



APLICAÇÃO DE APRENDIZADO DE MÁQUINA NA CLASSIFICAÇÃO DE EXTINÇÃO DE CURSOS DE GRADUAÇÃO NO BRASIL

Francisco Willem Romão Moreira



PROBLEMA

PROBLEMA

- Instituições de ensino enfrentam o desafio de lidar com a possível descontinuidade de cursos.
- A falta de previsibilidade dificulta a gestão e o planejamento.





DATASET

DATASET - Origem dos Dados

Portal de
Dados Abertos
DO MINISTÉRIO DA EDUCAÇÃO

Buscar no portal 

Dados abertos do Governo Federal | Portal MEC

[PÁGINA INICIAL](#) > [INDICADORES SOBRE ENSINO SUPERIOR](#) > [CURSOS DE GRADUAÇÃO DO BRASIL](#)

Conheça o Plano de
Dados Abertos do
MEC para o biênio
2020-2022

Cursos de Graduação do Brasil

Publicado: Quinta, 29 de Dezembro de 2022, 15h17 | Última atualização em Quinta, 29 de Dezembro de 2022, 20h13 | Acessos: 4716

 Tweetar

 Compartilhar

CONJUNTOS DE DADOS

Bolsa Formação

Brasil na Escola

EMTI

EPT

FIES

ID Estudantil

URL: https://dadosabertos.mec.gov.br/images/conteudo/Ind-ensino-superior/2022//PDA_Dados_Cursos_Graduacao_Brasil.csv 

Detalhamento do quantitativo de Cursos de Graduação (Licenciatura, Bacharelado, Tecnológico, Sequencial e ABI - Área Básica de Ingresso) no Brasil por: código da Instituição de Educação Superior (IES); nome da IES; categoria da IES; organização acadêmica; código do curso; nome do curso; grau; área OCDE; modalidade de ensino (presencial ou EaD); situação do curso (ativo ou inativo); vagas autorizadas; carga horária; segmentadas por código do município (IBGE); município; UF; região.

DATASET - Informações Gerais

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 902676 entries, 0 to 902675
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   CODIGOIES        902676 non-null   int64  
 1   NOMEIES          902676 non-null   object  
 2   CATEGORIAADMINISTRATIVA 902676 non-null   object  
 3   ORGANIZACAOACADEMICA    902676 non-null   object  
 4   CODIGOCURSO        902676 non-null   int64  
 5   NOMECURSO         902676 non-null   object  
 6   GRAU              902676 non-null   object  
 7   AREAOCDE          680577 non-null   object  
 8   MODALIDADE        902676 non-null   object  
 9   SITUACAOCURSO     902676 non-null   object  
 10  QT_VAGAS_AUTORIZADAS 902676 non-null   int64  
 11  CARGAHORARIA      902676 non-null   int64  
 12  CODIGOAREAOCDECINE 902649 non-null   object  
 13  AREAOCDECINE       902649 non-null   object  
 14  CODIGOMUNICIPIO    902676 non-null   int64  
 15  MUNICIPIO         902676 non-null   object  
 16  UF                902676 non-null   object  
 17  REGIAO            902676 non-null   object  
dtypes: int64(5), object(13)
memory usage: 124.0+ MB
```



LIMPEZA

LIMPEZA - Valores nulos

```
# Total de valores ausentes por coluna  
  
print(df.isna().sum())
```

CODIGOIES	0
NOMEIES	0
CATEGORIAADMINISTRATIVA	0
ORGANIZACAOACADEMICA	0
CODIGOCURSO	0
NOMECURSO	0
GRAU	0
AREAOCDE	222099
MODALIDADE	0
SITUACOOCURSO	0
QT_VAGAS_AUTORIZADAS	0
CARGA_HORARIA	0
CODIGOAREAOCDECINE	27
AREAOCDECINE	27
CODIGOMUNICIPIO	0
MUNICIPIO	0
UF	0
REGIAO	0
dtype:	int64



```
# Remover colunas com valores ausentes  
  
df = df.dropna(axis=1)
```

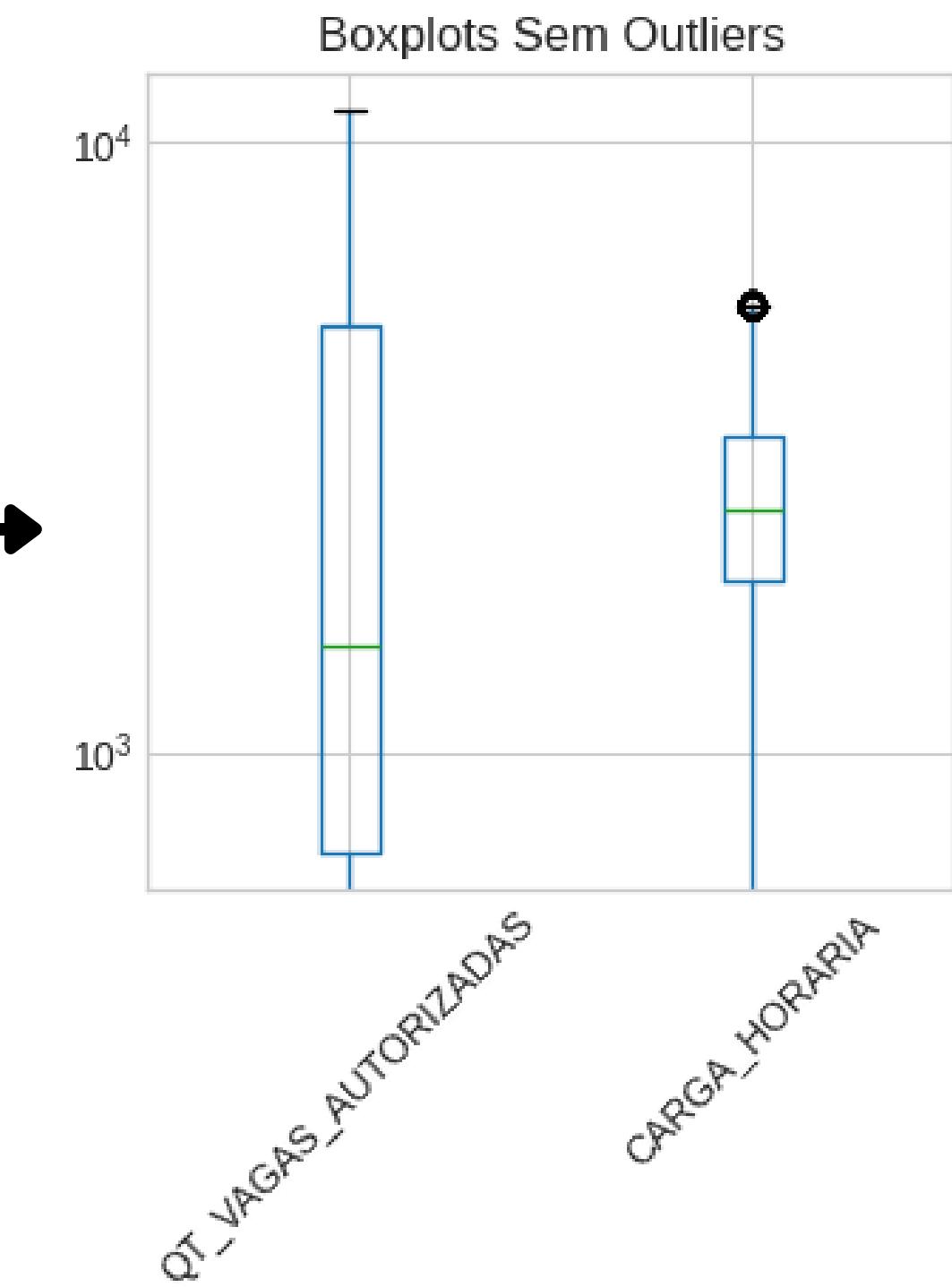
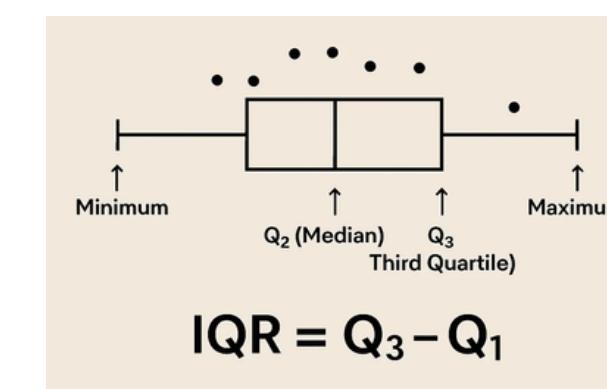
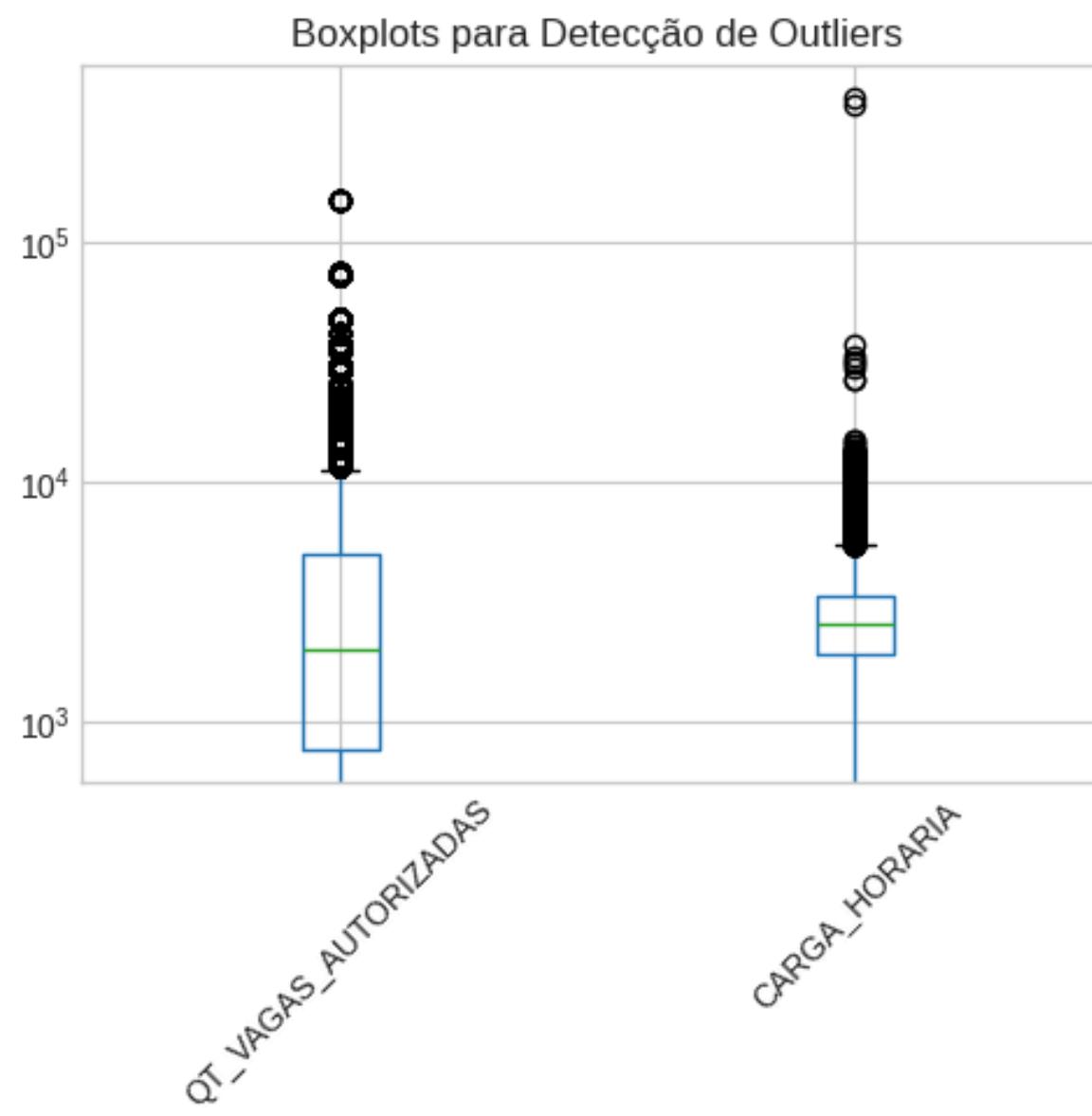
LIMPEZA - Registros duplicados

