâmetros com os valores padrão já produzem um bom resultado de predição. O número de hiperparâmetros não é tão grande e são de fácil compreensão.

Um dos problemas em Aprendizado de Máquina é o overfitting, mas, na grande maioria das vezes, isso não ocorrerá tão facilmente com o classificador Floresta Aleatória. Uma vez que, se há árvores suficientes na floresta, o classificador não irá apresentar um sobreajuste para o modelo.

A maior limitação do Floresta Aleatória é que uma quantidade grande de árvores pode tornar o algoritmo lento e ineficiente para predições em tempo real. Em geral, esses algoritmos são rápidos para treinar, mas muito lentos para fazer predições após o treinamento. Uma predição com uma maior acurácia requer mais árvores, o que faz o modelo ficar mais lento. Em muitas aplicações do mundo real, o Random Forest é rápido o suficiente, mas pode haver situações em que a performance em tempo de execução é importante e outras abordagens são mais adequadas.

Como o conjunto de dados é desbalanceado, outras medidas de avaliação do modelo também serão consideradas para verificar se o Random Forest com 101 árvores se ajusta bem aos dados de teste. Essas medidas de avaliação podem ser vistas na Tabela 12:

Tabela 12: Métricas de avaliação para o modelo Random Forest com 101 árvores

Métrica	Valor
Coeficiente de Correlação de Matthews (MCC)	0,92714
Acurácia Balanceada	0,93519
Medida F1	0,96262
Curva ROC	0,93519
Cohen's kappa	0,92524
Precisão	0,99380
Revocação	0,93519
Pontuação de Precisão Média	0,86174

Fonte: Elaborada pela autora.

Para o modelo Random Forest com 101 árvores, tem-se os seguintes valores:

- Verdadeiro negativo (na base de teste o registro foi classificado como não aprovado em Medicina e o modelo fez a previsão idêntica ao que está na base de teste): 612.636 registros;
- Falso positivo (na base de teste o registro foi classificado como não aprovado em Medicina e o modelo fez a previsão diferente do que está na base de teste): 52 registros;
- Falso negativo (na base de teste o registro foi classificado como aprovado em Medicina e o modelo fez a previsão diferente do que está na base de teste): 678 registros;
- Verdadeiro positivo (na base de teste o registro foi classificado como aprovado em Medicina e o modelo fez a previsão idêntica ao que está na base de teste): 4.556 registros.

Com base em todas essas informações, é possível concluir que o modelo Random Forest com 101 árvores e critério de Gini se ajusta muito bem aos dados de teste.

Na Figura 2, é possível perceber que as variáveis que tem a maior importância relativa para esse modelo Random Forest com 101 árvores são:

- Nota na prova de Ciências da Natureza;
- Nota na prova de Matemática;
- Nota na prova de Redação;

- Nota na prova de Ciências Humanas;
- Nota na prova de Linguagens e Códigos.

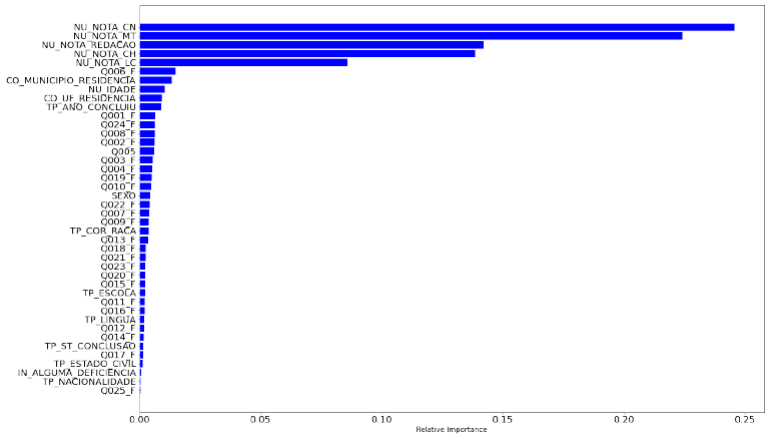

Figura 2: Importância relativa das variáveis do Random Forest com 101 árvores

Fonte: Elaborada pela autora.

Como o modelo Random Forest com 101 árvores teve um excelente desempenho no conjunto de teste, será realizada um verificação com o objetivo de observar se um modelo mais simples que o Random Forest, no caso, a Árvore de Decisão, também apresenta um bom desempenho para o conjunto de teste.

4.3.2 Árvore de Decisão

A árvove de decisão contruída pelo critério de Gini apresentou uma acurácia de 0,99861. Esse modelo foi construído a partir dos seguintes parâmetros:

- Técnica de poda: A profundidade máxima da árvore não foi definida. Assim, os nós foram expandidos até que todas as folhas fossem puras ou até que todas as folhas contivessem menos de 2 amostras;
- Técnica de partição: O número mínimo de amostras necessárias para dividir um nó interno foi de 2 amostras;
- Número de atributos na construção da árvore: Indica o número máximo de características a serem utilizadas pela árvore de decisão na construção de uma dada árvore ao procurar a melhor divisão. No caso, o número máximo de atributos foi de 7 (arredondando, a base apresenta 42 variáveis). É importante ressaltar que a busca por uma divisão não para até que pelo menos uma partição válida das amostras de nó seja encontrada, mesmo que seja necessário inspecionar efetivamente mais que o número máximo de características consideradas.

