

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Lucas Severiano Vieira

PREDIÇÃO DE APROVAÇÃO FIES COM NOTAS DO ENEM E IDHM

Belo Horizonte
2022

Lucas Severiano Vieira

PREDIÇÃO DE APROVAÇÃO FIES COM NOTAS DO ENEM E IDHM

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte
2022

SUMÁRIO

1. Introdução	4
1.1. Contextualização	4
1.2. O problema proposto	4
1.3. Objetivos.....	5
2. Coleta de Dados.....	6
3. Processamento/Tratamento de Dados	9
4. Análise e Exploração dos Dados	12
5. Criação de Modelos de Machine Learning.....	16
6. Interpretação dos Resultados	20
7. Apresentação dos Resultados	24
8. Links	25
REFERÊNCIAS	26
APÊNDICE	27

1. Introdução

1.1. Contextualização

A cada ano se busca mais profissionais qualificados no mercado de trabalho. Com isso cresce a necessidade de uma formação superior. Porém a demanda nas universidades públicas é muito maior que a quantidade de vagas disponíveis. Pensando em tentar minimizar este problema o Governo Federal por meio do Ministério da Educação (MEC) criou o Fundo de Financiamento Estudantil (FIES), programa destinado a concessão de financiamento para os estudantes do ensino superior privado. O Governo Federal paga as mensalidades durante o curso para a faculdade e o estudante paga ao Governo Federal em vários anos, em pequenas parcelas.

Todos os cursos superiores permitem utilização do FIES, porém o Governo avalia se o curso/faculdade possui uma boa qualidade de ensino por meio de indicadores do MEC.

Para poder utilizar o financiamento, o estudante formado a partir de 2010 deve ter feito o Exame Nacional do Ensino Médio (ENEM), obtendo nota média igual ou maior que 450 pontos e nota da redação igual ou maior que 400 pontos. Além disso a renda familiar do estudante, deve ser de até 3 salários mínimos. (GOV.BR)

1.2. O problema proposto

Nesta etapa foi utilizada a técnica dos 5-Ws, visando dar uma visão correta do problema e solução.

(Why?) Por que esse problema é importante?

Devido ao baixo valor das parcelas, no decorrer dos anos muitos alunos que tiveram aprovação no FIES, deixaram de pagar o financiamento ao Governo Federal. Com isso as regras foram aumentando, junto com a taxa de juros. Outro ponto que não existia vagas limitadas ao uso do FIES, hoje elas são, com isso a necessidade de se adequar a renda

familiar bem como as notas do ENEM se tornaram muito importantes para ter ou não aprovação no financiamento.

(Who?) De quem são os dados analisados? De um governo? Um ministério ou secretaria? Dados de clientes?

Os dados do FIES são do portal de dados abertos do Ministério da Educação (MEC) e os dados do IBGE foram obtidos por meio do Kaggle.

(What?): Quais os objetivos com essa análise? O que iremos analisar?

Analisar a importância das notas do ENEM na concessão do financiamento, bem como fatores econômicos do estudante.

(Where?): Trata dos aspectos geográficos e logísticos de sua análise.

O dataset do FIES consta os dados de todos os estudantes inscritos no Brasil e o dataset do IBGE consta os dados de todos os municípios Brasileiros.

(When?): Qual o período está sendo analisado? A última semana? Os últimos 6 meses? O ano passado?

O dataset do FIES são dos estudantes que tentaram o financiamento no ano de 2021 no segundo semestre. E a dataset do IBGE traz os dados dos anos de 2018, 2014 e 2010.

1.3. Objetivos

O objetivo deste trabalho é identificar através de algoritmos de Machine Learning, fatores que possam identificar a obtenção do FIES pelo estudante, dessa forma colaborando para o entendimento dele da importância do ENEM.

Além disso esta análise poderia ser útil na avaliação por meio do MEC, de quais inscrições poderiam ou não ser aceitas, diminuindo o tempo de verificação dos requisitos.

2. Coleta de Dados

Foram utilizadas 2 fontes de dados distintas para o trabalho, com dados do FIES e dados referentes aos municípios Brasileiros.

O primeiro dataset “*relatorio_inscricao_dados_abertos_fies_22021.csv*” foi extraído do Portal de dados abertos do Ministério da Educação (MEC) no dia 05/08/2022 através do link

http://dadosabertos.mec.gov.br/images/conteudo/fies/2021/relatorio_inscricao_dados_abertos_fies_22021.csv Os dados estão na seguinte estrutura:

Nome da coluna/campo	Descrição	Tipo
Ano do processo seletivo	Ano do processo FIES	int64
Semestre do processo seletivo	Semestre do processo FIES	int64
ID do estudante	Código do estudante no FIES	int64
Sexo	Sexo	object
Data de Nascimento	Data de Nascimento do estudante	object
UF de residência	Estado do estudante	object
Município de residência	Cidade do estudante	object
Etnia/Cor	Cor do estudante	object
Pessoa com deficiência?	Sim/Não	object
Tipo de escola no ensino médio	Sim/Não	object
Ano conclusão ensino médio	Ano que concluiu o ensino médio	int64
Concluiu curso superior?	Sim/Não	object
Professor rede pública ensino?	Sim/Não	object
Nº de membros Grupo Familiar	Quantidade de pessoas na família	int64
Renda familiar mensal bruta	Renda bruta da família	object
Renda mensal bruta per capita	Renda por pessoa da família	object
Região grupo de preferência	Região do Brasil da faculdade	object
UF	Estado da faculdade	object
Cod.Microrregião	Código da Microrregião	int64

Microrregião	Descrição da Microrregião	object
Cod.Mesorregião	Código da Mesorregião	int64
Mesorregião	Descrição da Mesorregião	object
Conceito de curso do GP	Conceito do curso	object
Área do conhecimento	Área do conhecimento do curso	object
Subárea do conhecimento	Subárea do conhecimento do curso	object
Cod. do Grupo de preferência	Código do curso de preferência	int64
Nota Corte Grupo Preferência	Nota de corte do curso de preferência	object
Opções de cursos da inscrição	Quantidade de cursos inscritos	int64
Nome mantenedora	Nome da faculdade	object
Natureza Jurídica Mantenedora	Natureza Jurídica da faculdade	object
CNPJ da mantenedora	CNPJ da faculdade	int64
Código e-MEC da Mantenedora	Código e-MEC da faculdade	int64
Nome da IES	Descrição da IES	object
Código e-MEC da IES	Código da IES	int64
Organização Acadêmica da IES	Organização da IES	object
Município da IES	Cidade da IES	object
UF da IES	Estado da IES	object
Nome do Local de oferta	Nome e campus da faculdade	object
Código do Local de Oferta	Código da faculdade	int64
Município do Local de Oferta	Cidade da faculdade	object
UF do Local de Oferta	Estado da faculdade	object
Código do curso	Código do curso escolhido	int64
Nome do curso	Nome do curso escolhido	object
Turno	Turno do curso escolhido	object
Grau	Semestral ou Anual	object
Conceito	Conceito do curso IES	object
Média nota Enem	Nota média do Enem	object
Ano do Enem	Ano realizado o Enem	int64
Redação	Nota Redação do Enem	int64
Matemática e suas Tecnologias	Nota Matemática do Enem	object

Linguagens, Códigos e suas Tec	Nota Linguagens do Enem	object
Ciências Natureza e suas Tec	Ciências Natureza do Enem	object
Ciências Humanas e suas Tec	Ciências Humanas do Enem	object
Situação Inscrição Fies	Situação da inscrição	object
Percentual de financiamento	Percentual de financiamento FIES	object
Semestre do financiamento	Semestre do financiamento FIES	object
Qtde semestre financiado	Qtde semestre financiado FIES	float64

O segundo dataset “*Cities_Brazil_IBGE.xlsx*” foi extraído do Kaggle dia 06/09/2022 através do link <https://www.kaggle.com/datasets/gabrielrs3/economy-and-population-of-cities-in-brazil-ibge/download?datasetVersionNumber=1> Os dados estão na seguinte estrutura:

Nome da coluna/campo	Descrição	Tipo
IBGECODE	Código da cidade no IBGE	float64
LocalCidade	Nome da Cidade	object
LocalUF	Sigla do Estado	object
LocalEstado	Nome do Estado	object
RegiaoBrasil	Região que faz parte no país	object
Latitude	Código da Latitude geográfica	object
Longitude	Código da Longitude geográfica	object
Gentilico	Nome dado a quem nasce na cidade	object
PopEstimada_2018	População estimada em 2018	float64
PopCenso 2010	Censo demográfico em 2010	float64
IDHM	Índice de desenvolvimento humano	object
ReceitasRealizadas_2014	Receitas em 2014	float64
DespesasEmpenhadas_2014	Despesas em 2014	float64
Pib_2014	Produto interno bruto de 2014	object

O relacionamento entre os datasets foi feito entre as cidades e estados, sendo estes valores únicos no segundo dataset.