```
# Total de linhas duplicadas  
print(df.duplicated().sum())
```



```
# Removendo linhas duplicadas  
df = df.drop_duplicates()
```

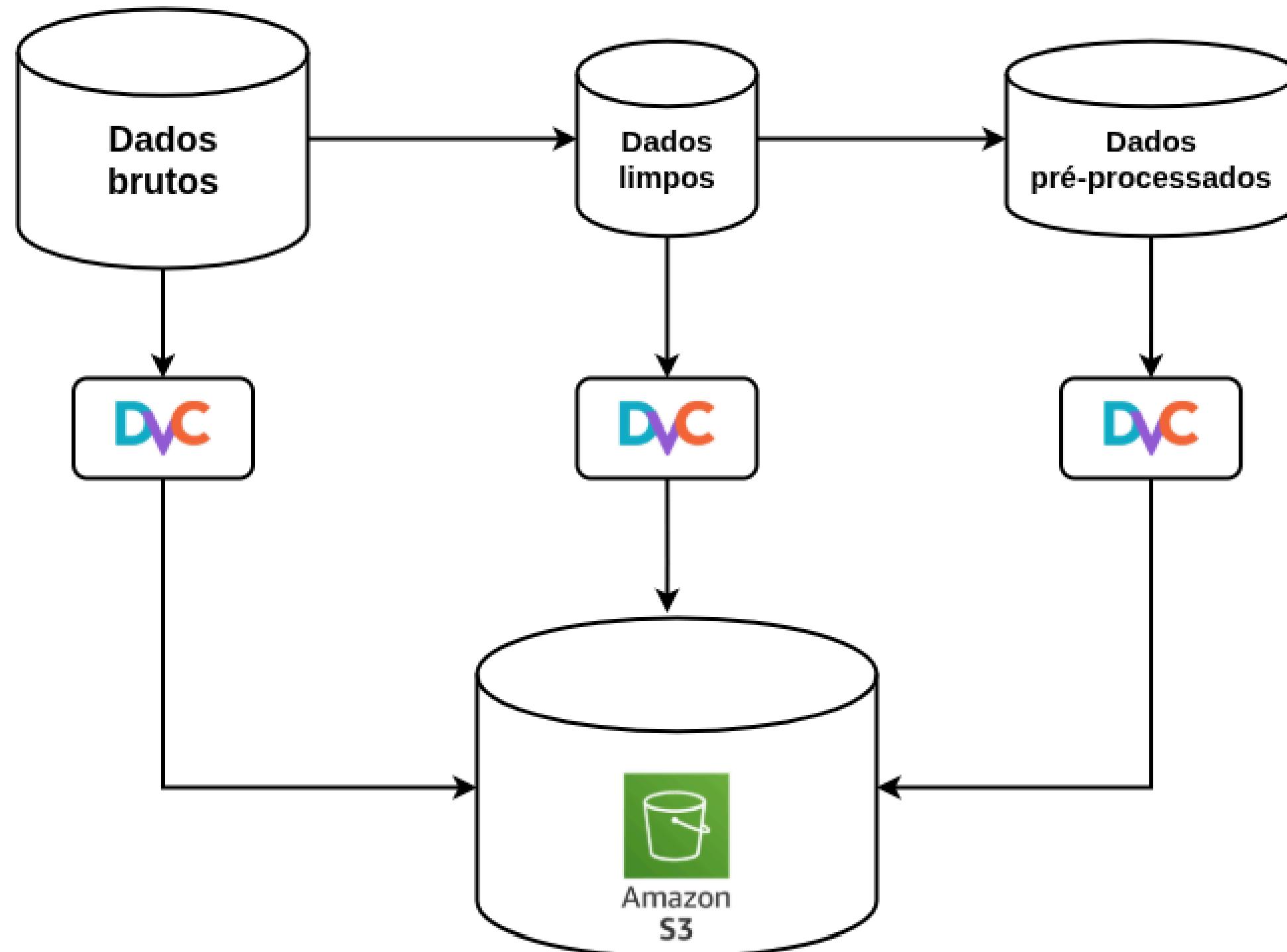
LIMPEZA - Ruídos/Outliers



LIMPEZA - Remoção de variáveis

```
# Remover colunas não necessárias  
  
columns_to_remove = ['CODIGOIES', 'NOMEIES', 'CODIGOCURSO', 'CODIGOMUNICIPIO']  
  
df = df.drop(columns=columns_to_remove)  
  
df.columns  
  
Index(['CATEGORIAADMINISTRATIVA', 'ORGANIZACAOACADEMICA', 'NOMECURSO',  
       'GRAU', 'MODALIDADE', 'SITUACOACURSO', 'QT_VAGAS_AUTORIZADAS',  
       'CARGA_HORARIA', 'MUNICIPIO', 'UF', 'REGIAO'],  
      dtype='object')
```

VERSIONAMENTO DOS DADOS





ANÁLISE EXPLORATÓRIA

ANÁLISE EXPLORATÓRIA

```
# Criar relatório de análise exploratória de dados com ydata_profiling  
  
profile = ProfileReport(df, title="Ánalise Exploratória de Dados - Cursos de Graduação Brasil")  
  