- Número de amostras no nó folha: Indica o número mínimo de amostras necessárias para estar em um nó folha. Um ponto de divisão em qualquer profundidade só será considerado se deixar pelo menos uma amostra de treinamento em cada um dos ramos, esquerdo e direito.

- Poda de custo-complexidade mínima: A poda de custo-complexidade mínima é um algoritmo usado para podar uma árvore para evitar o sobreajuste. A subárvore com a maior complexidade de custo menor do que 0 será escolhida. Por padrão, nenhuma poda é executada.

- Regra de classificação: A classe prevista para cada amostra de entrada é retornada pelo modelo.

Como o conjunto de dados é desbalanceado, outras medidas de avaliação do modelo também serão consideradas para verificar se a árvore de decisão se ajusta bem aos dados de teste. Essas medidas de avaliação podem ser vistas na Tabela 13:

Tabela 13: Métricas de avaliação para o modelo Árvore de Decisão

Métrica	Valor
Coeficiente de Correlação de Matthews (MCC)	0,91739
Acurácia Balanceada	0,95952
Medida F1	0,95869
Curva ROC	0,95952
Cohen's kappa	0,91739
Precisão	0,95787
Revocação	0,95952
Pontuação de Precisão Média	0,84357

Fonte: Elaborada pela autora.

Para a árvore de decisão, tem-se os seguintes valores:

- Verdadeiro negativo: 612.249 registros;

- Falso positivo: 439 registros;

- Falso negativo: 420 registros;

- Verdadeiro positivo: 4.814 registros.

Com base em todas essas informações, é possível perceber que o modelo de árvore de decisão erra mais no geral, visto que a soma dos falsos positivos e dos falsos negativos é superior a essa mesma soma no Random Forest com 101 árvores. Entretanto, também é interessante perceber que ela classifica menos registros como falso negativo, logo classifica mais participantes como verdadeiros positivos.

Como a árvore de decisão é um modelo mais simples que o Random Forest, já era esperado que o seu desempenho fosse pior. Entretanto, por ser um modelo de implementação mais rápida, quando se tem um conjunto de dados com mais de 3 milhões de registros, a árvore de decisão pode ser uma opção, visto que para este caso, o seu desempenho não pode ser considerado ruim.

4.3.3 Boosting

4.3.3.1 Adaboost com estimador base Random Forest

Foi construído o modelo Adaboost[2] com estimador base Random Forest, que tem como objetivo reduzir o viés do modelo, assumindo a hipótese de que um conjunto de modelos fracos é capaz de produzir modelos fortes por meio de uma combinação sequencial. A cada iteração, introduz um classificador e pondera cada exemplo do conjunto de dados completo pelo desempenho do classificador base, quanto mais difícil de ser aprendido, maior o peso associado ao exemplo e maior a probabilidade de ser escolhido na próxima iteração. A acurácia de 0,99884 foi obtida utilizando os seguintes parâmetros, alguns deles tem como objetivo aumentar o poder preditivo do modelo:

- Estimador base: O classificador base utilizado na construção do boosted ensemble foi o Random Forest com 10 árvores.
- Número de estimadores: O número máximo de classificadores no modelo foi 10. Em geral, uma quantidade elevada de estimadores aumenta a performance e torna as predições mais estáveis, mas também torna a computação mais lenta;
- Taxa de apredizagem: A taxa de aprendizagem utilizada foi de 0,01 e ela reduz a contribuição de cada classificador. Há uma compensação entre a taxa de aprendizado e o número de estimadores.
- Algoritmo: Foi utilizado o algoritmo real de boosting.

Uma das grandes vantagens de utilizar o Boosting é que a convergência é rápida e ele foca em exemplos difíceis de serem classificados, sendo pouco indicado para dados com ruídos e pequenos conjuntos de dados.

[2] Informações sobre Adaboost: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClas

Como o conjunto de dados é desbalanceado, outras medidas de avaliação do modelo também serão consideradas para verificar se o modelo de classificação Adaboost com estimador base Random Forest se ajusta bem aos dados de teste. Essas medidas de avaliação podem ser vistas na Tabela 14:

Tabela 14: Métricas de avaliação para o modelo de classificação Adaboost com estimador base Random Forest

Métrica	Valor
Coeficiente de Correlação de Matthews (MCC)	0,92868
Acurácia Balanceada	0,93605
Medida F1	0,96340
Curva ROC	0,93605
Cohen's kappa	0,92680
Precisão	0,99447
Revocação	0,93605
Pontuação de Precisão Média	0,86456

Fonte: Elaborada pela autora.

Para o modelo de classificação Adaboost com estimador base Random Forest, tem-se os seguintes valores:

- Verdadeiro negativo: 612.642 registros;
- Falso positivo: 46 registros;
- Falso negativo: 669 registros;
- Verdadeiro positivo: 4.565 registros.

Com base em todas essas informações, é possível concluir que o modelo de classificação Adaboost com estimador base Random Forest se ajusta muito bem aos dados de teste.