3. Processamento/Tratamento de Dados

O script foi feito em Python 3.0 no Google Colab.

Foi feita a importação das bibliotecas necessárias para o processamento, tratamento dos dados e criação dos modelos de machine learning.

```
# Import
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# ML
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import resample
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

Foi feita a importação dos 2 datasets, listando logo abaixo o total de dados obtidos de cada um.

```
# Carga dos dados - FIES
df_fies = pd.read_csv('relatorio_inscricao_dados_abertos_fies_22021.csv', encoding="ISO-8859-1", sep = ';')
df_fies.shape
(237965, 57)

# Carga dos dados - Cidades
df_cidades = pd.read_excel('Cities_Brazil_TBGE.xlsx')
df_cidades.shape
(5570, 14)
```

Após a leitura dos dados, se percebeu diferença na grafia e acentuação nos nomes de algumas cidades (campo chave do relacionamento entre os datasets), com isso foi necessário fazer um tratamento nestes valores nos dados do FIES. Foi criada uma função para ajuste da acentuação.

```
# Função para retinar acentos
def corrigir_nomes(nome):
    nome = nome.replace('ç', '').replace('ç', 'c').replace('ã', 'a').replace('õ', 'o').replace('ä', 'a').replace('é', 'e').replace('ö', 'o').replace('á', 'a').replace('ê', 'e').replace('í', 'i').replace('ô', 'o').replace('ú', 'u')
    return nome
```

E foi feito ajustes na grafia de algumas cidades.

```
[ ] # Ajuste cidades da base do FIES

df_fies["Município de residência"] = df_fies["Município de residência"].apply(corriger_nomes)

df_fies.replace(to_replace = "EMBU", value = "EMBU DAS ARTES", inplace=True)
df_fies.replace(to_replace = "MOJI MIRIM", value = "MOGI MIRIM", inplace=True)
df_fies.replace(to_replace = "JI-PARANA", value = "JI PARANA", inplace=True)
df_fies.replace(to_replace = "SANTA BARBARA D'OESTE", value = "SANTA BARBARA D OESTE", inplace=True)
df_fies.replace(to_replace = "LIVRAMENTO DE MOSSA SEHORA", value = "LIVRAMENTO DO BRUMADO", inplace=True)
df_fies.replace(to_replace = "GUAJARA-MIRIM", value = "GUAJARA MIRIM", inplace=True)
df_fies.replace(to_replace = "IGARAPÉ-HIRI", value = "IGARAPÉ-MIRIM", inplace=True)
df_fies.replace(to_replace = "NOVA BRASILÂNDIA D'OESTE", value = "NOVA BRASILÂNDIA D OESTE", inplace=True)
df_fies.replace(to_replace = "ESPIGAO D'OESTE", value = "ESPIGAO D OESTE", inplace=True)
df_fies.replace(to_replace = "PINGO-D'ÁGUA", value = "PINGO D'ÁGUA", inplace=True)
df_fies.replace(to_replace = "COLORADO DO OESTE", value = "COLORADO D OESTE", inplace=True)
df_fies.replace(to_replace = "ITABIRINHA", value = "ITABIRINHA DE MANTENA", inplace=True)
df_fies.replace(to_replace = "PEROLA D'OESTE", value = "PEROLA D OESTE", inplace=True)
df_fies.replace(to_replace = "SAO MIGUEL DO GOSTOSO", value = "SAO MIGUEL DE TOUROS", inplace=True)
df_fies.replace(to_replace = "BALNEARIO PICARRAS", value = "PICARRAS", inplace=True)
df_fies.replace(to_replace = "SAO JORGE D'OESTE", value = "SAO JORGE D OESTE", inplace=True)
df_fies.replace(to_replace = "PRESIDENTE CASTELLO BRANCO", value = "PRESIDENTE CASTELO BRANCO", inplace=True)
df_fies.replace(to_replace = "CAMPO DE SANTANA", value = "TACIMA", inplace=True)
```

Com os dados das cidades corrigidos, foi feito o merge entre os datasets.

```
[ ] # Merge dos Datasets

df_dados = pd.merge(df_fies, df_cidades, how='left', left_on=['Município de residência', 'UF de residência'], right_on = ['Localidade', 'LocalUF'])
df_dados.shape

(237965, 71)

[ ] # Leitura dos dados - Dados agrupados

df_dados.head()
```

	Ano do processo seletivo	Semestre do processo seletivo	ID do estudante	Sexo	Data de Nascimento	UF de residência	Município de residência	Etnia/Cor	Pessoa com deficiência?	Tipo de escola no ensino médio	...	RegiãoBrasil	Latitude	Longitude	Gentílico	PopEstimada_2018	PopCenso 2010	IDM	ReceitasRealizadas_2014	DespesasEmpenhadas
0	2021	2	281805657	M	15/05/00	PR	COLOMBO	PARDA	NÃO	SIM	...	SUL	-252925	-492262	colombense	240840.0	212967.0	0.733		318326.0
1	2021	2	281805657	M	15/05/00	PR	COLOMBO	PARDA	NÃO	SIM	...	SUL	-252925	-492262	colombense	240840.0	212967.0	0.733		318326.0
2	2021	2	351067941	M	10/07/94	PR	CURITIBA	BRANCA	NÃO	NÃO	...	SUL	-254195	-492646	curitibano	1917185.0	1751907.0	0.823		6962143.0
3	2021	2	205906170	M	25/09/89	PR	CURITIBA	BRANCA	NÃO	SIM	...	SUL	-254195	-492646	curitibano	1917185.0	1751907.0	0.823		6962143.0
4	2021	2	351067085	F	20/07/99	PR	PARANAGUA	BRANCA	NÃO	SIM	...	SUL	-255161	-485225	paranguara	153666.0	140469.0	0.75		355256.0

5 rows x 71 columns

Foi excluída colunas que não seriam relevantes para o trabalho.

```
# Deleção das colunas não utilizadas

df_dados = df_dados.drop(['Nome mantenedora', 'Natureza Jurídica Mantenedora', 'CNPJ da mantenedora', 'Código e-MEC da Mantenedora', 'Nome da IES', 'Código e-MEC da IES', 'Organização Acadêmica da IES'], axis=1)
df_dados = df_dados.drop(['Município da IES', 'UF da IES', 'Nome do Local de oferta', 'Código do Local de Oferta', 'Município do Local de Oferta', 'UF do Local de Oferta', 'Grau', 'Conceito'], axis=1)
df_dados = df_dados.drop(['Região grupo de preferência', 'UF', 'Cod.Microrregião', 'Microrregião', 'Cod.Mesoregião', 'Mesoregião', 'Conceito de curso do GP', 'Subárea do conhecimento', 'Área do conhecimento'], axis=1)
df_dados = df_dados.drop(['Opções de cursos da inscrição', 'Código do curso', 'Percentual de financiamento', 'Semestre do financiamento', 'Qtde semestre financiado', 'IBGECode'], axis=1)
df_dados = df_dados.drop(['RegiãoBrasil', 'Latitude', 'Longitude', 'Gentílico', 'Ano do processo seletivo', 'Semestre do processo seletivo', 'LocalUF', 'Município de residência', 'Data de Nascimento'], axis=1)
df_dados = df_dados.drop(['Professor rede pública ensino', 'Ano do Enem', 'ReceitasRealizadas_2014', 'DespesasEmpenhadas_2014', 'PopEstimada_2018', 'PopCenso 2010'], axis=1)
df_dados = df_dados.drop(['Tipo de escola no ensino médio', 'UF de residência', 'Cod. do Grupo de preferência', 'Ano conclusão ensino médio', 'ID do estudante', 'Pib_2014'], axis=1)
df_dados = df_dados.drop(['Renda familiar mensal bruta', 'Pessoa com deficiência?', 'Concluiu curso superior?', 'Nº de membros Grupo Familiar'], axis=1)
```

Observado os campos restantes e seus respectivos tipos. Em seguida renomeada algumas colunas para facilitar na sua manipulação.