profile.to_file("relatorio_dados.html")
```

```
Summarize dataset: 0% | 0/5 [00:00<?, ?it/s]  
100% | 11/11 [00:13<00:00, 1.26s/it]  
Generate report structure: 0% | 0/1 [00:00<?, ?it/s]  
Render HTML: 0% | 0/1 [00:00<?, ?it/s]  
Export report to file: 0% | 0/1 [00:00<?, ?it/s]
```

sex 11 de jul 23:50

Ánalise Exploratória de Dados - Cursos de Graduação Brasil

Overview Variables Interactions Correlations Missing values Sample Duplicate rows

Overview

Brought to you by YData

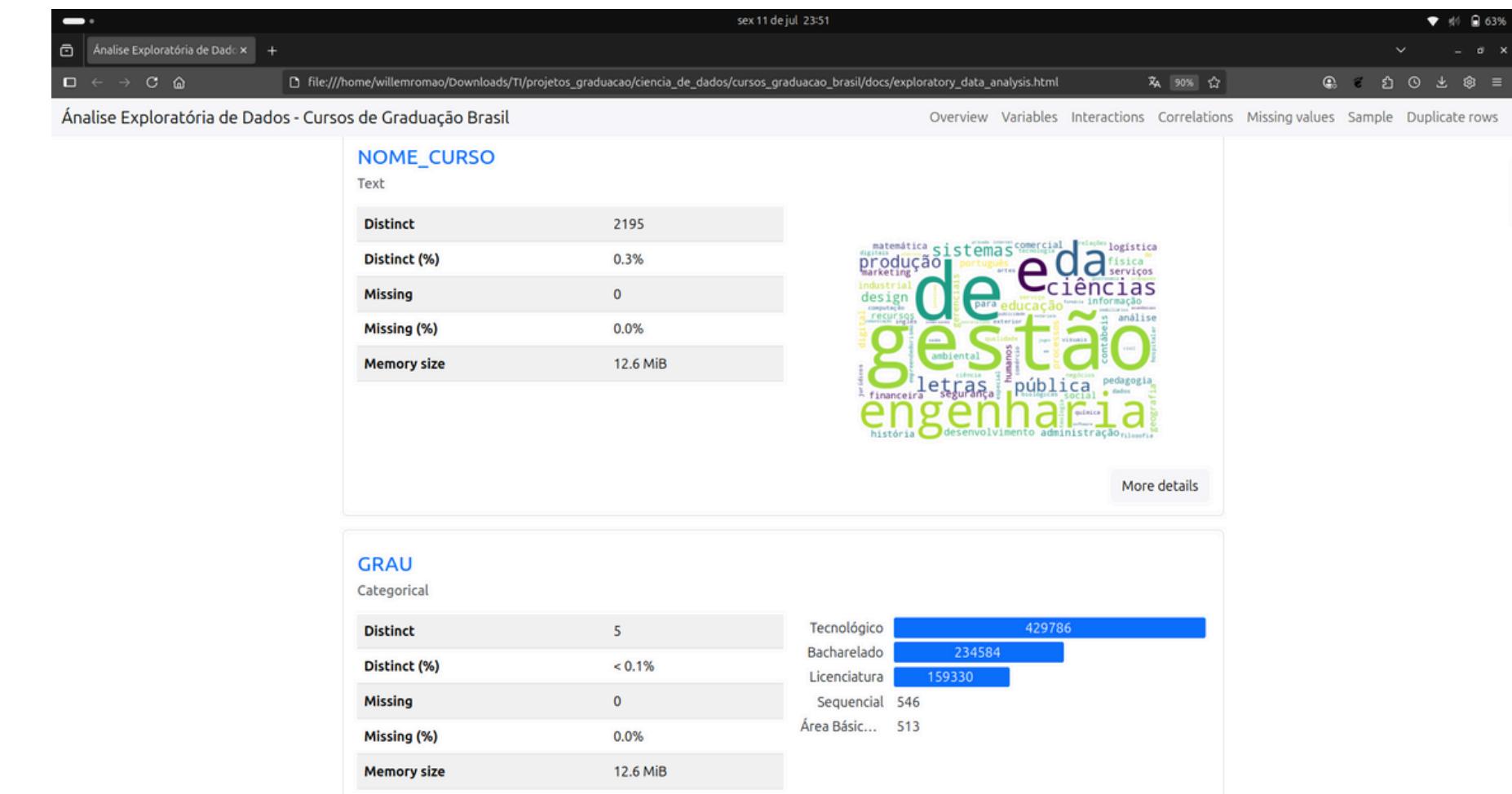
Overview Alerts 7 Reproduction

Dataset statistics

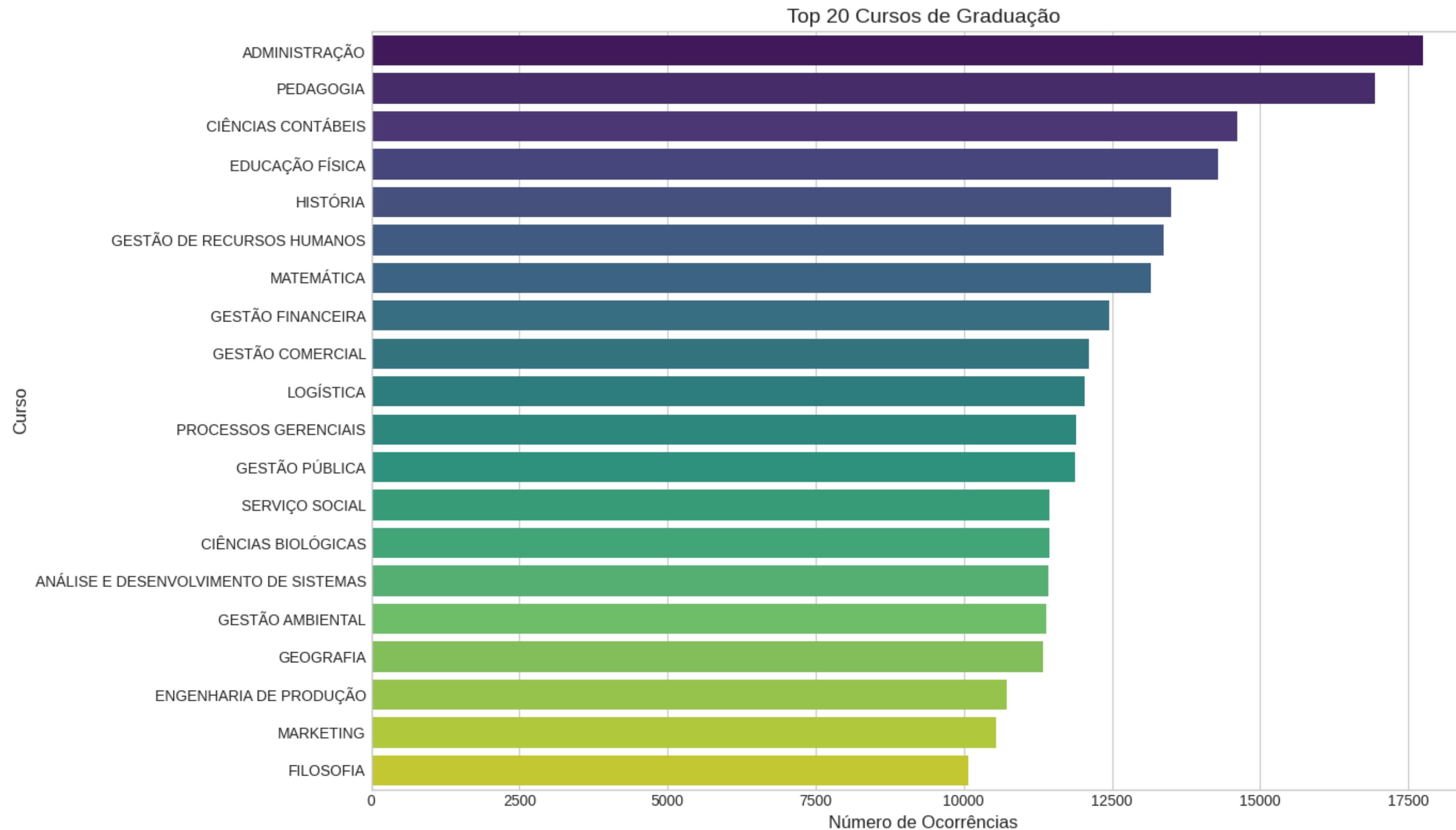
Number of variables	11
Number of observations	824759
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	9142
Duplicate rows (%)	1.1%