Como o modelo de classificação Adaboost com estimador base Random Forest teve um excelente desempenho no conjunto de teste, será realizada um verificação com o objetivo de observar se um modelo mais simples, no caso, o Adaboost com estimador base árvore de decisão, também apresenta um bom desempenho para o conjunto de teste.

4.3.3.2 Adaboost com estimador base Árvore de Decisão

Foi construído o modelo Adaboost com estimador base árvore de decisão. Esse modelo foi estruturado a partir dos seguintes parâmetros e apresentou uma acurácia de 0,99884, mesma acurácia obtida no modelo anterior, o Adaboost com estimador base Random Forest:

- Estimador base: O classificador base utilizado na construção do boosted ensemble foi a árvore de decisão.
- Número de estimadores: O número máximo de classificadores no modelo foi 1.000. Em geral, uma quantidade elevada de estimadores aumenta a performance e torna as predições mais estáveis, mas também torna a computação mais lenta;
- Taxa de apredizagem: A taxa de aprendizagem utilizada foi de 0,01 e ela reduz a contribuição de cada classificador. Há uma compensação entre a taxa de aprendizado e o número de estimadores.
- Algoritmo: Foi utilizado o algoritmo real de boosting.

Como o conjunto de dados é desbalanceado, outras medidas de avaliação do modelo também serão consideradas para verificar se o modelo de classificação Adaboost com estimador base árvore de decisão se ajusta bem aos dados de teste. Essas medidas de avaliação podem ser vistas na Tabela 15:

Tabela 15: Métricas de avaliação para o modelo de classificação Adaboost com estimador base Árvore de Decisão

Métrica	Valor
Coeficiente de Correlação de Matthews (MCC)	0,92008
Acurácia Balanceada	0,92945
Medida F1	0,95892
Curva ROC	0,92945
Cohen's kappa	0,91784
Precisão	0,99281
Revocação	0,92945
Pontuação de Precisão Média	0,84888

Fonte: Elaborada pela autora.

Para o modelo de classificação Adaboost com estimador base árvore de decisão, tem-se os seguintes valores:

- Verdadeiro negativo: 612.628 registros;
- Falso positivo: 60 registros;
- Falso negativo: 738 registros;
- Verdadeiro positivo: 4.496 registros.

Com base em todas essas informações, é possível perceber que o modelo Adaboost com estimador base árvore de decisão erra mais no geral, visto que a soma dos falsos positivos e dos falsos negativos é superior a essa mesma soma no modelo Adaboost com estimador base Random Forest.

Como o modelo Adaboost com estimador base árvore de decisão é um modelo mais simples que o modelo Adaboost com estimador base Random Forest já era esperado que o seu desempenho fosse pior. Entretanto, apesar de ser mais simples, sua implementação não foi mais rápida, visto que, para ter um desempenho próximo ao que o modelo Adaboost com estimador base Random Forest teve, foi necessário utilizar 1.000 estimadores, 100 vezes mais do que foi utilizado no modelo Adaboost com estimador base Random Forest.

4.4 Trabalhos Futuros

Os objetivos nos trabalhos futuros são:

- Aprimorar os modelos desenvolvidos neste trabalho, visto que como a base de dados é muito grande, o tempo de treinamento dos modelos foi considerável, e, por isso, não foi possível testar os modelos fazendo muitas alterações nos parâmetros de entrada. Assim, a intenção é que futuramente sejam construídos modelos em que seja realizado o Grid Search em vários hiperparâmetros do modelo;
- Desenvolver um modelo utilizando Boosting com Regressão Logística para esses dados;
- Expandir a construção dos modelos, utilizando dados de outros anos, o que possibilitaria analisar se todos os anos apresentam o mesmo comportamento ou se existem diferenças significativas em relação à aprovação em Medicina dos alunos que fizeram a prova do Enem.

5 CONCLUSÃO

Um excelente desempenho na predição para os dados de teste foi obtido pelo modelo Adaboost com estimador base Random Forest, pois foi possível verificar com uma precisão alta qual participante do Enem 2019 seria aprovado ou não para o curso de Medicina na universidade com a menor nota de corte nacional considerando o desempenho obtido por ele no SiSU 2020. Para os dados de teste, o Coeficiente de Correlação de Matthews (MCC) foi igual a 0,92868.

O modelo Random Forest com 101 árvores teve um desempenho próximo, mas um pouco inferior ao do modelo anterior, visto que o Coeficiente de Correlação de Matthews (MCC) foi igual a 0,92714. O Random Forest possibilita verificar quais atributos apresentam maior importância relativa para o modelo e, nesse caso, as variáveis que tiveram a maior importância relativa foram, respectivamente, Nota na prova de Ciências da Natureza, Nota na prova de Matemática, Nota na prova de Redação, Nota na prova de Ciências Humanas e Nota na prova de Linguagens e Códigos.

O modelo Adaboost com estimador base Árvore de Decisão não teve um desempenho muito inferior, visto que o Coeficiente de Correlação de Matthews (MCC) foi igual a 0,92008. Entretanto, para que tivesse esse desempenho, foi necessário utilizar 100 vezes mais estimadores do que o foi utilizado no modelo Adaboost com estimador base Random Forest.