```
[ ] # Informações sobre tipo dos dados
df_dados.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 237965 entries, 0 to 237964
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  -
 0   Sexo                237965 non-null object
 1   Etnia/Cor           237965 non-null object
 2   Renda mensal bruta per capita  237965 non-null object
 3   Nota Corte Grupo Preferência  237965 non-null object
 4   Nome do curso       237965 non-null object
 5   Turno               237965 non-null object
 6   Média nota Enem     237965 non-null object
 7   Redação            237965 non-null int64
 8   Matemática e suas Tecnologias  237965 non-null object
 9   Línguagens, Códigos e suas Tec  237965 non-null object
10   Ciências Natureza e suas Tec  237965 non-null object
11   Ciências Humanas e suas Tec  237965 non-null object
12   Situação Inscrição Fies  237965 non-null object
13   Localidade         237968 non-null object
14   LocalEstado        237968 non-null object
15   IDMH               237948 non-null object
dtypes: int64(1), object(15)
memory usage: 38.9+ MB

[ ] # Renomeando colunas
df_dados.rename(columns={'Etnia/Cor': 'Cor',
                        'Renda mensal bruta per capita': 'Renda_per_capita',
                        'Média nota Enem': 'Media_Enem',
                        'Matemática e suas Tecnologias': 'Matematica',
                        'Línguagens, Códigos e suas Tec': 'Linguagens',
                        'Ciências Natureza e suas Tec': 'Ciencias_natureza',
                        'Ciências Humanas e suas Tec': 'Ciencias_humanas',
                        'Nome do curso': 'Curso',
                        'Nota Corte Grupo Preferência': 'Nota_corte',
                        'Situação Inscrição Fies': 'Situacao'}, inplace = True)
```

Verificação da existência de valores nulos.

```
[15] # Análise de dados faltantes
df_dados.isnull().sum().sort_values(ascending=False)[:10]

IDMH          17
Localidade     5
LocalEstado    5
Sexo           0
Cor            0
Renda_per_capita 0
Nota_corte     0
Curso          0
Turno          0
Media_Enem     0
dtype: int64
```

Como foram identificados poucos registros nulos, foi decidido remover estes.

```
[16] # Limpeza de dados nulos
df_dados.dropna(subset=["IDMH"], inplace=True)
df_dados.dropna(subset=["LocalEstado"], inplace=True)
df_dados.dropna(subset=["Localidade"], inplace=True)
```

Em seguida foram feitas as conversões das variáveis para processamento nos modelos.

```
[18] # Conversão numéricos
df_dados['Nota_corte'] = df_dados['Nota_corte'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['Media_Enem'] = df_dados['Media_Enem'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['Matematica'] = df_dados['Matematica'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['Linguagens'] = df_dados['Linguagens'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['Ciencias_natureza'] = df_dados['Ciencias_natureza'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['Ciencias_humanas'] = df_dados['Ciencias_humanas'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['IDMH'] = df_dados['IDMH'].apply(lambda x: x.replace(",",".").astype(float))
df_dados['Renda_per_capita'] = df_dados['Renda_per_capita'].apply(lambda x: x.replace(",",".").astype(float))

[19] # Conversão categóricos
label_encoder = LabelEncoder()
df_dados['Curso'] = label_encoder.fit_transform(df_dados['Curso'])
df_dados['LocalEstado'] = label_encoder.fit_transform(df_dados['LocalEstado'])
df_dados['Localidade'] = label_encoder.fit_transform(df_dados['Localidade'])
```

Foi feita a limpeza da coluna alvo, pois existem algumas situações que não são relevantes, sobrando apenas aqueles que contratam ou não conseguiram acesso ao FIES.

```
[19] # Limpeza da coluna alvo, apenas vai ser verificado dados contratados ou não
df_dados.drop(df_dados.loc[df_dados['Situacao']=='INSCRIÇÃO POSTERGADA'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='LISTA DE ESPERA'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='OPÇÃO NÃO CONTRATADA'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='PARTICIPACAO CANCELADA PELO CANDIDATO'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='PRÉ-SELECIONADO'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='REJEITADA PELA CPSA'].index, inplace=True)

# Conversão para numérico
df_dados['Situacao'].replace(['NÃO CONTRATADO', 'CONTRATADA'], [0, 1], inplace=True)

print(df_dados.groupby('Situacao').size())
```

```
0    Situacao
0    115773
1     22442
dtype: int64
```

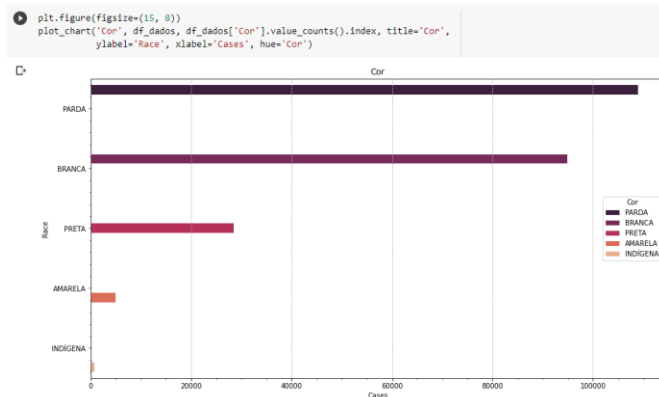
Após o tratamento o dataset ficou com a seguinte estrutura.

```
[21] # Estrutura dos dados
df_dados.info()

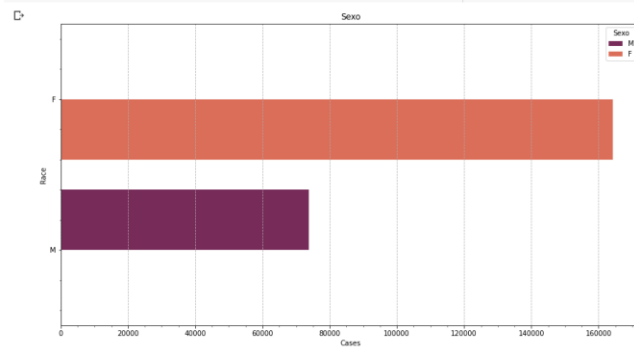
<class 'pandas.core.frame.DataFrame'>
Int64Index: 138215 entries, 0 to 237961
Data columns (total 13 columns):
 # Column          Non-Null Count  Dtype
---  --
0  Renda_per_capita  138215 non-null  Float64
1  Nota_corte        138215 non-null  Float64
2  Curso             138215 non-null  int64
3  Media_enem        138215 non-null  Float64
4  Redação           138215 non-null  int64
5  Matematica        138215 non-null  Float64
6  Linguagens        138215 non-null  Float64
7  Ciencias_natureza  138215 non-null  Float64
8  Ciencias_humanas  138215 non-null  Float64
9  Situacao          138215 non-null  int64
10 LocalCidade      138215 non-null  int64
11 LocalEstado      138215 non-null  int64
12 IDMEI            138215 non-null  Float64
dtypes: float64(8), int64(5)
memory usage: 14.8 MB
```

4. Análise e Exploração dos Dados

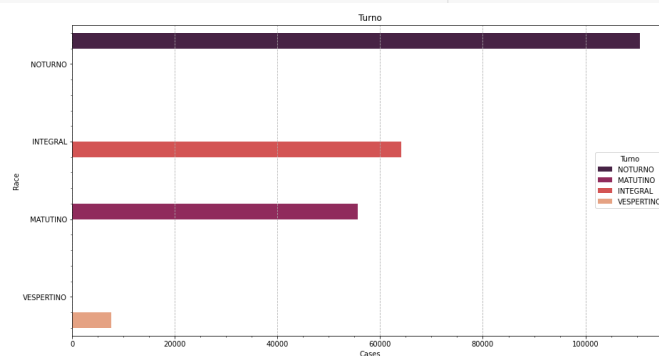
Foram plotados gráficos de como estava a distribuição dos dados por Cor, Sexo e Turno dos cursos.



```
plt.figure(figsize=(15, 8))
plot_chart('Sexo', df_dados, df_dados['Sexo'].value_counts().index, title='Sexo',
           ylabel='Race', xlabel='Cases', hue='Sexo')
```



```
[ ] plt.figure(figsize=(15, 8))
plot_chart('Turno', df_dados, df_dados['Turno'].value_counts().index, title='Turno',
           ylabel='Race', xlabel='Cases', hue='Turno')
```



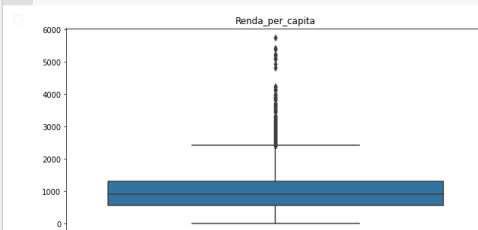
As 3 variáveis (Cor, Sexo, Turno) foram removidas, pois não foram consideradas relevantes para os modelos.

```
[17] #Exclusão de colunas não utilizadas no modelo
df_dados = df_dados.drop(['Cor', 'Sexo', 'Turno'], axis=1)
```

Feita análise de outliers de todas as variáveis. Observado desvio na variável Renda_per_capita de alguns registros.