Total size in memory	75.5 MiB
Average record size in memory	96.0 B

Variable types

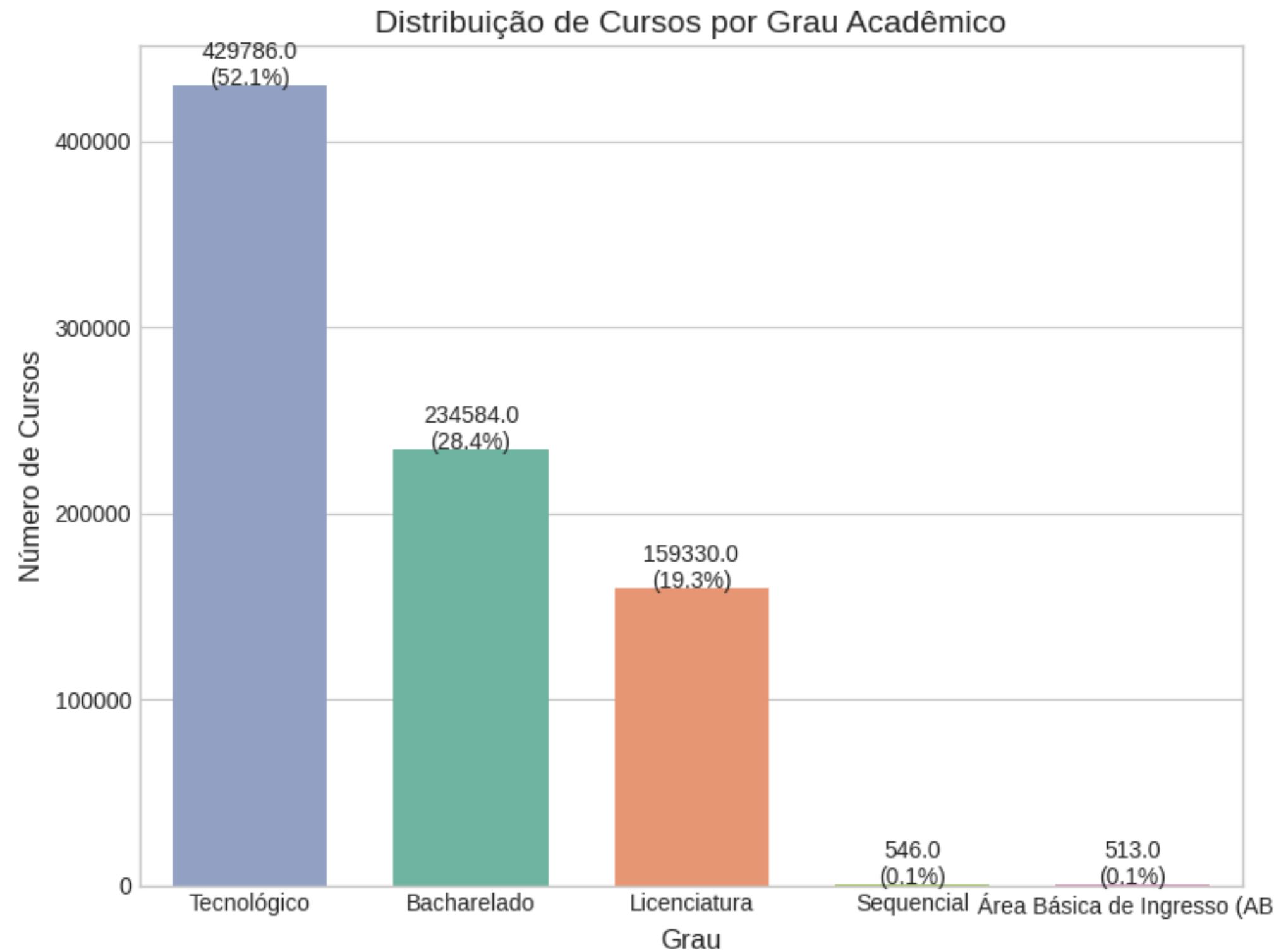
Text	2
Categorical	7
Numeric	2



ANÁLISE EXPLORATÓRIA

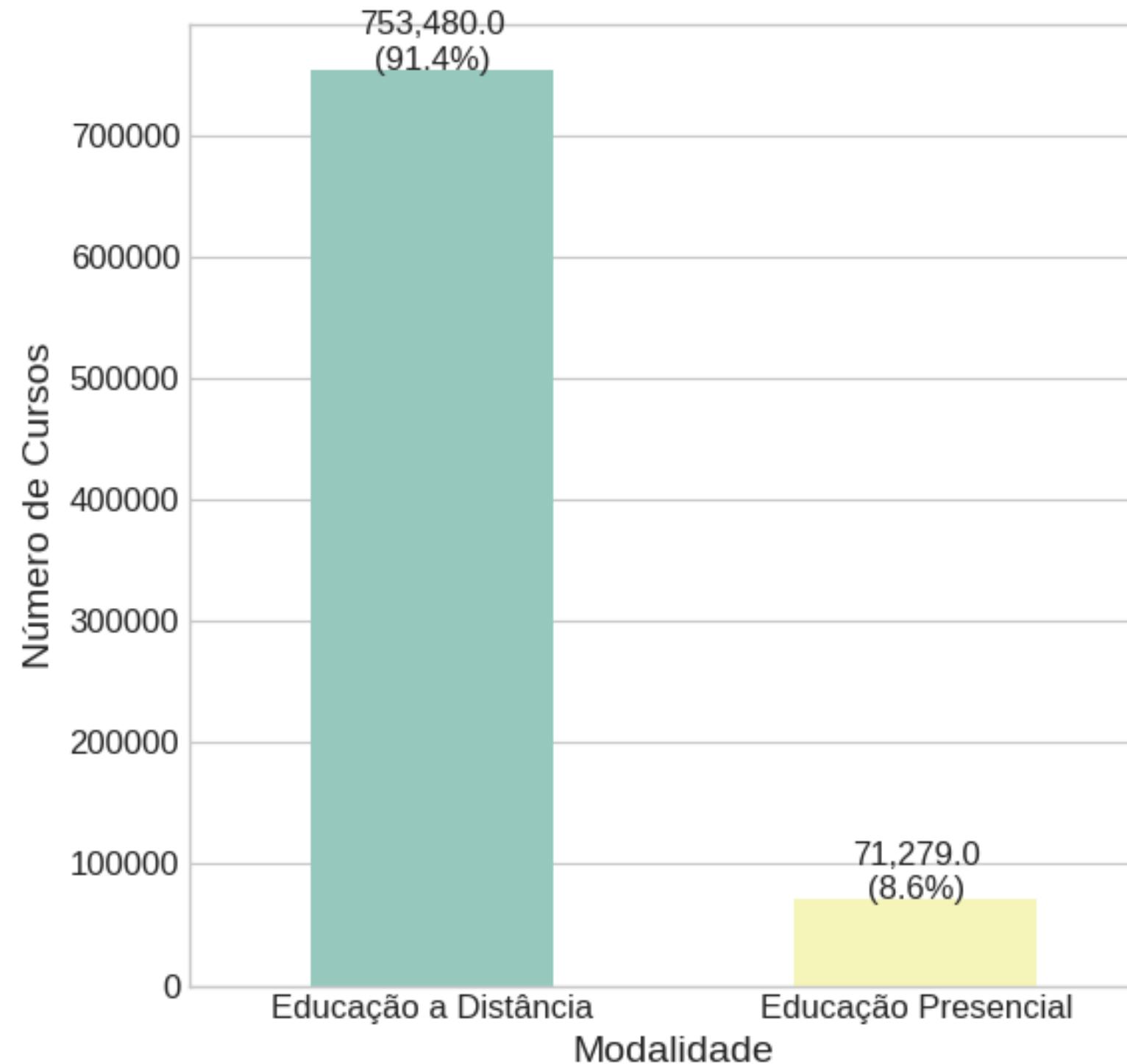


ANÁLISE EXPLORATÓRIA

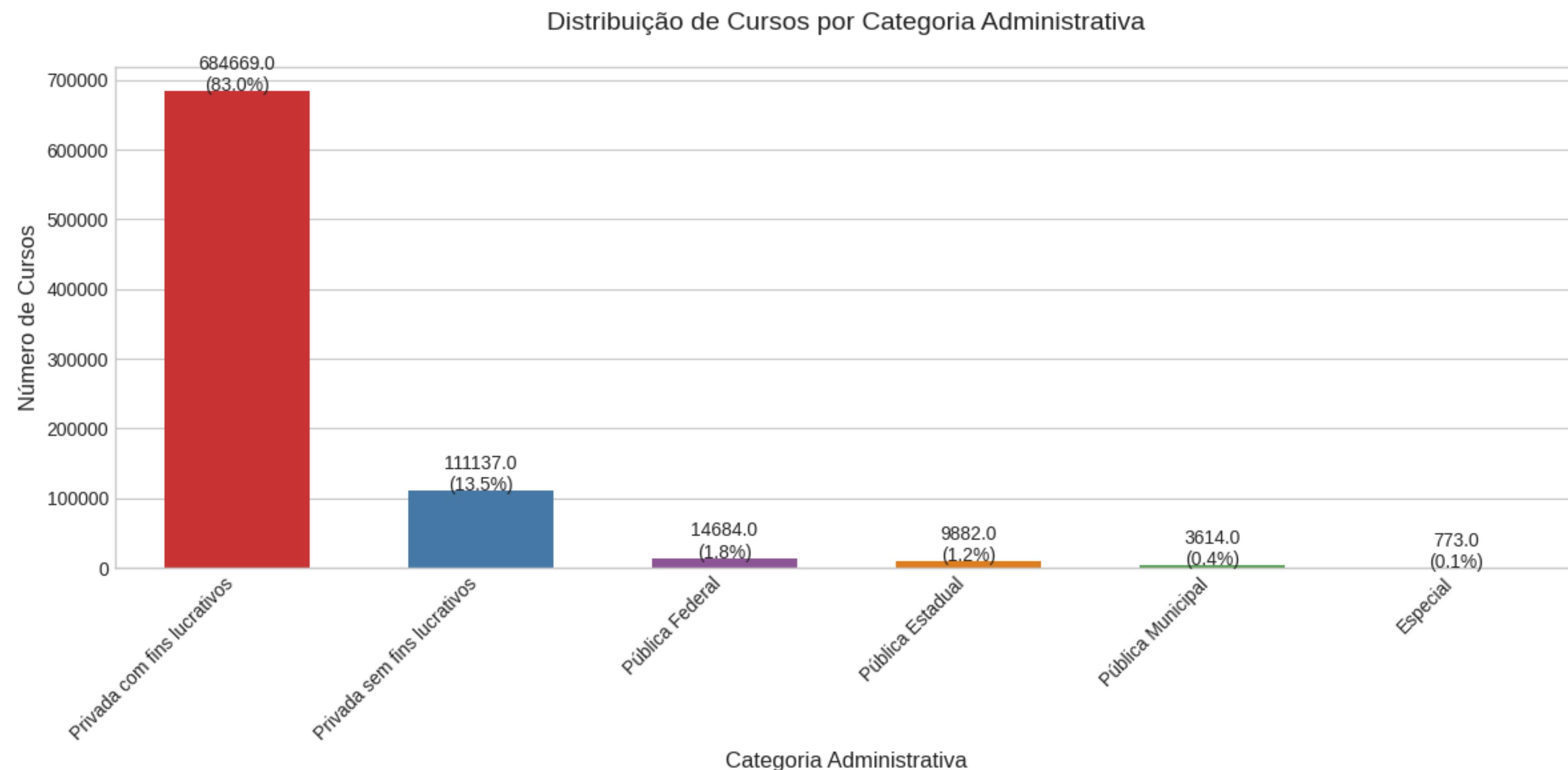


ANÁLISE EXPLORATÓRIA

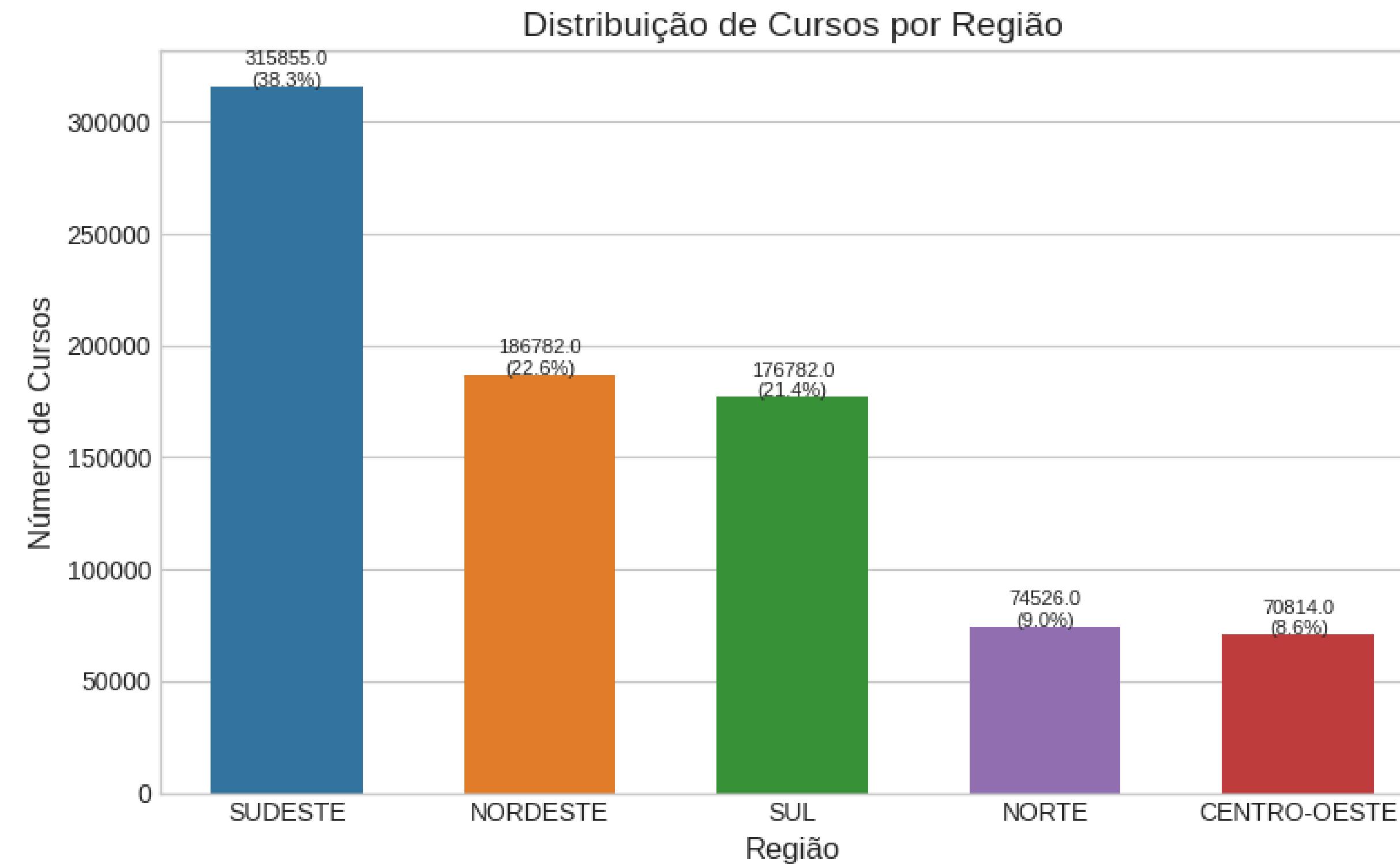
Distribuição de Cursos por Modalidade



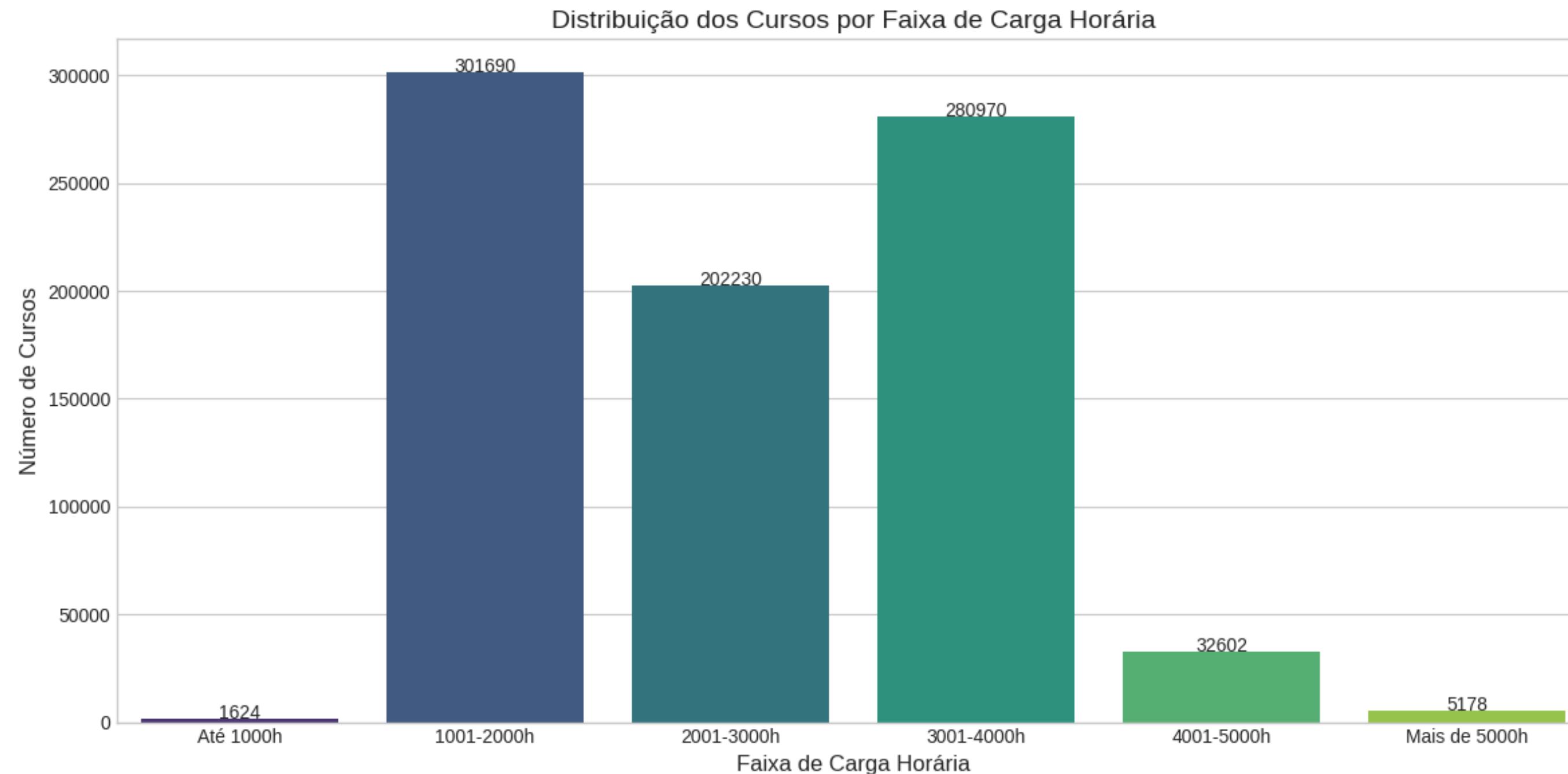
ANÁLISE EXPLORATÓRIA



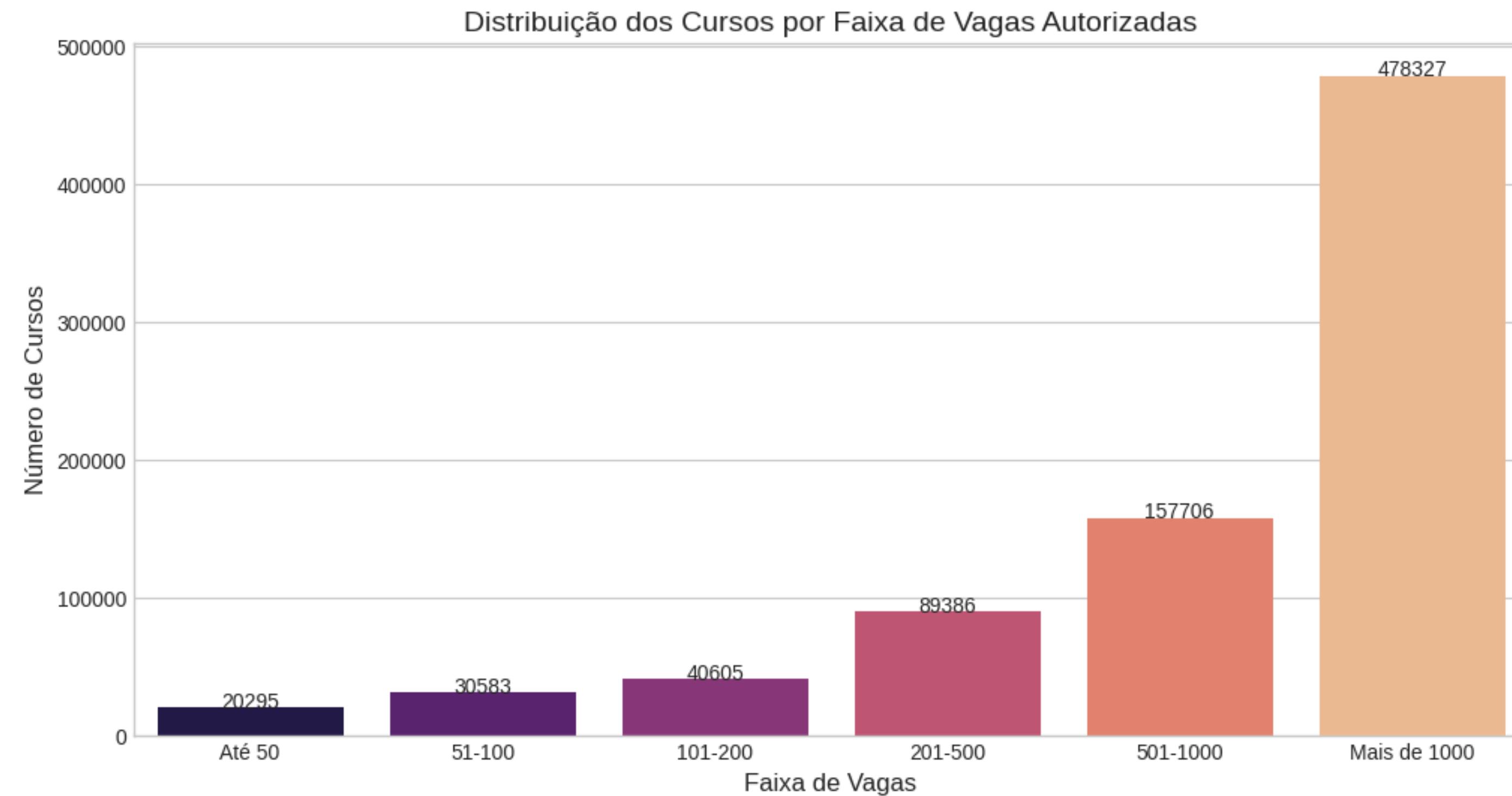
ANÁLISE EXPLORATÓRIA



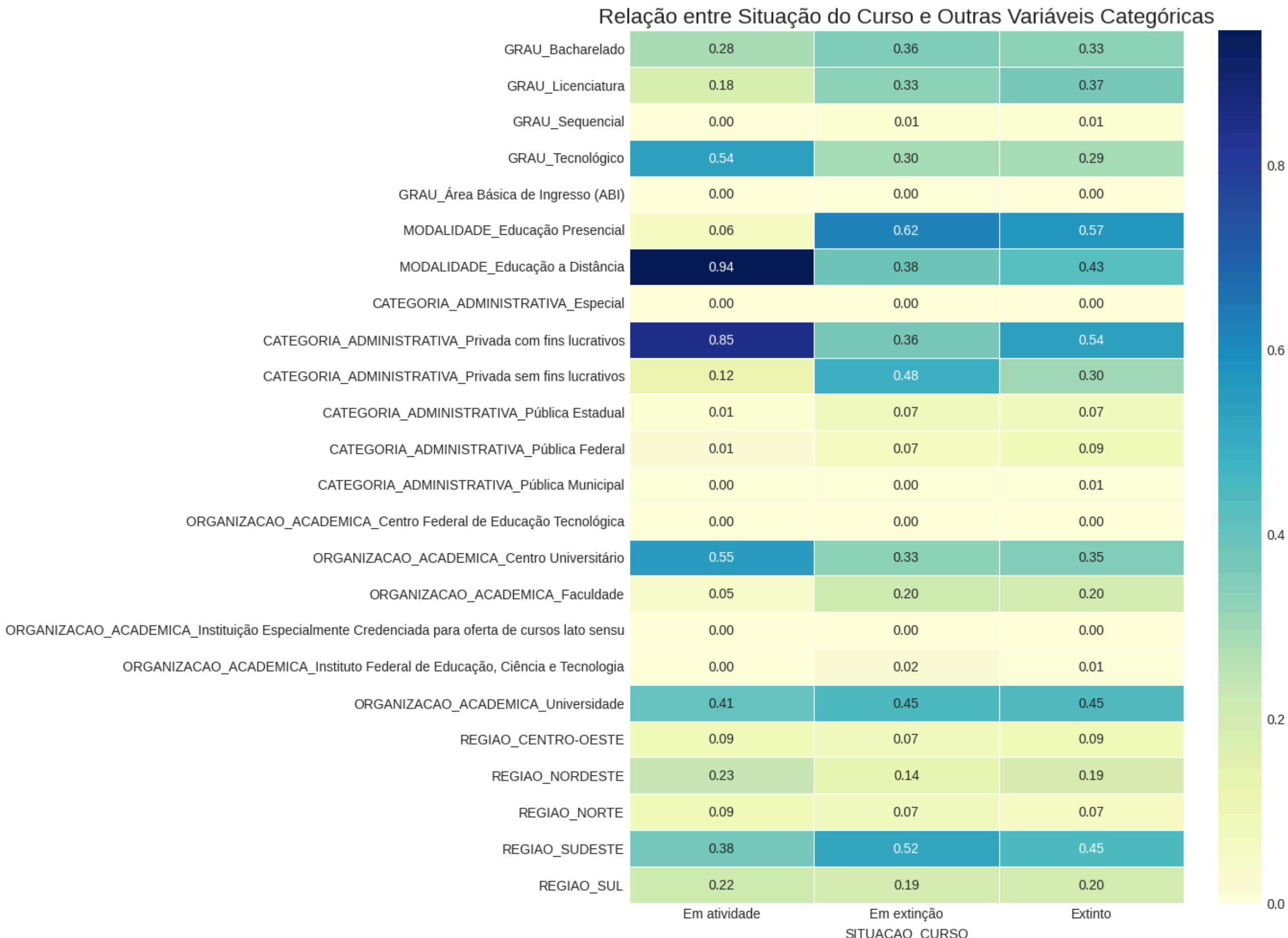
ANÁLISE EXPLORATÓRIA



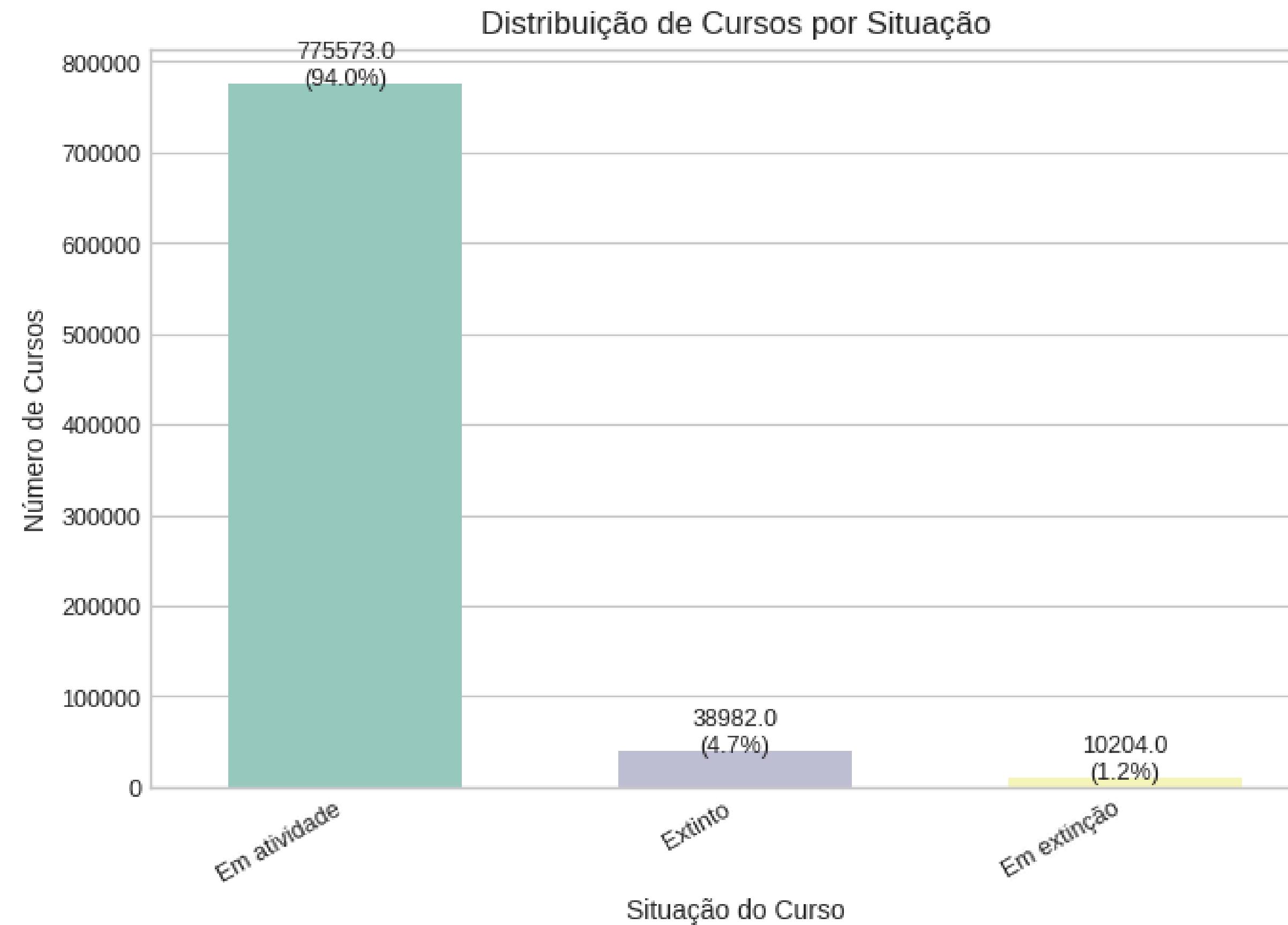
ANÁLISE EXPLORATÓRIA



ANÁLISE EXPLORATÓRIA



ANÁLISE EXPLORATÓRIA





PRÉ-PROCESSAMENTO

PRÉ-PROCESSAMENTO - Transformação da variável alvo