O modelo Árvore de Decisão apesar de ser bem mais simples que o Random Forest não teve um desempenho muito inferior, visto que o Coeficiente de Correlação de Matthews (MCC) foi igual a 0,91739. Sendo assim, por ser um modelo de implementação mais fácil, quando se tem um conjunto de dados com muitos registros, a árvore de decisão pode ser uma alternativa, quando seu desempenho for próximo ao que o Random Forest fornece.

Assim, é possível concluir que para uma base de dados com mais de 3 milhões de registros, todos os algoritmos utilizados obtiveram um resultado satisfatório, visto que em todos os casos o Coeficiente de Correlação de Matthews (MCC) foi superior a 0,90.

REFERÊNCIAS

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. **Proceedings of the 20th International Conference on Very Large Data Bases**, p. 487–499, 1994.

BREIMAN, L. Random forests. **Machine learning**, p. 5–32, 2001.

INEP. **Microdados do Enem 2015**. Disponível em: <http://portal.inep.gov.br/web/guest/microdados>. Acesso em: 11/06/2020.

_____. **Microdados do Enem 2016**. Disponível em: <http://portal.inep.gov.br/web/guest/microdados>. Acesso em: 11/06/2020.

_____. **Microdados do Enem 2017**. Disponível em: <http://portal.inep.gov.br/web/guest/microdados>. Acesso em: 11/06/2020.

_____. **Microdados do Enem 2018**. Disponível em: <http://portal.inep.gov.br/web/guest/microdados>. Acesso em: 11/06/2020.

_____. **Microdados do Enem 2019**. Disponível em: <http://portal.inep.gov.br/web/guest/microdados>. Acesso em: 27/06/2020.

MARCIANO, C. E.; DAVIM, J. V. Modelos preditivos de notas de redação do enem 2015. **Universidade Federal do Rio de Janeiro**, 2017.

MARTINS, C.; FARIA, D. E. de. **A análise do perfil socioeconômico dos candidatos ao Exame Nacional do Ensino Médio (ENEM) do estado de Santa Catarina**. 2010. Monografia (Graduação) — Universidade do Sul do Estado de Santa Catarina (UNISUL), Palhoça, 2010.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

SAFAVIAN, S. R.; LANDGREBE, D. A survey of decision tree classifier methodology. **IEEE Trans. Systems, Man and Cybernetics**, XXI, p. 660–674, 1991.

SCHAPIRE, R. E.; FREUND, Y. **Boosting - Foundations and Algorithms**. [S.l.]: Adaptive Computation and Machine Learning series, 2012.

SILVA, L. A.; MORINO, A. H.; SATO, T. M. C. Prática de mineração de dados no exame nacional do ensino médio. **Anais dos Workshops do Congresso Brasileiro de Informática na Educação**, III, p. 651–660, 2014.

STEARNS, B. et al. Prevendo desempenho dos candidatos do enem através de dados socioeconômicos. **Anais do Congresso da Sociedade Brasileira de Computação**, p. 2522–2530, 2014.

Apêndices

APÊNDICE A – PACOTES EM PYTHON UTILIZADOS NO TCC

Para fazer a análise dos dados e construir os modelos de Machine Learning foram utilizados os seguintes pacotes do Python:

- Pandas versão 1.1.3;
- Numpy versão 1.19.2;
- Seaborn versão 0.11.0;
- Matplotlib versão 3.3.2: Pyplot;
- Scikit-learn (PEDREGOSA et al., 2011) versão 0.23.2: Preprocessing, Model_selection, Ensemble, Metrics e Tree.