```
# Análise de Outliers
numeric_columns = df_dados.select_dtypes(np.number).columns

for col in numeric_columns:
    plt.figure(figsize=(10,5))
    sns.boxplot(y=col, data=df_dados)
    plt.title(col)
    plt.ylabel(None)
    plt.show()
```

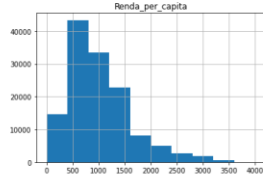


Estes registros da Renda_per_capita foram removidos, bem como foram eliminados os registros que não se adequam as regras do FIES, com relação a Renda, Nota do Enem e Nota da Redação.

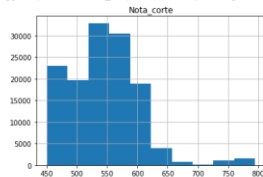
```
[27] # Remoção de outliers e cortes da regra do FIES
df_remove = df_dados.loc[(df_dados['Renda_per_capita'] > 4000) | (df_dados['Media_Enem'] < 450) | (df_dados['Redação'] < 400)]
df_dados = df_dados.drop(df_remove.index)
```

Plotado histograma das variáveis para entendimento das distribuições.

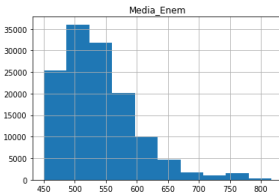
```
[28] print(df_dados.hist(column = ['Renda_per_capita']))
[[<matplotlib.axes._subplots.AxesSubplot object at 0x7ffb4be7a590>]]
```



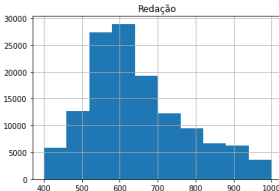
```
[29] print(df_dados.hist(column = ['Nota_corte']))
[[<matplotlib.axes._subplots.AxesSubplot object at 0x7ffb4bef5e90>]]
```



```
[ ] print(df_dados.hist(column = ['Media_Enem']))
[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f049504ec50>]]
```

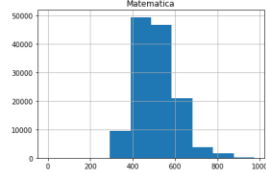


```
[ ] print(df_dados.hist(column = ['Redação']))
[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f0494fcc310>]]
```



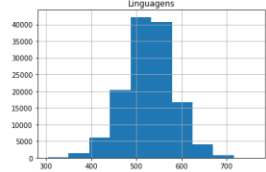
```
[ ] print(df_dados.hist(column = ['Matematica']))
```

[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f0494f48d0>]]



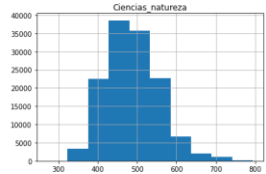
```
[ ] print(df_dados.hist(column = ['Linguagens']))
```

[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f0494f62e90>]]



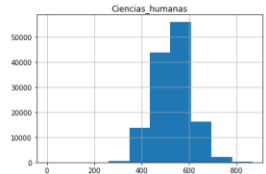
```
[ ] print(df_dados.hist(column = ['Ciencias_natureza']))
```

[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f0494eeaf90>]]



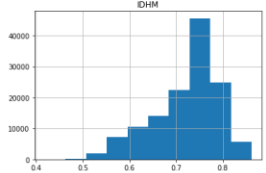
```
[ ] print(df_dados.hist(column = ['Ciencias_humanas']))
```

[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f0494e05810>]]



```
[ ] print(df_dados.hist(column = ['IDHM']))
```

[[<matplotlib.axes._subplots.AxesSubplot object at 0x7f0494de9ed0>]]



Após foi feita a normalização dos dados.

```
[ ] # Normalização dos dados
df_max_scaled = df_dados.copy()
for column in df_max_scaled.columns:
    df_max_scaled[column] = df_max_scaled[column].abs().max()

display(df_max_scaled)
```

	Renda_per_capita	Nota_corte	Curso	Media_Enem	Redação	Matematica	Linguagens	Ciencias_natureza	Ciencias_humanas	Situacao	LocalCidade	LocalEstado	IDHM
0	0.354420	0.780957	0.277344	0.763668	0.64	0.692718	0.844094	0.679809	0.716325	0.0	0.236602	0.500000	0.850348
1	0.354420	0.780957	0.277344	0.763668	0.64	0.692718	0.844094	0.679809	0.716325	0.0	0.236602	0.500000	0.850348
4	0.208333	0.780957	0.277344	0.626695	0.68	0.419077	0.655906	0.610278	0.561708	0.0	0.649989	0.500000	0.870070
5	0.208333	0.780957	0.277344	0.626695	0.68	0.419077	0.655906	0.610278	0.561708	0.0	0.649989	0.500000	0.870070
7	0.091667	0.780957	0.277344	0.589056	0.60	0.370667	0.729265	0.620733	0.457172	0.0	0.649989	0.500000	0.870070
...
237954	0.381250	0.566720	0.093750	0.771695	0.76	0.677231	0.775984	0.682076	0.690997	0.0	0.827218	0.961538	0.933875
237955	0.300000	0.577842	0.347656	0.761906	0.56	0.691795	0.839633	0.732586	0.756850	0.0	0.195686	0.961538	0.868910
237959	0.095833	0.577842	0.347656	0.565856	0.50	0.497333	0.644619	0.494521	0.510592	0.0	0.873916	0.961538	0.933875
237960	0.108332	0.577842	0.347656	0.698204	0.60	0.660000	0.783202	0.630054	0.590030	0.0	0.815877	0.961538	0.945476
237961	0.275000	0.585189	0.398438	0.612917	0.56	0.440821	0.707743	0.645799	0.532696	0.0	0.873916	0.961538	0.933875

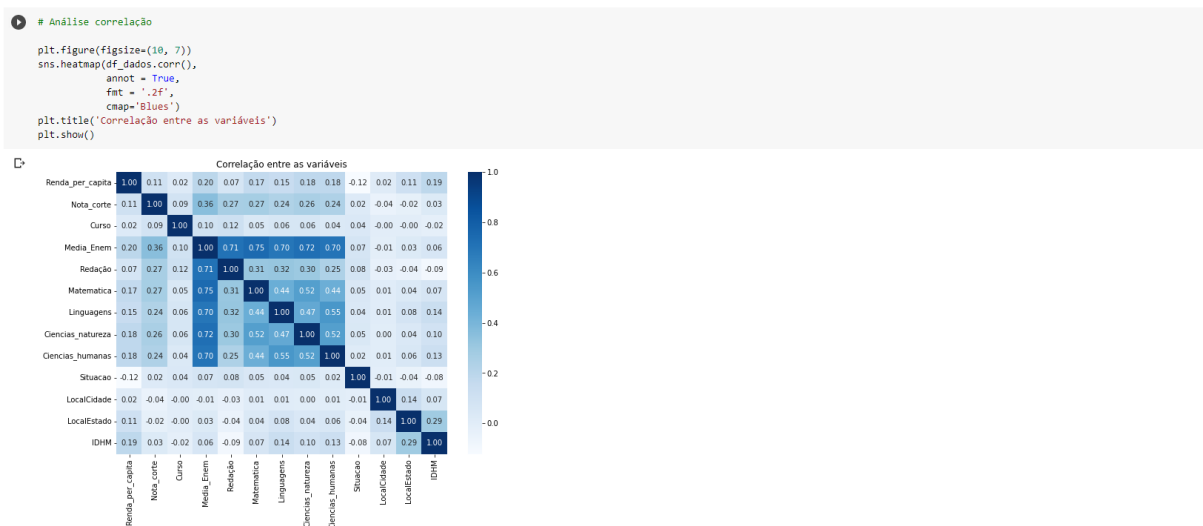
132529 rows x 13 columns

Verificado dados estatísticos do dataset.

```
#Estatística dos dados
df_dados.describe()
```

	Renda_per_capita	Nota_corte	Curso	Media_Enem	Redação	Matematica	Linguagens	Ciencias_natureza	Ciencias_humanas	Situacao	LocalCidade	LocalEstado	IDHM
count	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000	132529.000000
mean	1014.299655	544.871342	116.661644	538.934670	636.863720	511.273027	528.092357	487.908448	530.535797	0.163013	2281.116337	13.384920	0.723308
std	590.784878	58.829836	65.521012	60.356753	130.014709	94.252337	54.570407	66.770418	74.086035	0.369379	1335.735446	7.323398	0.070527
min	0.000000	450.020000	0.000000	450.000000	400.000000	0.000000	304.400000	271.000000	0.000000	0.000000	0.000000	0.000000	0.418000
25%	550.000000	503.180000	71.000000	495.340000	540.000000	441.800000	493.900000	438.700000	480.600000	0.000000	1060.000000	6.000000	0.583000
50%	900.000000	545.460000	113.000000	528.660000	600.000000	500.300000	529.300000	482.800000	532.700000	0.000000	2342.000000	13.000000	0.737000
75%	1300.000000	579.980000	192.000000	569.680000	720.000000	568.900000	563.200000	531.200000	579.900000	0.000000	3505.000000	19.000000	0.770000
max	4000.000000	794.820000	256.000000	817.240000	1000.000000	975.000000	762.000000	793.900000	868.600000	1.000000	4497.000000	26.000000	0.962000

Mapa de correlação.



5. Criação de Modelos de Machine Learning

Visando identificar a contratação ou não para o FIES, foram aplicados 3 modelos de Machine Learning. Porém a variável alvo (SITUACAO) se percebeu que estava desbalanceada, assim foi feito o devido tratamento. Foi utilizado o resample para equalizar os valores, assim os algoritmos não serão enviesados devido diferença no tamanho das amostras.