```
# Tornar feature alvo em um problema binário ao invés de multiclasse

print("Categorias de SITUACAO_CURSO antes da transformação:")
print(df['SITUACAO_CURSO'].unique())

df = df.rename(columns={'SITUACAO_CURSO': 'EXTINTO'})

df['EXTINTO'] = df['EXTINTO'].replace(['Em extinção', 'Extinto'], 'Sim')
df['EXTINTO'] = df['EXTINTO'].replace(['Em atividade'], 'Não')

print("\nCategorias depois da transformação:")
print(df['EXTINTO'].unique())
```

Categorias de SITUACAO_CURSO antes da transformação:
['Em atividade' 'Em extinção' 'Extinto']

Categorias depois da transformação:
['Não' 'Sim']

PRÉ-PROCESSAMENTO - Discretização das variáveis numéricas

```
# Discretização de CARGA_HORARIA

bins_carga = [0, 1000, 2000, 3000, 4000, 5000, float('inf')]
labels_carga = ['Até 1000h', '1001-2000h', '2001-3000h', '3001-4000h', '4001-5000h', 'Mais de 5000h']

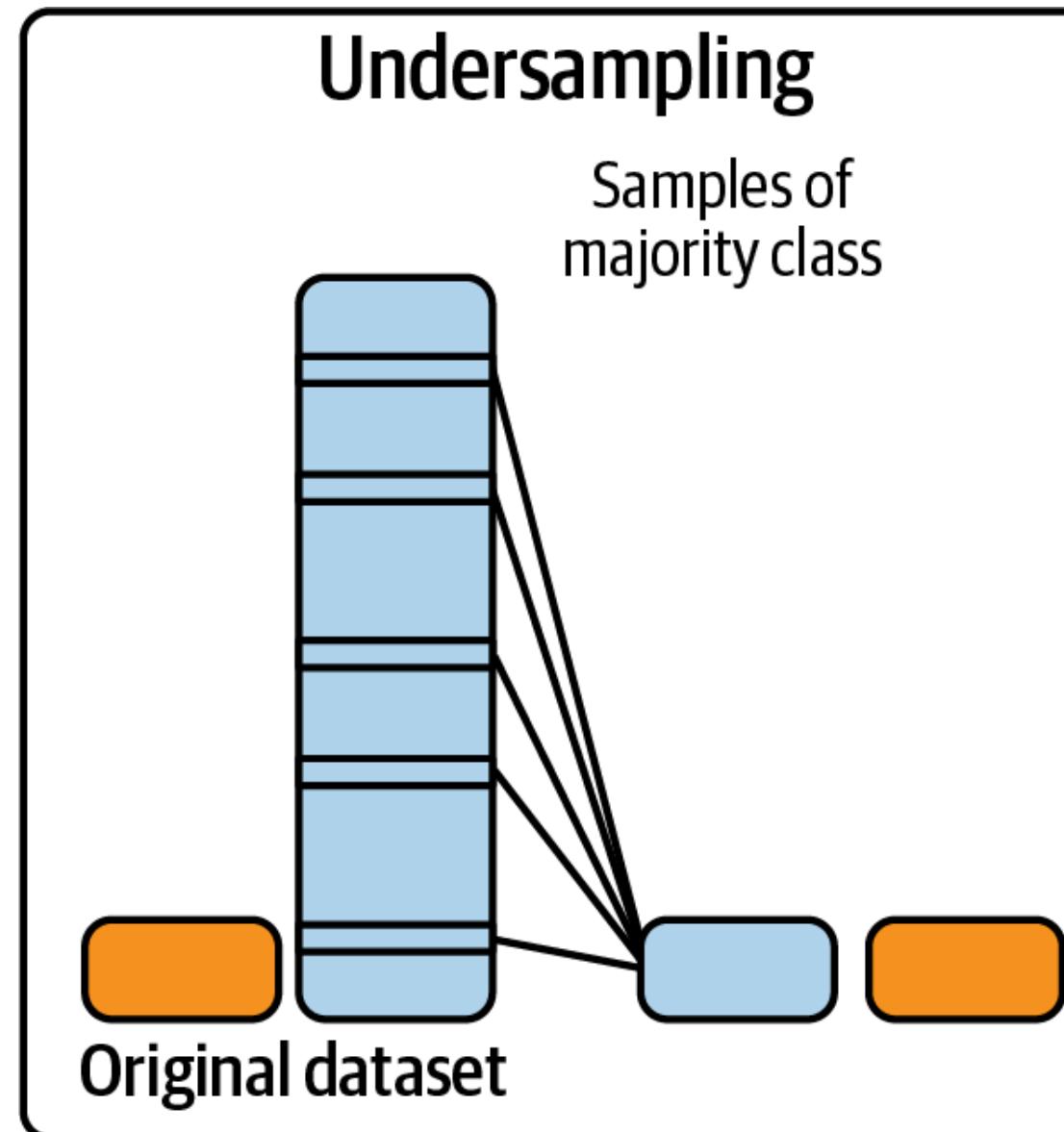
df['CARGA_HORARIA'] = pd.cut(
    df['CARGA_HORARIA'],
    bins=bins_carga,
    labels=labels_carga
)
```

```
# Discretização de QT_VAGAS_AUTORIZADAS

bins_vagas = [0, 50, 100, 200, 500, 1000, float('inf')]
labels_vagas = ['Até 50', '51-100', '101-200', '201-500', '501-1000', 'Mais de 1000']

df['QT_VAGAS_AUTORIZADAS'] = pd.cut(
    df['QT_VAGAS_AUTORIZADAS'],
    bins=bins_vagas,
    labels=labels_vagas
)
```

PRÉ-PROCESSAMENTO - Balanceamento das classes



Distribuição original:

EXTINTO

Não 774923

Sim 44746

Name: count, dtype: int64

EXTINTO

Não 0.94541

Sim 0.05459

Name: proportion, dtype: float64

Após balanceamento

Counter({'Não': 44746, 'Sim': 44746})

PRÉ-PROCESSAMENTO - Remoção de variáveis

```
# Remoção de variáveis desnecessárias

print("Antes da remoção das colunas NOME_CURSO, UF e MUNICIPIO:\n")
print(df.columns)

df = df.drop(columns=['NOME_CURSO', 'UF', 'MUNICIPIO'])
print("\nDepois da remoção:\n")
print(df.columns)
```

Antes da remoção das colunas NOME_CURSO, UF e MUNICIPIO:

```
Index(['NOME_CURSO', 'GRAU', 'MODALIDADE', 'CATEGORIA_ADMINISTRATIVA',
       'ORGANIZACAO_ACADEMICA', 'MUNICIPIO', 'UF', 'REGIAO',
       'QT_VAGAS_AUTORIZADAS', 'CARGA_HORARIA', 'EXTINTO'],
      dtype='object')
```

Depois da remoção:

```
Index(['GRAU', 'MODALIDADE', 'CATEGORIA_ADMINISTRATIVA',
       'ORGANIZACAO_ACADEMICA', 'REGIAO', 'QT_VAGAS_AUTORIZADAS',
       'CARGA_HORARIA', 'EXTINTO'],
      dtype='object')
```

PRÉ-PROCESSAMENTO - Visão geral dataset

GRAU
Tecnológico 38035
Bacharelado 27927
Licenciatura 23091
Sequencial 430
Área Básica de Ingresso (ABI) 9
Name: count, dtype: int64

MODALIDADE
Educação a Distância 61190
Educação Presencial 28302
Name: count, dtype: int64

CATEGORIA ADMINISTRATIVA
Privada com fins lucrativos 60845
Privada sem fins lucrativos 21030
Pública Federal 3939
Pública Estadual 3044
Pública Municipal 446
Especial 188
Name: count, dtype: int64

ORGANIZACAO_ACADEMICA
Centro Universitário 40755
Universidade 36730
Faculdade 11447
Instituto Federal de Educação, Ciência e Tecnologia 552
Centro Federal de Educação Tecnológica 8
Name: count, dtype: int64

REGIAO
SUDESTE 37501
SUL 18866
NORDESTE 18110
CENTRO-OESTE 7905
NORTE 7110
Name: count, dtype: int64

QT_VAGAS_AUTORIZADAS
Mais de 1000 33057
501-1000 13679
201-500 11678
51-100 11534
101-200 9849
Até 50 9695
Name: count, dtype: int64

CARGA_HORARIA
3001-4000h 32375
1001-2000h 27547
2001-3000h 25120
4001-5000h 3729
Mais de 5000h 369
Até 1000h 352
Name: count, dtype: int64

EXTINTO
Não 44746
Sim 44746
Name: count, dtype: int64

PRÉ-PROCESSAMENTO - Codificação das variáveis categóricas

```
# Codificação das variáveis categóricas

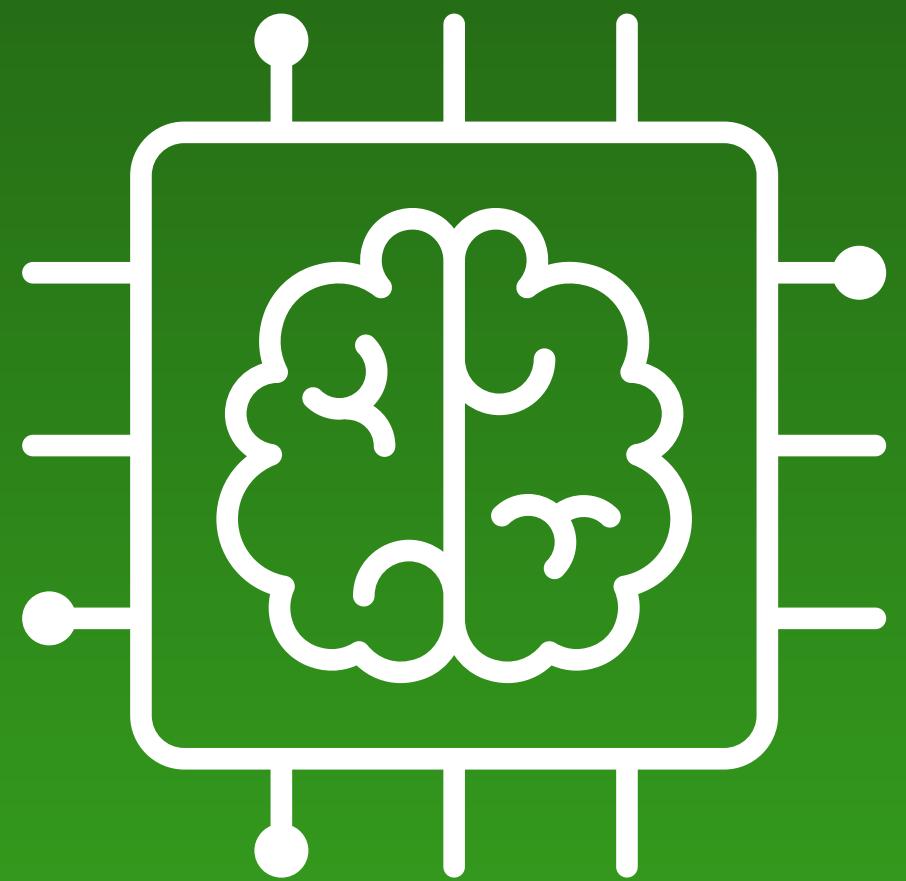
# Definir quais colunas são categóricas nominais e quais são ordinais
colunas_nominais = ['GRAU', 'MODALIDADE', 'CATEGORIA_ADMINISTRATIVA',
                     'ORGANIZACAO_ACADEMICA', 'REGIAO']

colunas_ordinais = ['QT_VAGAS_AUTORIZADAS', 'CARGA_HORARIA']

# Definir as categorias ordenadas para as colunas ordinais
categorias_vagas = ['Até 50', '51-100', '101-200', '201-500', '501-1000', 'Mais de 1000']
categorias_carga = ['Até 1000h', '1001-2000h', '2001-3000h', '3001-4000h', '4001-5000h', 'Mais de 5000h']

# Criar o transformador de colunas
preprocessor = ColumnTransformer(
    transformers=[
        ('onehot', OneHotEncoder(sparse_output=False, drop='first'), colunas_nominais),
        ('ordinal', OrdinalEncoder(categories=[categorias_vagas, categorias_carga]), colunas_ordinais),
    ],
    remainder='passthrough' # manter outras colunas que não foram especificadas
)

# Transformar a variável alvo EXTINTO para numérica
df['EXTINTO'] = df['EXTINTO'].map({'Sim': 1, 'Não': 0})
```



TREINAMENTO

TREINAMENTO

```
# CARREGAMENTO E PREPARAÇÃO DE DADOS

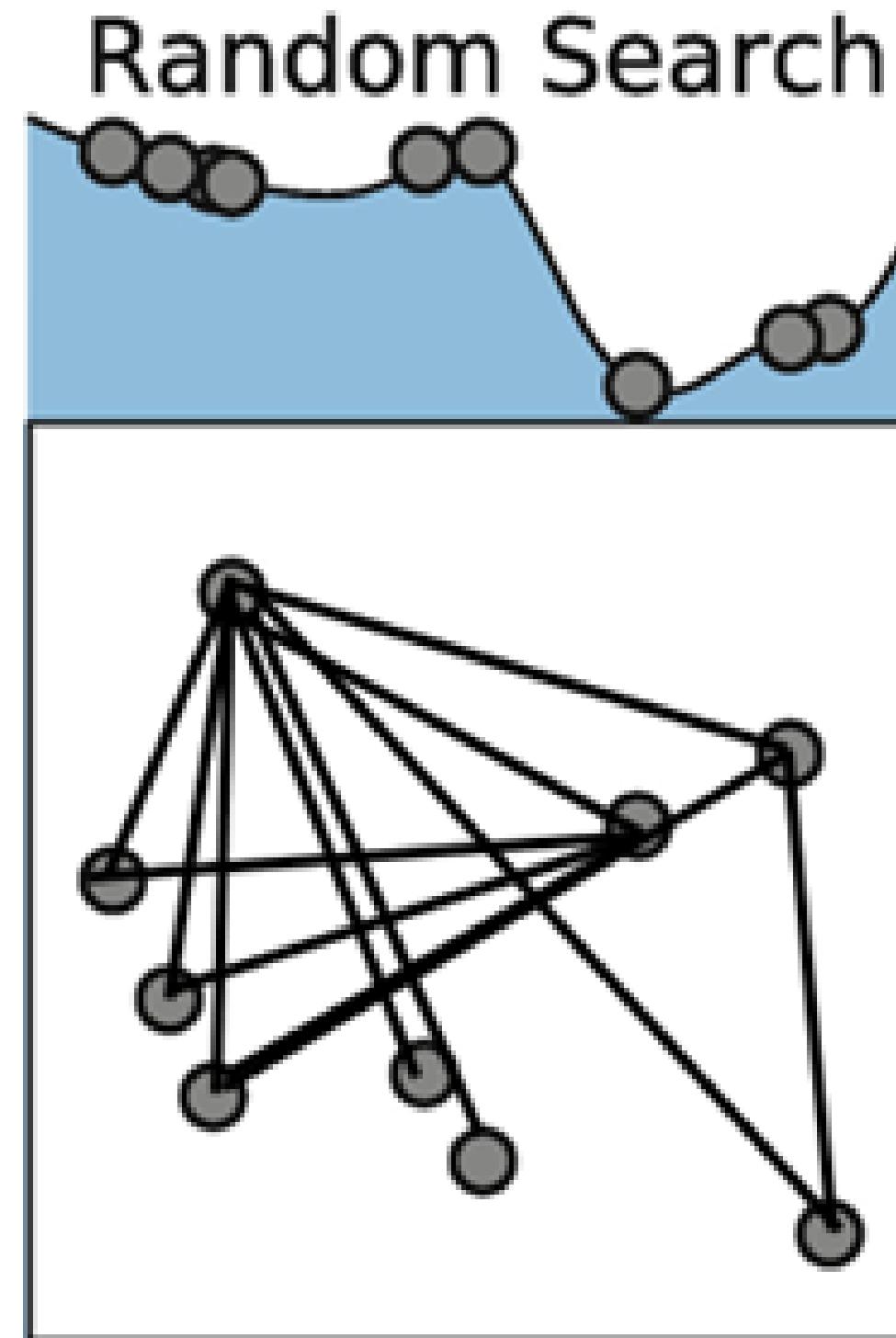
df = pd.read_csv(f"{DATA_DIR}/preprocessed.csv")

print("Informações do dataset:")
df.info()