APÊNDICE B – PROGRAMAÇÃO EM PYTHON - CÓDIGOS

```
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import mutual_info_classif
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import make_scorer, matthews_corrcoef,
confusion_matrix, classification_report, cohen_kappa_score,
accuracy_score, average_precision_score, roc_auc_score

# Carregando o arquivo "MICRODADOS_FINAL.csv" e armazenando
# em um pandas DataFrame
dados = pd.read_csv('MICRODADOS_FINAL.csv', sep=';', encoding='latin-1')
dados.head()

print("Numero de linhas e colunas na matriz de atributos:", dados.shape)

# Verificando o tipo de dado de cada coluna do arquivo
dados.dtypes

label_encoder = LabelEncoder()

# Atribuicao do rotulo a variavel TP_SEXO
dados['SEXO'] = label_encoder.fit_transform(dados['TP_SEXO'])
dados['SEXO'].value_counts()

# Atribuicao do rotulo a variavel Q001
dados['Q001_F'] = label_encoder.fit_transform(dados['Q001'])
dados['Q001_F'].value_counts()
```

```
# Atribuicao do rotulo a variavel Q002
dados['Q002_F'] = label_encoder.fit_transform(dados['Q002'])
dados['Q002_F'].value_counts()

# Atribuicao do rotulo a variavel Q003
dados['Q003_F'] = label_encoder.fit_transform(dados['Q003'])
dados['Q003_F'].value_counts()

# Atribuicao do rotulo a variavel Q004
dados['Q004_F'] = label_encoder.fit_transform(dados['Q004'])
dados['Q004_F'].value_counts()

# Atribuicao do rotulo a variavel Q006
dados['Q006_F'] = label_encoder.fit_transform(dados['Q006'])
dados['Q006_F'].value_counts()

# Atribuicao do rotulo a variavel Q007
dados['Q007_F'] = label_encoder.fit_transform(dados['Q007'])
dados['Q007_F'].value_counts()

# Atribuicao do rotulo a variavel Q008
dados['Q008_F'] = label_encoder.fit_transform(dados['Q008'])
dados['Q008_F'].value_counts()

# Atribuicao do rotulo a variavel Q009
dados['Q009_F'] = label_encoder.fit_transform(dados['Q009'])
dados['Q009_F'].value_counts()

# Atribuicao do rotulo a variavel Q010
dados['Q010_F'] = label_encoder.fit_transform(dados['Q010'])
dados['Q010_F'].value_counts()

# Atribuicao do rotulo a variavel Q011
dados['Q011_F'] = label_encoder.fit_transform(dados['Q011'])
dados['Q011_F'].value_counts()

# Atribuicao do rotulo a variavel Q012
dados['Q012_F'] = label_encoder.fit_transform(dados['Q012'])
```

```
dados['Q012_F'].value\_counts()

# Atribuicao do rotulo a variavel Q013
dados['Q013_F'] = label_encoder.fit_transform(dados['Q013'])
dados['Q013_F'].value_counts()

# Atribuicao do rotulo a variavel Q014
dados['Q014_F'] = label_encoder.fit_transform(dados['Q014'])
dados['Q014_F'].value_counts()

# Atribuicao do rotulo a variavel Q015
dados['Q015_F'] = label_encoder.fit_transform(dados['Q015'])
dados['Q015_F'].value_counts()

# Atribuicao do rotulo a variavel Q016
dados['Q016_F'] = label_encoder.fit_transform(dados['Q016'])
dados['Q016_F'].value_counts()

# Atribuicao do rotulo a variavel Q017
dados['Q017_F'] = label_encoder.fit_transform(dados['Q017'])
dados['Q017_F'].value_counts()

# Atribuicao do rotulo a variavel Q018
dados['Q018_F'] = label_encoder.fit_transform(dados['Q018'])
dados['Q018_F'].value_counts()

# Atribuicao do rotulo a variavel Q019
dados['Q019_F'] = label_encoder.fit_transform(dados['Q019'])
dados['Q019_F'].value_counts()

# Atribuicao do rotulo a variavel Q020
dados['Q020_F'] = label_encoder.fit_transform(dados['Q020'])
dados['Q020_F'].value_counts()

# Atribuicao do rotulo a variavel Q021
dados['Q021_F'] = label_encoder.fit_transform(dados['Q021'])
dados['Q021_F'].value_counts()

# Atribuicao do rotulo a variavel Q022
```

```
dados['Q022_F'] = label_encoder.fit_transform(dados['Q022'])
dados['Q022_F'].value_counts()

# Atribuicao do rotulo a variavel Q023
dados['Q023_F'] = label_encoder.fit_transform(dados['Q023'])
dados['Q023_F'].value_counts()

# Atribuicao do rotulo a variavel Q024
dados['Q024_F'] = label_encoder.fit_transform(dados['Q024'])
dados['Q024_F'].value_counts()

# Atribuicao do rotulo a variavel Q025
dados['Q025_F'] = label_encoder.fit_transform(dados['Q025'])
dados['Q025_F'].value_counts()
dados.head()

# Verificando o tipo de dado de cada coluna do arquivo
dados.dtypes

# Excluindo algumas colunas da base de dados
df = dados.drop(['NU_INSCRICAO', 'NO_MUNICIPIO_RESIDENCIA',
'SG_UF_RESIDENCIA', 'TP_SEXO', 'Q001', 'Q002', 'Q003', 'Q004',
'Q006', 'Q007', 'Q008', 'Q009', 'Q010', 'Q011', 'Q012', 'Q013',
'Q014', 'Q015', 'Q016', 'Q017', 'Q018', 'Q019', 'Q020', 'Q021',
'Q022', 'Q023', 'Q024', 'Q025'], axis=1)

dados.head()

df.head()