```

# Verificando balanceamento da coluna alvo
balance = df_dados['Situacao'].value_counts(normalize=True).round(2)
balance.index = balance.index.map({0: 'NÃO CONTRATADO', 1: 'CONTRATADA'})
balance * 100

NÃO CONTRATADO    84.0
CONTRATADA        16.0
Name: Situacao, dtype: float64

[71] # Total não efetivados
df_maior = df_dados[df_dados['Situacao'] == 0]
print('linhas:', df_maior.shape[0])

linhas: 110925

[72] # Total efetivados
df_menor = df_dados[df_dados['Situacao'] == 1]
print('linhas:', df_menor.shape[0])

linhas: 21604

[73] # Igualando quantidade de registros SIM/NÃO
df_unsampled = resample(df_maior, replace = True, n_samples=21603, random_state=123)

```

Após o tratamento, foi avaliado o resultado.

```

# Concatenando registros SIM/NÃO
df_dados = pd.concat([df_unsampled, df_menor])
df_dados.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 43207 entries, 41337 to 237924
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ---
 0   Renda_per_capita    43207 non-null  float64
 1   Nota_corte          43207 non-null  float64
 2   Curso               43207 non-null  int64
 3   Media_Enem          43207 non-null  float64
 4   Redação             43207 non-null  int64
 5   Matematica          43207 non-null  float64
 6   Linguagens          43207 non-null  float64
 7   Ciencias_natureza   43207 non-null  float64
 8   Ciencias_humanas    43207 non-null  float64
 9   Situacao            43207 non-null  int64
10  LocalCidade         43207 non-null  int64
11  LocalEstado         43207 non-null  int64
12  IDHM                43207 non-null  float64
dtypes: float64(8), int64(5)
memory usage: 4.6 MB

```

```

# Análise balanceamento
import plotly.express as px
plot_ = df_dados['Situacao'].map({1: 'CONTRATADA', 0: 'NÃO CONTRATADO'}).value_counts().reset_index()
fig = px.bar(plot_,
             x='index',
             y='Situacao',
             labels={'index': 'Alvo', 'Situacao': ''},
             title='Balanceamento do Alvo',
             width=600)
fig.update_xaxes(type='category')
fig.show()

```



A variável alvo (SITUACAO) foi separada das demais que vão ser utilizadas para prever o alvo. Em seguida foram separados os dados de treino e teste, sendo 20% teste e 80% treino do algoritmo.

```
# Separação dos Dados - Dataframe
Var_Caracteristicas = df_dados.drop(columns='Situacao')
Var_Previsao = df_dados['Situacao']

# Separação dos Dados - Treino/Teste
x_treino, x_teste, y_treino, y_teste = train_test_split(Var_Caracteristicas, Var_Previsao, test_size=0.20, random_state=10)

print(f'Dados de treino: {len(x_treino)}')
print(f'Dados de teste: {len(x_teste)}')
```

Dados de treino: 34565
Dados de teste: 8642

O primeiro modelo escolhido foi a Regressão Logística que é algoritmo que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável. Para sua utilização foi importada a classe *LogisticRegression* do pacote *sklearn.linear_model*.

```
# Criação do modelo

logistic_regression = LogisticRegression(random_state= 42, max_iter=10000)
logistic_regression.fit(x_treino, y_treino)

# Classificação
Train_predict = logistic_regression.predict(x_teste)
print(classification_report(y_teste, Train_predict))

# Acurácia
resultado = logistic_regression.score(x_teste, y_teste)
print('Acurácia:', resultado)
```

	precision	recall	f1-score	support
0	0.60	0.54	0.57	4294
1	0.59	0.64	0.61	4348
accuracy			0.59	8642
macro avg	0.59	0.59	0.59	8642
weighted avg	0.59	0.59	0.59	8642

Acurácia: 0.5930340199028002

O segundo modelo escolhido foi a Árvore de Decisão que é um algoritmo utilizado para classificação e para regressão. Assim como um fluxograma, a árvore de decisão estabelece nós que se relacionam entre si por uma hierarquia. Existe o nó-raiz, que é o mais importante, e os nós-folha, que são os resultados. Para sua utilização foi importada a classe *DecisionTreeClassifier* do pacote *sklearn.tree*.

```
[▶] # Criação do modelo

tree = DecisionTreeClassifier()
tree.fit(x_treino, y_treino)

# Classificação
Train_predict = tree.predict(x_teste)
print(classification_report(y_teste, Train_predict ))

# Acurácia
resultado = tree.score(x_teste, y_teste)
print('Acurácia:', resultado)
```

```
↳
```

	precision	recall	f1-score	support
0	0.62	0.67	0.64	4294
1	0.65	0.59	0.62	4348
accuracy			0.63	8642
macro avg	0.63	0.63	0.63	8642
weighted avg	0.63	0.63	0.63	8642

Acurácia: 0.6308724832214765

O terceiro modelo escolhido foi a Random Forest que é um algoritmo utilizado para problemas que envolvam classificação ou regressão. Ele se baseia em uma coleção de árvores de decisão, combinando os resultados de todos eles para se obter um resultado único e mais eficiente. Para sua utilização foi importada a classe *RandomForestClassifier* do pacote *sklearn.ensemble*.

```
[379] # Criação do modelo

random_forest = RandomForestClassifier(random_state= 42)
random_forest.fit(x_treino, y_treino)