# Separação features e target
X = df.drop('EXTINTO', axis=1)
y = df['EXTINTO']

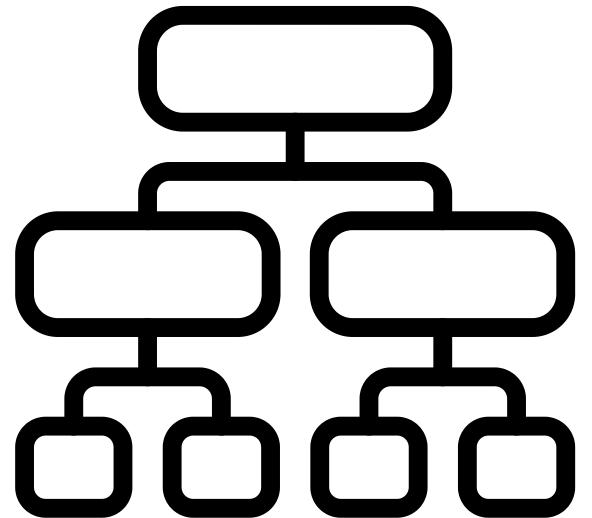
# Divisão conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1, stratify=y
)
```

TREINAMENTO - Busca dos hiperparâmetros

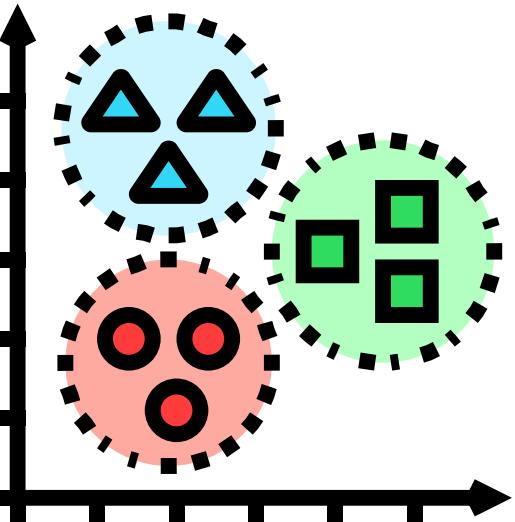


Designed by Author -Shanthababu

TREINAMENTO - Seleção de modelos



Árvore de
Decisão

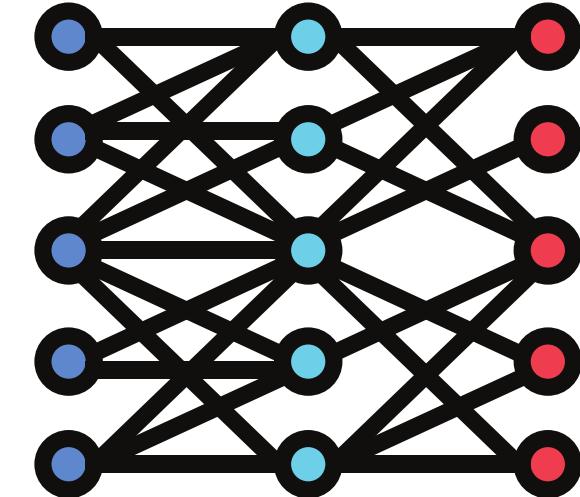


KNN

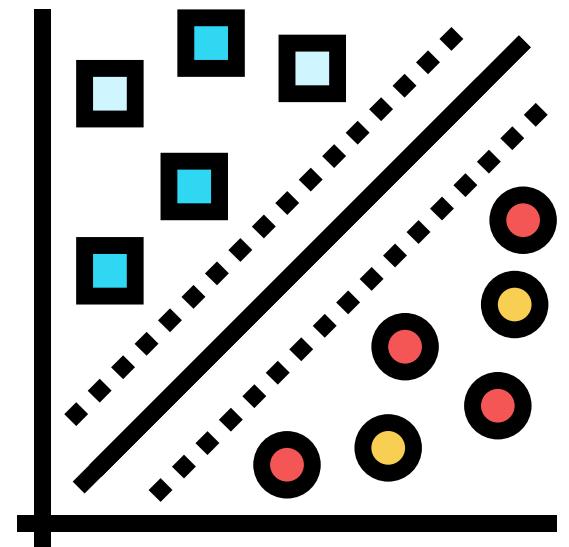
The formula for Naive Bayes classification is displayed:

$$\frac{P(B|A) \times P(A)}{P(B)}$$

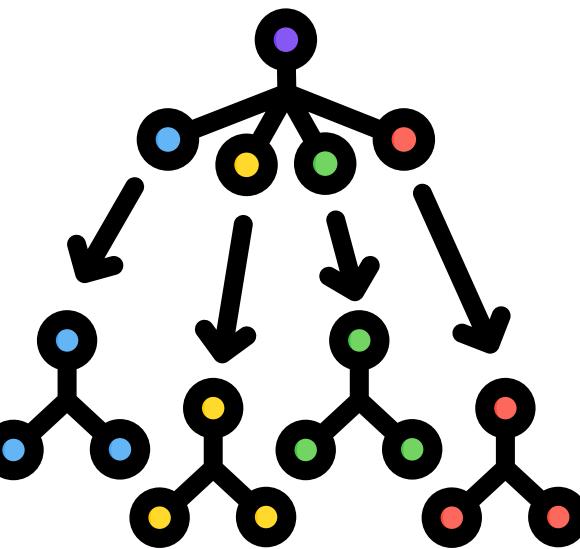
Naive Bayes



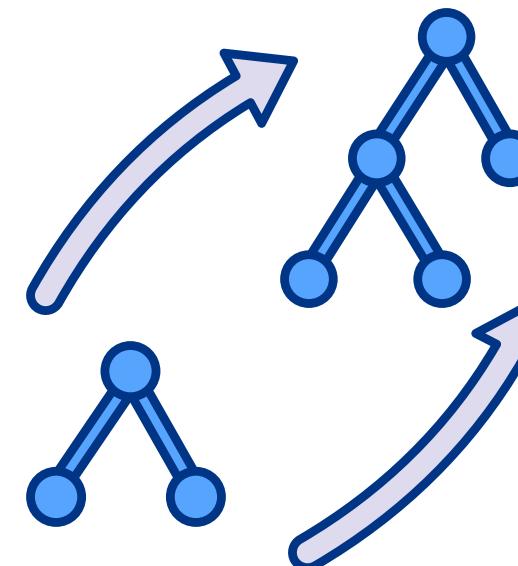
MLP



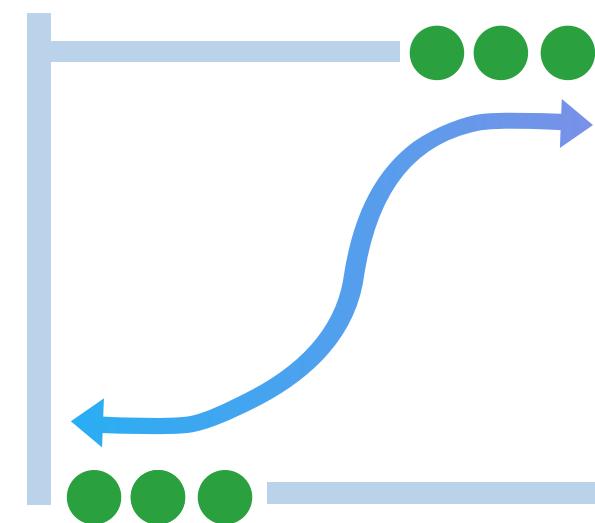
SVM



Random
Forest



XGBoost



Regressão
Logística

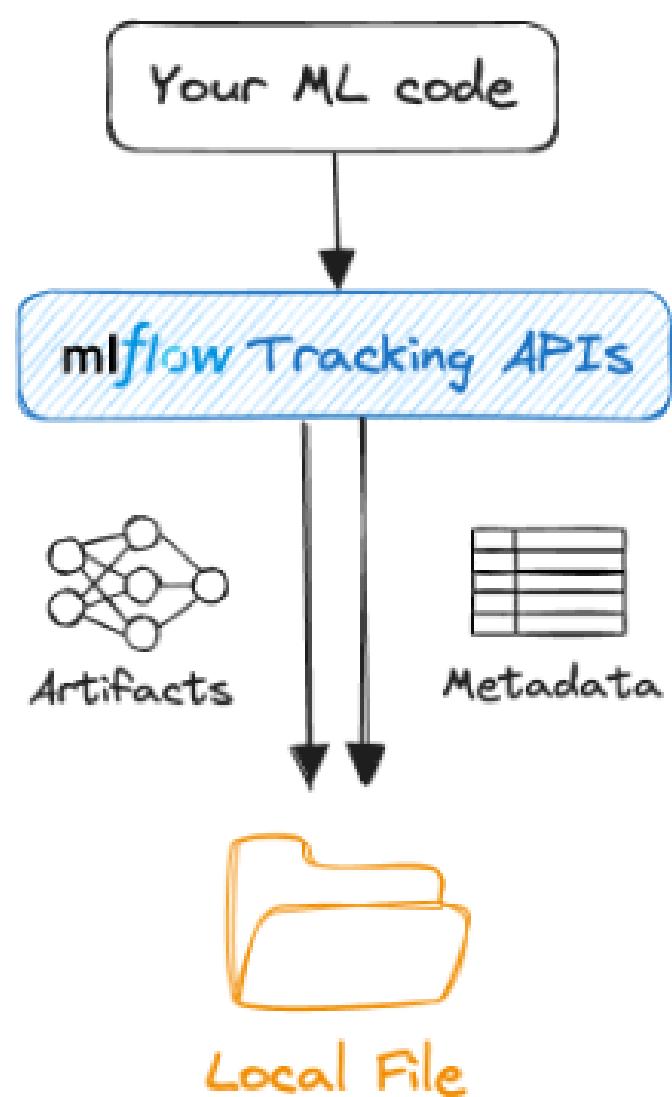
TREINAMENTO - Parâmetros globais de treinamento