# Verificando o tipo de dado de cada coluna do arquivo
df.dtypes

# Para o algoritmo Random Forest nao e necessario normalizar
# os atributos numericos NU_IDADE, TP_ANO_CONCLUIU, NU_NOTA_CN,
# NU_NOTA_CH, NU_NOTA_LC, NU_NOTA_MT, NU_NOTA_REDACAO, Q005,
# NU_NOTA_MEDIA, MENOR_NOTA_CORTE_MEDICINA

# Verificando se existe alguma variavel com missing
```

```
df.isnull().any().any()

# Verificando se a variavel resposta esta balanceada
df['IN_APROVACAO_MEDICINA'].value_counts()
df['IN_APROVACAO_MEDICINA'].value_counts(normalize=True)

# 0.85% (26,169 out of 3,089,609) dos participantes do Enem
# seriam aprovados no curso de Medicina.

%matplotlib inline
plt.figure(figsize=(45,40))
sns.heatmap(dados.corr(), annot=True, linewidths=0.5, linecolor='black',
cmap='Blues')
plt.xticks(rotation=90)
plt.show()

features = ['NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_MT',
'NU_NOTA_REDACAO', 'NU_NOTA_MEDIA', 'IN_APROVACAO_MEDICINA']

%matplotlib inline
plt.figure(figsize=(5,5))
sns.heatmap(dados[features].corr(), annot=True, linewidths=0.5,
linecolor='black', cmap='Blues')
plt.xticks(rotation=90)
plt.show()

features1 = ['NU_NOTA_CN', 'NU_NOTA_CH', 'NU_NOTA_LC', 'NU_NOTA_MT',
'NU_NOTA_REDACAO', 'IN_APROVACAO_MEDICINA']

%matplotlib inline
plt.figure(figsize=(5,5))
sns.heatmap(dados[features1].corr(), annot=True, linewidths=0.5,
linecolor='black', cmap='Blues')
plt.xticks(rotation=90)
plt.show()

# Atributos
X = df.drop(labels='IN_APROVACAO_MEDICINA', axis=1)
```

```
# Variavel Resposta
y = df.loc[:, 'IN_APROVACAO_MEDICINA']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1, stratify=y)

# Quantidade de dados no conjunto de treino
X_train.shape

# Quantidade de dados no conjunto de teste
X_test.shape

X_train.is_copy = False
X_test.is_copy = False

# Modelagem Random Forest

# Ensemble methods

# Ensemble methods permitem diminuir a variancia de modelos preditivos.
# Os metodos mais populares sao bagging, boosting e random forest.
# Nessa analise sera utilizado o metodo Random Forest.

# O metodo florestas aleatorias considera a amostragem de observacoes
# e atributos.

# Como as variaveis 'NU_NOTA_MEDIA' e 'MENOR_NOTA_CORTE_MEDICINA' foram
# utilizadas na construcao da variavel resposta (aprovacao ou nao no
# curso de Medicina), elas serao retiradas do conjunto de dados no
# ajuste do modelo Random Forest.

X2_train = X_train.drop(['NU_NOTA_MEDIA', 'MENOR_NOTA_CORTE_MEDICINA'],
axis=1)
X2_test = X_test.drop(['NU_NOTA_MEDIA', 'MENOR_NOTA_CORTE_MEDICINA'],
axis=1)

# Quantidade de dados no conjunto de treino
X2_train.shape
```

```
# Quantidade de dados no conjunto de teste
X2_test.shape

X2_train.is_copy = False
X2_test.is_copy = False

vscore = []
vn = []
for n in range(1,500,50):
    model_gini_3 = RandomForestClassifier(n_estimators=n,
    criterion='gini')
    model_gini_3.fit(X2_train,y_train)
    y_pred_gini_3 = model_gini_3.predict(X2_test)
    score_gini_3 = accuracy_score(y_pred_gini_3, y_test)
    print('Number of Estimators:', n, 'Accuracy:', score_gini_3)
    vscore.append(score_gini_3)
    vn.append(n)
best_n = vn[np.argmax(vscore)]
print('Melhor n:', best_n, ' com acuracia:', vscore[np.argmax(vscore)])
plt.figure(figsize=(10,5))
plt.plot(vn, vscore, '-bo')
plt.xlabel('Number of Estimators', fontsize = 15)
plt.ylabel('Accuracy', fontsize = 15)
plt.show()

from sklearn.metrics import precision_score, recall_score,
classification_report, accuracy_score, f1_score, balanced_accuracy_score

model_gini_3 = RandomForestClassifier(n_estimators=best_n)
model_gini_3.fit(X2_train,y_train)
y_pred_gini_3 = model_gini_3.predict(X2_test)
score_gini_3 = accuracy_score(y_pred_gini_3, y_test)

print('Accuracy:', accuracy_score(y_test, y_pred_gini_3))
print('F1 score:', f1_score(y_test, y_pred_gini_3, average="macro"))
print('Precision:', precision_score(y_test, y_pred_gini_3,
average="macro"))
print('Recall:', recall_score(y_test, y_pred_gini_3, average="macro"))
print('Balanced accuracy:', balanced_accuracy_score(y_test,
```

```
y_pred_gini_3))

print('Accuracy:', accuracy_score(y_test, y_pred_gini_3))
print('F1 score:', f1_score(y_test, y_pred_gini_3, average="micro"))
print('Precision:', precision_score(y_test, y_pred_gini_3, average="micro"))
print('Recall:', recall_score(y_test, y_pred_gini_3, average="micro"))

from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_test, y_pred_gini_3)

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

print('AUPRC', average_precision_score(y_test, y_pred_gini_3))