# Classificação
Train_predict = random_forest.predict(x_teste)
print(classification_report(y_teste, Train_predict))

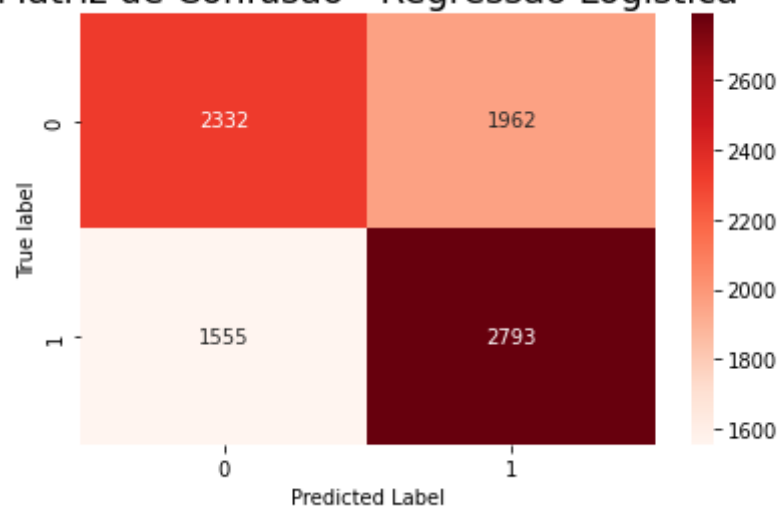
# Acurácia
resultado = random_forest.score(x_teste, y_teste)
print('Acurácia:', resultado)
```

	precision	recall	f1-score	support
0	0.68	0.71	0.69	4294
1	0.70	0.68	0.69	4348
accuracy			0.69	8642
macro avg	0.69	0.69	0.69	8642
weighted avg	0.69	0.69	0.69	8642

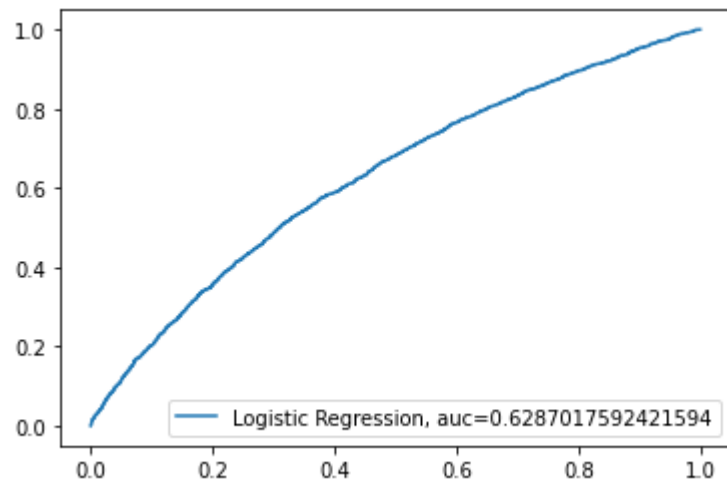
Acurácia: 0.6912751677852349

6. Interpretação dos Resultados

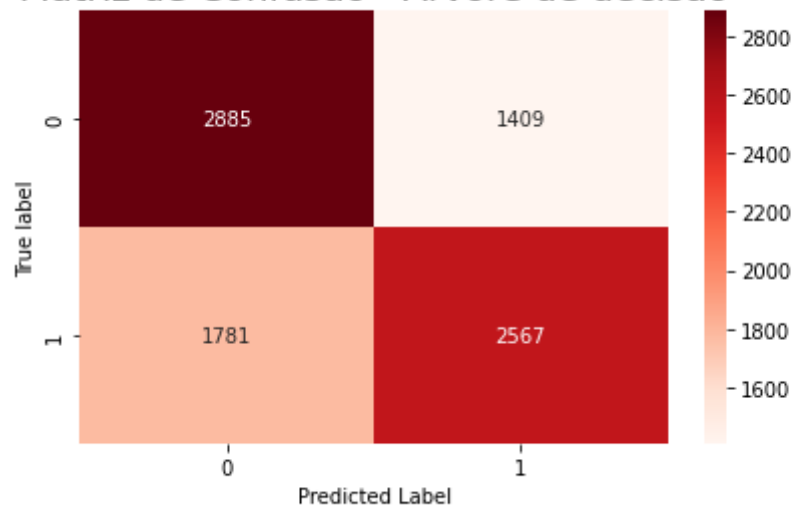
Matriz de Confusão - Regressão Logística



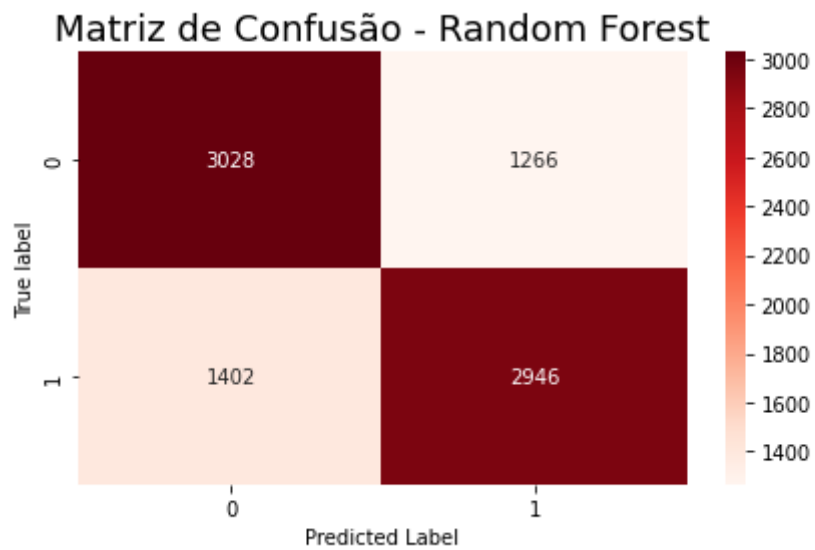
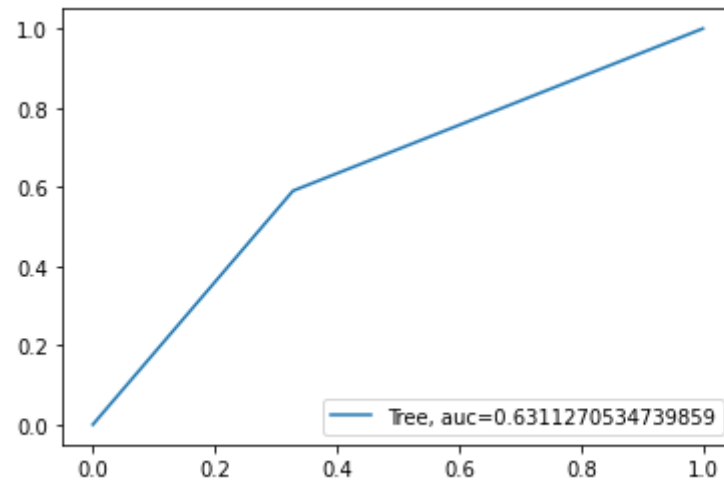
Curva ROC da Regressão Logística.



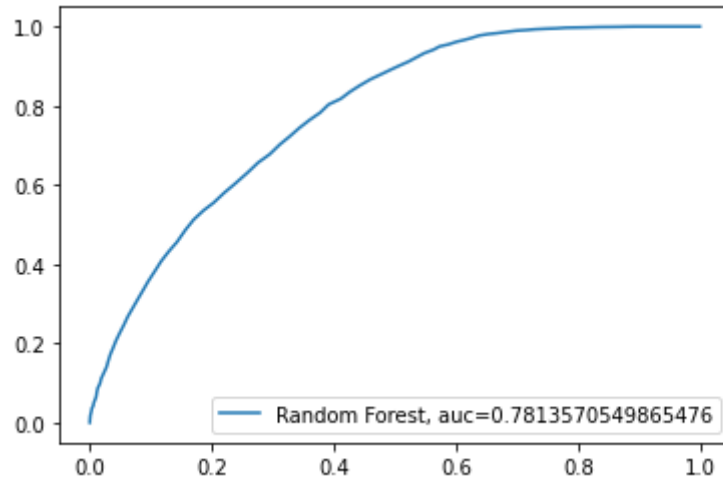
Matriz de Confusão - Árvore de decisão



Curva ROC da Árvore de Decisão.



Curva ROC da Random Forest.



Modelo	Accuracy	Precision	Recall	F1-Score
Regressão logística	59%	59%	54%	57%
Árvore de decisão	63%	65%	67%	64%
Random Forest	69%	70%	71%	69%

O balanceamento da coluna alvo foi importante para melhorar acurácia dos “Não contratados”, porém ambos os modelos não tiveram uma assertividade tão grande sendo o Random Forest o melhor com 69%. Assim cabe testar os modelos com outros conjuntos de dados buscando uma acurácia melhor para poder ser utilizado de forma efetiva.

7. Apresentação dos Resultados

Data Science Workflow Canvas*

Start here. The sections below are ordered intentionally to make you state your goals first, followed by steps to achieve those goals. You're allowed to switch orders of these steps!

Title: PREDIÇÃO DE APROVAÇÃO FIES COM NOTAS DO ENEM E IDHM		
1 Problem Statement What problem are you trying to solve? What larger issues do the problem address? Quantidade limitada de vagas no FIES x alto número de estudantes buscando uma vaga no ensino superior.	2 Outcomes/Predictions What prediction(s) are you trying to make? Identify applicable predictor (X) and/or target (y) variables. Determinar, a partir de notas do ENEM, de características econômicas do estudante e do local onde mora, a possibilidade de aprovação no FIES. Variável alvo "Situacao" da inscrição e as demais variáveis preditoras.	3 Data Acquisition Where are you sourcing your data from? Is there enough data? Can you work with it? Dados do portal aberto do MEC dos inscritos no FIES no 2 semestre de 2021 e dados do IBGE.
4 Modeling What models are appropriate to use given your outcomes? Foram feitas classificações com Regressão Logística, Árvore de decisão e Random Forest.	5 Model Evaluation How can you evaluate your model's performance? Índice de acuracidade Classification_Report Matriz de confusão Curva ROC	6 Data Preparation What do you need to do to your data in order to run your model and achieve your outcomes? Feita coleta dos dados, exclusão de colunas, renomeadas algumas colunas, tratamento de nulos, transformação de tipo de dados, correção de grafia e união dos datasets. Para aplicação dos modelos foi feito balanceamento da coluna alvo, visto quantidade de amostras "não" ser minoritária.

Activation

When you finish filling out the canvas above, now you can begin implementing your data science workflow in roughly this order.

1 Problem Statement → 2 Data Acquisition → 3 Data Prep → 4 Modeling → 5 Outcomes/Preds → 6 Model Eval

* Note: This canvas is intended to be used as a starting point for your data science projects. Data science workflows are typically nonlinear.

8. Links

Link para o vídeo: https://www.youtube.com/watch?v=_jtBEps-Og8

Link para o repositório: https://github.com/l7vieira/TCC_FIES

REFERÊNCIAS

GOV.BR Disponível em < <https://www.gov.br/pt-br/servicos/obter-financiamento-do-fies> >
Acesso em: 30 out. 2022

APÊNDICE

Programação/Scripts

PUC MINAS

TCC - PÓS GRADUAÇÃO EM CIÊNCIA DE DADOS E BIG DATA

PREDIÇÃO DE APROVAÇÃO DO FIES COM NOTAS DO ENEM E IDHM

LUCAS SEVERIANO VIEIRA

▼ Bibliotecas/Constantes