print('AUROC', roc_auc_score(y_test, y_pred_gini_3))
print("Cohen's kappa", cohen_kappa_score(y_test, y_pred_gini_3))

print('CONFUSION MATRIX')
print(confusion_matrix(y_test, y_pred_gini_3))

tn, fp, fn, tp = confusion_matrix(y_test, y_pred_gini_3).ravel()
tn, fp, fn, tp

features_names2 = X2_train.columns
importances2 = model_gini_3.feature_importances_
indices2 = np.argsort(importances2)
lmeas_order = []
for i in indices2:
    lmeas_order.append(features_names2[i])
plt.figure(figsize=(24,16))
plt.barh(range(len(indices2)), importances2[indices2], color='b',
align='center')
plt.yticks(range(len(indices2)), lmeas_order, fontsize=15)
plt.xlabel('Relative Importance',fontsize=15)
plt.xticks(color='k', size=20)
plt.yticks(color='k', size=20)
plt.show()
```

```
# Sera ajustado o modelo de Arvore de Decisao tambem sem as variaveis
# 'NU_NOTA_MEDIA' e 'MENOR_NOTA_CORTE_MEDICINA'.

X3_train = X_train.drop(['NU_NOTA_MEDIA', 'MENOR_NOTA_CORTE_MEDICINA'],
axis=1)
X3_test = X_test.drop(['NU_NOTA_MEDIA', 'MENOR_NOTA_CORTE_MEDICINA'],
axis=1)

# Quantidade de dados no conjunto de treino
X3_train.shape

# Quantidade de dados no conjunto de teste
X3_test.shape

X3_train.is_copy = False
X3_test.is_copy = False

# Arvore de Decisao

# Ajustando uma arvore de decisao pelo criterio Gini.

from sklearn import tree
# Cria o modelo usando o criterio Gini
model_gini_4 = tree.DecisionTreeClassifier(criterion = 'gini',
random_state = 101)
# Ajusta o modelo usando os dados de treinamento
model_gini_4.fit(X3_train,y_train)
# Realiza a predicao
y_pred_gini_4 = model_gini_4.predict(X3_test)

from sklearn.metrics import accuracy_score
score_gini_4 = accuracy_score(y_pred_gini_4, y_test)
print('Accuracy:', score_gini_4)

print('Accuracy:', accuracy_score(y_test, y_pred_gini_4))
print('F1 score:', f1_score(y_test, y_pred_gini_4, average="micro"))
print('Precision:', precision_score(y_test, y_pred_gini_4,
average="micro"))
print('Recall:', recall_score(y_test, y_pred_gini_4, average="micro"))
```

```
print('F1 score:', f1_score(y_test, y_pred_gini_4, average="macro"))
print('Precision:', precision_score(y_test, y_pred_gini_4, average="macro"))
print('Recall:', recall_score(y_test, y_pred_gini_4, average="macro"))
print('Balanced accuracy:', balanced_accuracy_score(y_test, y_pred_gini_4))

from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_test, y_pred_gini_4)

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

print('AUPRC', average_precision_score(y_test, y_pred_gini_4))
print('AUROC', roc_auc_score(y_test, y_pred_gini_4))
print("Cohen's kappa", cohen_kappa_score(y_test, y_pred_gini_4))

print('CONFUSION MATRIX')
print(confusion_matrix(y_test, y_pred_gini_4))

tn, fp, fn, tp = confusion_matrix(y_test, y_pred_gini_4).ravel()
tn, fp, fn, tp

plt.figure(figsize=(45,40))
tree.plot_tree(model_gini_4.fit(X3_train,y_train),filled=True)
plt.show(True)

# Ensemble methods

# Ensemble methods permitem diminuir a variancia de modelos preditivos.
# Os metodos mais populares sao bagging, boosting e random forest.
# Nessa analise sera utilizado o metodo Boosting.

# Classificador AdaBoost

# Adaboost com Estimador Base Random Forest

# Imports
from sklearn.ensemble import AdaBoostClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

# Cria o classificador
est = AdaBoostClassifier(base_estimator=RandomForestClassifier
(n_estimators = 10), n_estimators = 10, learning_rate=0.01)

# Cria o modelo
est.fit(X2_train, y_train)

# Previsoes das classes (labels)
y_pred_est = est.predict(X2_test)

# Score nos dados de teste (Acuracia)
acc = est.score(X2_test, y_test)
print('Acuracia: %.4f' % acc)

from sklearn.metrics import accuracy_score

# Predict the response for test dataset
y_pred_est = est.predict(X2_test)
score_est = accuracy_score(y_pred_est, y_test)
print('Accuracy:', score_est)

from sklearn.metrics import precision_score, recall_score,
classification_report, accuracy_score, f1_score

print('F1 score:', f1_score(y_test, y_pred_est, average="macro"))
print('Precision:', precision_score(y_test, y_pred_est,
average="macro"))
print('Recall:', recall_score(y_test, y_pred_est, average="macro"))

print('F1 score:', f1_score(y_test, y_pred_est, average="micro"))
print('Precision:', precision_score(y_test, y_pred_est,
average="micro"))
print('Recall:', recall_score(y_test, y_pred_est, average="micro"))

from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_test, y_pred_est)
```

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

print('AUPRC', average_precision_score(y_test, y_pred_est))
print('AUROC', roc_auc_score(y_test, y_pred_est))
print("Cohen's kappa", cohen_kappa_score(y_test, y_pred_est))

print('CONFUSION MATRIX')
print(confusion\_matrix(y_test, y_pred_est))

# Adaboost com Estimador Base Arvore de Decisao

# Cria o classificador
est11 = AdaBoostClassifier(n_estimators = 1000, learning_rate=0.01)

# Cria o modelo
est11.fit(X2_train, y_train)

# Previsoes das classes (labels)
y_pred_est11 = est11.predict(X2_test)

# Score nos dados de teste (Acuracia)
acc11 = est11.score(X2_test, y_test)
print('Acuracia: %.4f' % acc11)

from sklearn.metrics import accuracy_score

# Predict the response for test dataset
y_pred_est11 = est.predict(X2_test)
score_est = accuracy_score(y_pred_est11, y_test)
print('Accuracy:', score_est)

from sklearn.metrics import matthews_corrcoef
matthews_corrcoef(y_test, y_pred_est11)

from sklearn.metrics import precision_score, recall_score,
classification_report, accuracy_score, f1_score
```

```
print('F1 score:', f1_score(y_test, y_pred_est11, average="macro"))
print('Precision:', precision_score(y_test, y_pred_est11,
average="macro"))
print('Recall:', recall_score(y_test, y_pred_est11, average="macro"))

print('F1 score:', f1_score(y_test, y_pred_est11, average="micro"))
print('Precision:', precision_score(y_test, y_pred_est11,
average="micro"))
print('Recall:', recall_score(y_test, y_pred_est11, average="micro"))

print('AUPRC', average_precision_score(y_test, y_pred_est11))
print('AUROC', roc_auc_score(y_test, y_pred_est11))
print("Cohen's kappa", cohen_kappa_score(y_test, y_pred_est11))

print('CONFUSION MATRIX')
print(confusion_matrix(y_test, y_pred_est11))
```

Anexos

ANEXO A – VARIÁVEIS UTILIZADAS NOS MODELOS DE MACHINE LEARNING

- CO_MUNICIPIO_RESIDENCIA: Código do Município de residência;
- CO_UF_RESIDENCIA: Código da Unidade da Federação de residência;
- NU_IDADE: Idade;
- TP_ESTADO_CIVIL: Estado civil;
- TP_COR_RACA: Cor/raça;
- TP_NACIONALIDADE: Nacionalidade;
- TP_ST_CONCLUSAO: Situação de conclusão do Ensino Médio;
- TP_ANO_CONCLUIU: Ano de conclusão do Ensino Médio;
- TP_ESCOLA: Tipo de escola do Ensino Médio;
- IN_ALGUMA_DEFICIENCIA: Indica se o inscrito tem alguma deficiência (baixa visão, cegueira, surdez, deficiência auditiva, surdo-cegueira, deficiência física, deficiência mental, déficit de atenção, dislexia, discalculia, autismo, visão monocular, outra deficiência ou condição especial);
- NU_NOTA_CN: Nota da prova de Ciências da Natureza;
- NU_NOTA_CH: Nota da prova de Ciências Humanas;
- NU_NOTA_LC: Nota da prova de Linguagens e Códigos;
- NU_NOTA_MT: Nota da prova de Matemática;
- TP_LINGUA: Indica a Língua Estrangeira que o inscrito escolheu (Inglês ou Espanhol);
- NU_NOTA_REDACAO: Nota da prova de redação;
- Q005: Quantidade de pessoas que moram na residência do inscrito;
- IN_APROVACAO_MEDICINA: Indica se um determinado participante do Enem seria aprovado em Medicina na universidade com a menor nota de corte nacional;
- SEXO: Sexo (variável numérica);
- Q001_F: Nível de escolaridade do pai do inscrito (variável numérica);

- Q002_F: Nível de escolaridade da mãe do inscrito (variável numérica);
- Q003_F: Ocupação do pai do inscrito (variável numérica);
- Q004_F: Ocupação da mãe do inscrito (variável numérica);
- Q006_F: Renda mensal familiar do inscrito (variável numérica);
- Q007_F: Se na residência do inscrito tem empregado(a) doméstico(a) (variável numérica);
- Q008_F: Se na residência do inscrito tem banheiro (variável numérica);
- Q009_F: Quantidade de quartos para dormir que tem na residência do inscrito (variável numérica);
- Q010_F: Se na residência do inscrito tem carro (variável numérica);
- Q011_F: Se na residência do inscrito tem motocicleta (variável numérica);
- Q012_F: Se na residência do inscrito tem geladeira (variável numérica);
- Q013_F: Se na residência do inscrito tem freezer (variável numérica);
- Q014_F: Se na residência do inscrito tem máquina de lavar roupa (variável numérica);
- Q015_F: Se na residência do inscrito tem máquina de secar roupa (variável numérica);
- Q016_F: Se na residência do inscrito tem micro-ondas (variável numérica);
- Q017_F: Se na residência do inscrito tem máquina de lavar louça (variável numérica);
- Q018_F: Se na residência do inscrito tem aspirador de pó (variável numérica);
- Q019_F: Se na residência do inscrito tem televisão em cores (variável numérica);
- Q020_F: Se na residência do inscrito tem aparelho de DVD (variável numérica);
- Q021_F: Se na residência do inscrito tem TV por assinatura (variável numérica);
- Q022_F: Se na residência do inscrito tem telefone celular (variável numérica);
- Q023_F: Se na residência do inscrito tem telefone fixo (variável numérica);
- Q024_F: Se na residência do inscrito tem computador (variável numérica);
- Q025_F: Se na residência do inscrito tem acesso à Internet (variável numérica).