```
[318] # Import

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# ML

from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import resample
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
# Função para montagem gráficos
def plot_chart(desired_variable, df, order, title=None, xlabel=None, ylabel=None,
               hue=None, ax=None):
    sns.countplot(y=desired_variable,
                  hue=hue,
                  data=df,
                  palette='rocket',
                  order=order,
                  ax=ax).set_title(title)
    plt.xlabel(xlabel=xlabel)
    plt.ylabel(ylabel=ylabel)
    plt.grid(axis='x', linestyle='--')
    plt.minorticks_on()
```

```
[320] # Função para retirar acentos
def corrigir_nomes(nome):
    nome = nome.replace('ç', '').replace('ç', 'c').replace('ã', 'a').replace('õ', 'o').replace('ä', 'a').replace('é', 'e').replace('ö', 'o').replace('á', 'a').replace('ê', 'e').replace('í', 'i').replace('ó', 'o').replace('ú', 'u')
    return nome
```

▼ Carga e tratamento dos dados

```
[321] # Carga dos dados - FIES
df_fies = pd.read_csv('relatorio_inscricao_dados_abertos_fies_22021.csv', encoding='ISO-8859-1', sep = ';')
df_fies.shape
```

```
[322] # Carga dos dados - Cidades
df_cidades = pd.read_excel('cities_brazil_1806.xlsx')
df_cidades.shape
```

✓ [323] # Leitura dos dados - FIES

```
df_fies.head()
```

✓ [324] # Leitura dos dados - Cidades

```
df_cidades.head()
```

✓ [325] # Ajuste cidades da base do FIES

```
df_fies['Município de residência'] = df_fies['Município de residência'].apply(corrigir_nomes)

df_fies.replace(to_replace = "EMBU", value = "EMBU DAS ARTES", inplace=True)
df_fies.replace(to_replace = "MOJI MIRIM", value = "MOGI MIRIM", inplace=True)
df_fies.replace(to_replace = "JI-PARANA", value = "JI PARANA", inplace=True)
df_fies.replace(to_replace = "SANTA BARBARA D'OESTE", value = "SANTA BARBARA D OESTE", inplace=True)
df_fies.replace(to_replace = "LIVRAMENTO DE NOSSA SENHORA", value = "LIVRAMENTO DO BRUMADO", inplace=True)
df_fies.replace(to_replace = "GUAJARA-MIRIM", value = "GUAJARA MIRIM", inplace=True)
df_fies.replace(to_replace = "IGARAPE-MIRI", value = "IGARAPE-MIRIM", inplace=True)
df_fies.replace(to_replace = "NOVA BRASILANDIA D'OESTE", value = "NOVA BRASILANDIA D OESTE", inplace=True)
df_fies.replace(to_replace = "ESPIGAO D'OESTE", value = "ESPIGAO D OESTE", inplace=True)
df_fies.replace(to_replace = "PINGO-D'AGUA", value = "PINGO D'AGUA", inplace=True)
df_fies.replace(to_replace = "COLORADO DO OESTE", value = "COLORADO D OESTE", inplace=True)
df_fies.replace(to_replace = "ITABIRINHA", value = "ITABIRINHA DE MANTENA", inplace=True)
df_fies.replace(to_replace = "PEROLA D'OESTE", value = "PEROLA D OESTE", inplace=True)
df_fies.replace(to_replace = "SAO MIGUEL DO GOSTOSO", value = "SAO MIGUEL DE TOUROS", inplace=True)
df_fies.replace(to_replace = "BALNEARIO PICARRAS", value = "PICARRAS", inplace=True)
df_fies.replace(to_replace = "SAO JORGE D'OESTE", value = "SAO JORGE D OESTE", inplace=True)
df_fies.replace(to_replace = "PRESIDENTE CASTELO BRANCO", value = "PRESIDENTE CASTELO BRANCO", inplace=True)
df_fies.replace(to_replace = "CAMPO DE SANTANA", value = "TACIMA", inplace=True)
```

✓ [326] # Merge dos Datasets

```
df_dados = pd.merge(df_fies, df_cidades, how='left', left_on=['Município de residência', 'UF de residência'], right_on = ['LocalCidade', 'LocalUF'])
df_dados.shape
```

✓ [327] # Leitura dos dados - Dados agrupados

```
df_dados.head()
```

✓ [328] # Informações sobre tipo dos dados

```
df_dados.info()
```

✓ [329] # Deleção das colunas não utilizadas

```
df_dados = df_dados.drop(['Nome mantenedora', 'Natureza Jurídica Mantenedora', 'CNPJ da mantenedora', 'Código e-MEC da Mantenedora', 'Nome da IES', 'Código e-MEC da IES', 'Organização Acadêmica da IES'], axis=1)
df_dados = df_dados.drop(['Município da IES', 'UF da IES', 'Nome do Local de oferta', 'Código do Local de Oferta', 'Município do Local de Oferta', 'UF do Local de Oferta', 'Grau', 'Conceito'], axis=1)
df_dados = df_dados.drop(['Região grupo de preferência', 'UF', 'Cod.Microrregião', 'Microrregião', 'Cod.Mesoregião', 'Mesoregião', 'Conceito de curso do GP', 'Subárea do conhecimento', 'Área do conhecimento'], axis=1)
df_dados = df_dados.drop(['Opções de cursos da inscrição', 'Código do curso', 'Percentual de financiamento', 'Semestre do financiamento', 'Qtde semestre financiado', 'IBGECode'], axis=1)
df_dados = df_dados.drop(['RegiãoBrasil', 'Latitude', 'Longitude', 'Gentílico', 'Ano do processo seletivo', 'Semestre do processo seletivo', 'LocalUF', 'Município de residência', 'Data de Nascimento'], axis=1)
df_dados = df_dados.drop(['Professor rede pública ensino?', 'Ano do Enem', 'ReceitasRealizadas_2014', 'DespesasEmpenhadas_2014', 'PopEstimada_2018', 'PopCenso 2010'], axis=1)
df_dados = df_dados.drop(['Tipo de escola no ensino médio', 'UF de residência', 'Cod. do Grupo de preferência', 'Ano conclusão ensino médio', 'ID do estudante', 'Pib_2014'], axis=1)
df_dados = df_dados.drop(['Renda familiar mensal bruta', 'Pessoa com deficiência', 'Concluiu curso superior?', 'Nº de membros Grupo Familiar'], axis=1)
```

✓ [330] # Informações sobre tipo dos dados

```
df_dados.info()
```

✓ [331] # Renomeando colunas

```
df_dados.rename(columns={'Etnia/Con': 'Con',
                        'Renda mensal bruta per capita': 'Renda_per_capita',
                        'Média nota Enem': 'Media_Enem',
                        'Matemática e suas Tecnologias': 'Matematica',
                        'Linguagens, Códigos e suas Tec': 'Linguagens',
                        'Ciências Natureza e suas Tec': 'Ciencias_natureza',
                        'Ciências Humanas e suas Tec': 'Ciencias_humanas',
                        'Nome do curso': 'Curso',
                        'Nota Corte Grupo Preferência': 'Nota_corte',
                        'Situação Inscrição Fies': 'Situacao'}, inplace = True)
```

✓ [332] # Análise de dados faltantes

```
df_dados.isnull().sum().sort_values(ascending=False)[:10]
```

✓ [333] # Limpeza de dados nulos

```
df_dados.dropna(subset=["IDHM"], inplace=True)
df_dados.dropna(subset=["LocalEstado"], inplace=True)
df_dados.dropna(subset=["LocalCidade"], inplace=True)
```

✓ [334] # Conversão numéricos

```
df_dados['Nota_corte'] = df_dados['Nota_corte'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['Media_Enem'] = df_dados['Media_Enem'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['Matematica'] = df_dados['Matematica'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['Linguagens'] = df_dados['Linguagens'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['Ciencias_natureza'] = df_dados['Ciencias_natureza'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['Ciencias_humanas'] = df_dados['Ciencias_humanas'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['IDHM'] = df_dados['IDHM'].apply(lambda x: x.replace(",", ".")).astype(float)
df_dados['Renda_per_capita'] = df_dados['Renda_per_capita'].apply(lambda x: x.replace(",", ".")).astype(float)
```

✓ [335] # Conversão categóricos

```
label_encoder = LabelEncoder()
df_dados['Curso'] = label_encoder.fit_transform(df_dados['Curso'])
df_dados['LocalEstado'] = label_encoder.fit_transform(df_dados['LocalEstado'])
df_dados['LocalCidade'] = label_encoder.fit_transform(df_dados['LocalCidade'])
```

```

[336] # Limpeza da coluna alvo, apenas vai ser verificado dados contratados ou não

df_dados.drop(df_dados.loc[df_dados['Situacao']=='INSCRIÇÃO POSTERGADA'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='LISTA DE ESPERA'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='OPÇÃO NÃO CONTRATADA'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='PARTICIPACAO CANCELADA PELO CANDIDATO'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='PRÉ-SELECIONADO'].index, inplace=True)
df_dados.drop(df_dados.loc[df_dados['Situacao']=='REJEITADA PELA CPSA'].index, inplace=True)

# Conversão para numérico
df_dados['Situacao'].replace(['NÃO CONTRATADO', 'CONTRATADA'], [0, 1], inplace=True)

print(df_dados.groupby('Situacao').size())

[337] # Estrutura dos dados

df_dados.info()

[338] plt.figure(figsize=(15, 8))
      plot_chart('Cor', df_dados, df_dados['Cor'].value_counts().index, title='Cor',
                  ylabel='Race', xlabel='Cases', hue='Cor')

[339] plt.figure(figsize=(15, 8))
      plot_chart('Sexo', df_dados, df_dados['Sexo'].value_counts().index, title='Sexo',
                  ylabel='Race', xlabel='Cases', hue='Sexo')

[340] plt.figure(figsize=(15, 8))
      plot_chart('Turno', df_dados, df_dados['Turno'].value_counts().index, title='Turno',
                  ylabel='Race', xlabel='Cases', hue='Turno')

[341] #Exclusão de colunas não utilizadas no modelo
3
      df_dados = df_dados.drop(['Cor', 'Sexo', 'Turno'],axis=1)

[▶] # Análise de Outliers

numeric_columns = df_dados.select_dtypes(np.number).columns

for col in numeric_columns:
    plt.figure(figsize=(10,5))
    sns.boxplot(y=col, data=df_dados)
    plt.title(col)
    plt.ylabel(None)
    plt.show()

[343] # Remoção de outliers e cortes da regra do FIES
5
      df_remove = df_dados.loc[(df_dados['Renda_per_capita'] > 4000) | (df_dados['Media_Enem'] < 450) | (df_dados['Redação'] < 400) ]
      df_dados = df_dados.drop(df_remove.index)

[344] print(df_dados.hist(column = ['Renda_per_capita']))

[345] print(df_dados.hist(column = ['Nota_corte']))

[346] print(df_dados.hist(column = ['Media_Enem']))

[347] print(df_dados.hist(column = ['Redação']))

[348] print(df_dados.hist(column = ['Matematica']))

```

```

✓ [349] print(df_dados.hist(column = ['Linguagens']))

✓ [350] print(df_dados.hist(column = ['Ciencias_natureza']))

✓ [351] print(df_dados.hist(column = ['Ciencias_humanas']))

✓ [352] print(df_dados.hist(column = ['IDHM']))

✓ [353] # Normalização dos dados

        df_max_scaled = df_dados.copy()
        for column in df_max_scaled.columns:
            df_max_scaled[column] = df_max_scaled[column] / df_max_scaled[column].abs().max()

        display(df_max_scaled)

✓ [354] #Estatística dos dados

        df_dados.describe()

```

ML

```

✓ [355] # Análise correlação

        plt.figure(figsize=(10, 7))
        sns.heatmap(df_dados.corr(),
                    annot = True,
                    fmt = '.2f',
                    cmap='Blues')
        plt.title('Correlação entre as variáveis')
        plt.show()

✓ [356] # Verificando balanceamento da coluna alvo

        balance = df_dados['Situacao'].value_counts(normalize=True).round(2)
        balance.index = balance.index.map({0:'NÃO CONTRATADO', 1:'CONTRATADA'})
        balance * 100

✓ [357] # Total não efetivados
        df_maior = df_dados[df_dados['Situacao'] == 0]
        print('linhas:', df_maior.shape[0])

✓ [358] # Total efetivados
        df_menor = df_dados[df_dados['Situacao'] == 1]
        print('linhas:', df_menor.shape[0])

```

```

✓ [359] # Igualando quantidade de registros SIM/NÃO
df_unsampled = resample(df_maior, replace = True, n_samples=21603, random_state=123)

✓ [360] # Concatenando registros SIM/NÃO
df_dados = pd.concat([df_unsampled, df_menor])
df_dados.info()

✓ [361] # Análise balanceamento

import plotly.express as px
plot_ = df_dados['Situacao'].map({1:'CONTRATADA', 0:'NÃO CONTRATADO'}).value_counts().reset_index()
fig = px.bar(plot_,
             x='index',
             y='Situacao',
             labels={'index':'Alvo', 'Situacao':''},
             title='Balanceamento do Alvo',
             width=600)
fig.update_xaxes(type='category')
fig.show()

✓ [362] # Separação dos Dados - Dataframe
Var_Caracteristicas = df_dados.drop(columns='Situacao')
Var_Previsao = df_dados['Situacao']

# Separação dos Dados - Treino/Teste
x_treino, x_teste, y_treino, y_teste = train_test_split(Var_Caracteristicas, Var_Previsao, test_size=0.20, random_state=10)

print(f'Dados de treino: {len(x_treino)} ')
print(f'Dados de teste: {len(x_teste)} ')

```

Regressão Logística

```

✓ [372] # Criação do modelo

logistic_regression = LogisticRegression(random_state= 42, max_iter=10000)
logistic_regression.fit(x_treino, y_treino)

# Classificação
Train_predict = logistic_regression.predict(x_teste)
print(classification_report(y_teste, Train_predict))

# Acurácia
resultado = logistic_regression.score(x_teste, y_teste)
print('Acurácia:', resultado)

✓ [373] # Matriz de Confusão - Regressão Logística

fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_teste, Train_predict), annot=True,
            ax=ax, fmt='d', cmap='Reds')
ax.set_title("Matriz de Confusão - Regressão Logística", fontsize=18)
ax.set_ylabel("True label")
ax.set_xlabel("Predicted Label")
plt.tight_layout()

✓ [374] # Curva ROC - Regressão Logística

y_pred_probability = logistic_regression.predict_proba(x_teste)[:,1]
fpr, tpr, _ = metrics.roc_curve(y_teste, y_pred_probability)
auc = metrics.roc_auc_score(y_teste, y_pred_probability)
plt.plot(fpr,tpr,label="Logistic Regression, auc="+str(auc))
plt.legend(loc=4)
plt.show()

```

Árvore de decisão

✓ [375] # Criação do modelo

```
tree = DecisionTreeClassifier()
tree.fit(x_treino, y_treino)

# Classificação
Train_predict = tree.predict(x_teste)
print(classification_report(y_teste, Train_predict ))

# Acurácia
resultado = tree.score(x_teste, y_teste)
print('Acurácia:', resultado)
```

✓ [377] # Matriz de Confusão - Árvore de decisão

```
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_teste, Train_predict), annot=True,
            ax=ax, fmt='d', cmap='Reds')
ax.set_title("Matriz de Confusão - Árvore de decisão", fontsize=18)
ax.set_ylabel("True label")
ax.set_xlabel("Predicted Label")
plt.tight_layout()
```

✓ [378] # Curva ROC - Árvore de decisão

```
y_pred_probability = tree.predict_proba(x_teste)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_teste, y_pred_probability)
auc = metrics.roc_auc_score(y_teste, y_pred_probability)
plt.plot(fpr,tpr,label="Tree, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

Random Forest

✓ [379] # Criação do modelo

```
random_forest = RandomForestClassifier(random_state= 42)
random_forest.fit(x_treino, y_treino)

# Classificação
Train_predict = random_forest.predict(x_teste)
print(classification_report(y_teste, Train_predict))

# Acurácia
resultado = random_forest.score(x_teste, y_teste)
print('Acurácia:', resultado)
```


✓ [381] # Matriz de Confusão - Random Forest

```
fig, ax = plt.subplots()
sns.heatmap(confusion_matrix(y_teste, Train_predict), annot=True,
            ax=ax, fmt='d', cmap='Reds')
ax.set_title("Matriz de Confusão - Random Forest", fontsize=18)
ax.set_ylabel("True label")
ax.set_xlabel("Predicted Label")
plt.tight_layout()
```

✓  # Curva ROC - Random Forest

```
y_pred_probability = random_forest.predict_proba(x_teste)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_teste, y_pred_probability)
auc = metrics.roc_auc_score(y_teste, y_pred_probability)
plt.plot(fpr,tpr,label="Random Forest, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```