```
# Parâmetros globais para treinamento
N_ITER = 10    # Número de iterações para RandomizedSearchCV
CV_FOLDS = 3   # Número de folds para validação cruzada
N_JOBS = 6     # Número de cores para paralelismo
```



RESULTADOS

RESULTADOS - Rastreamento dos experimentos



The screenshot displays the MLflow interface for the experiment named 'extincao_cursos_graduacao'. The 'Runs' tab is selected, showing a list of 27 matching runs. Each run is identified by its name, creation time, duration, source, and model type.

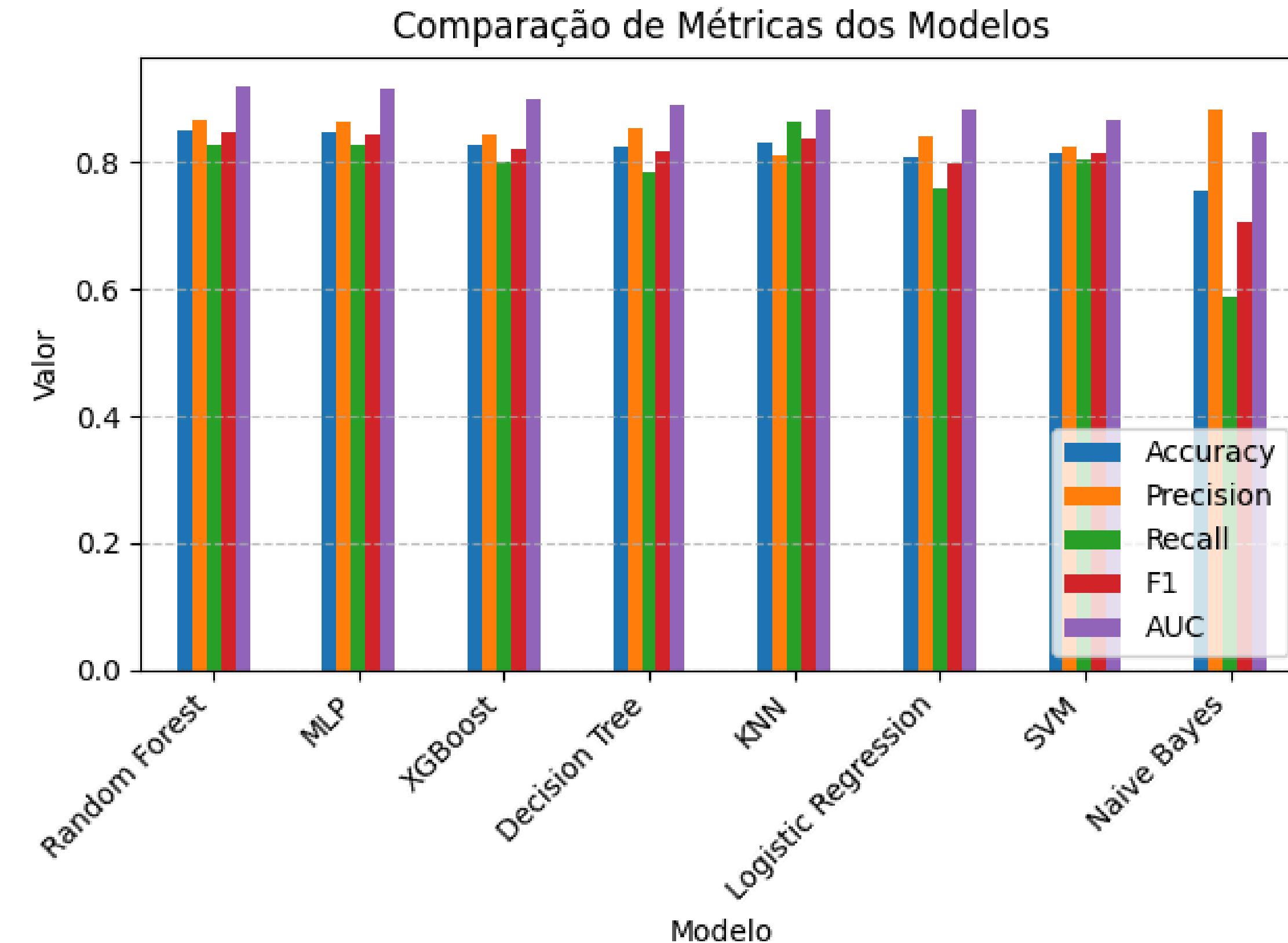
Run Name	Created	Dataset	Duration	Source	Models
XGBoost	25 minutes ago	-	3.3s	ipykern...	modelo_xgboost
Random Forest	25 minutes ago	-	9.4s	ipykern...	modelo_random_forest
SVM	1 hour ago	-	34.1min	ipykern...	modelo_svm
Logistic Regression	1 hour ago	-	3.7s	ipykern...	modelo_logistic_regression
MLP	1 hour ago	-	54.0s	ipykern...	modelo_mlp
Naive Bayes	1 hour ago	-	2.7s	ipykern...	modelo_naive_bayes
KNN	1 hour ago	-	1.1min	ipykern...	modelo_knn
Decision Tree	1 hour ago	-	7.3s	ipykern...	modelo_decision_tree
XGBoost	5 days ago	-	3.9s	ipykern...	modelo_xgboost
KNN	5 days ago	-	1.0min	ipykern...	modelo_knn
SVM	5 days ago	-	1.5min	ipykern...	-

27 matching runs

RESULTADOS - Comparaçāo dos modelos

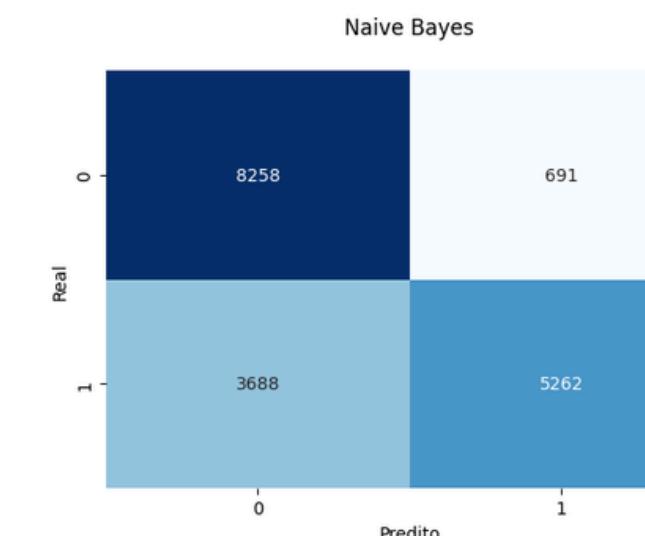
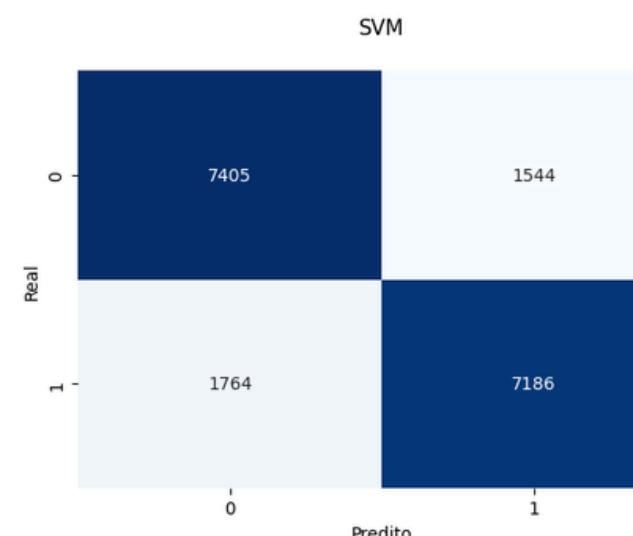
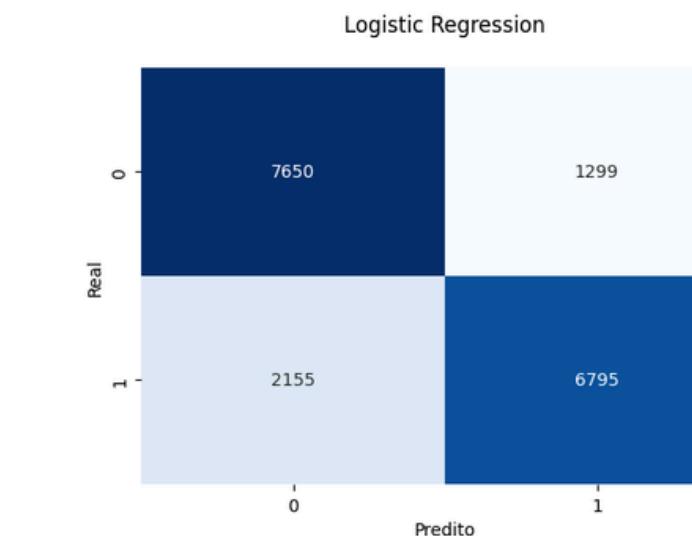
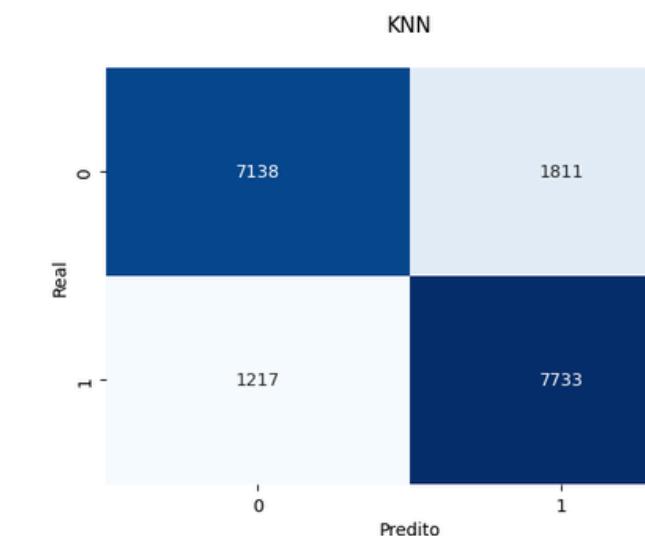
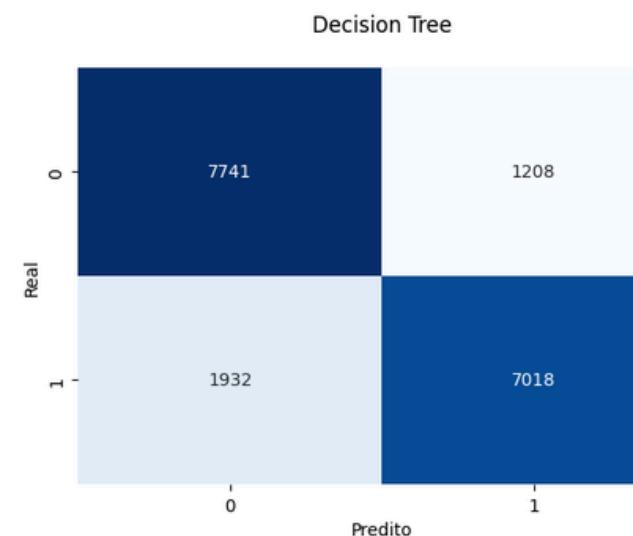
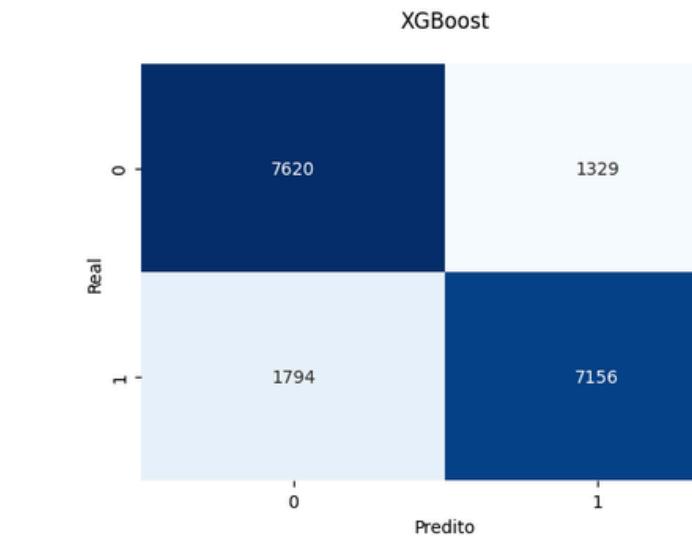
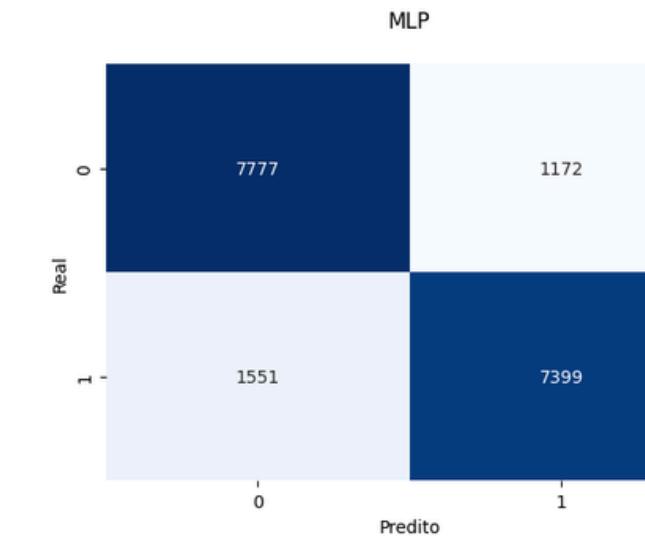
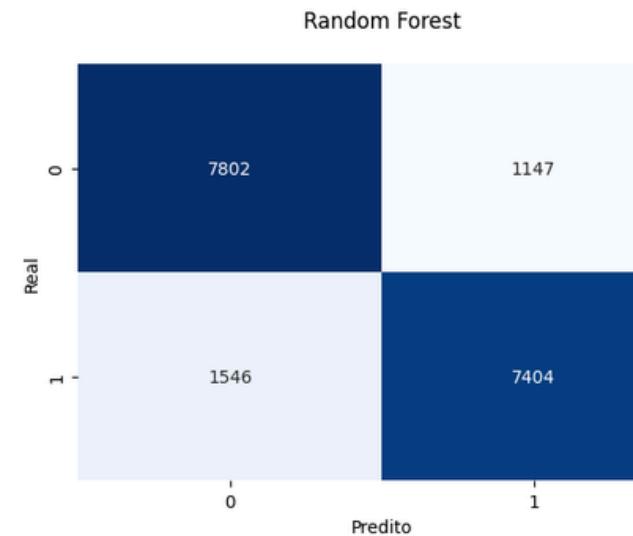
	Accuracy	Precision	Recall	F1	AUC	CV AUC	Time (s)
Random Forest	0.849545	0.865864	0.827263	0.846123	0.919653	0.916918	7.477529
MLP	0.847869	0.863260	0.826704	0.844586	0.916353	0.913495	51.755635
XGBoost	0.825521	0.843371	0.799553	0.820878	0.899356	0.898182	1.369468
Decision Tree	0.824571	0.853149	0.784134	0.817187	0.888861	0.886973	2.692434
KNN	0.830829	0.810247	0.864022	0.836271	0.883408	0.890769	60.744705
Logistic Regression	0.807028	0.839511	0.759218	0.797348	0.881345	0.880294	1.513759
SVM	0.815185	0.823139	0.802905	0.812896	0.865494	0.867680	2041.093085
Naive Bayes	0.755349	0.883924	0.587933	0.706167	0.848013	0.847413	0.456252

RESULTADOS - Comparação das métricas

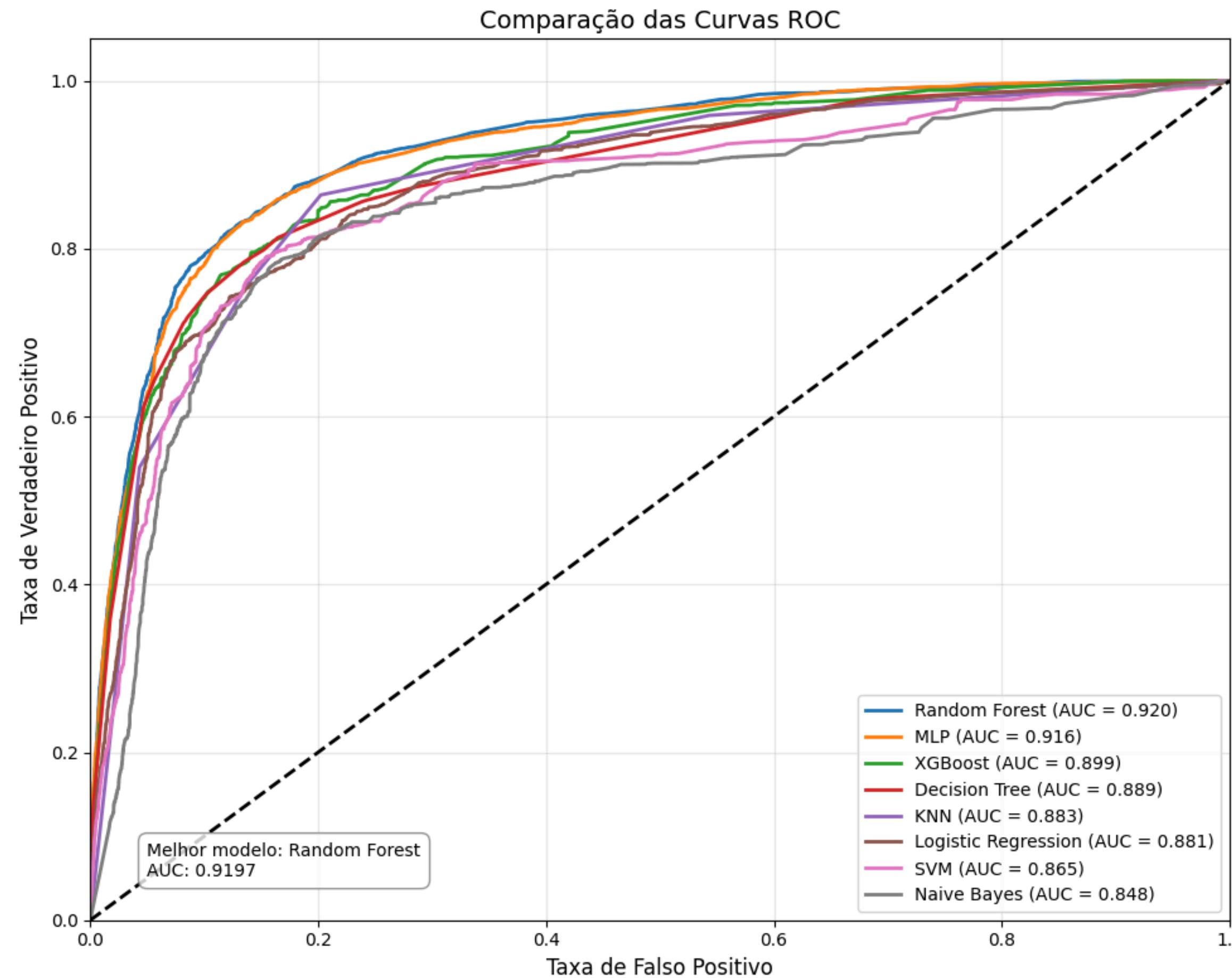


RESULTADOS - Matrizes de confusão

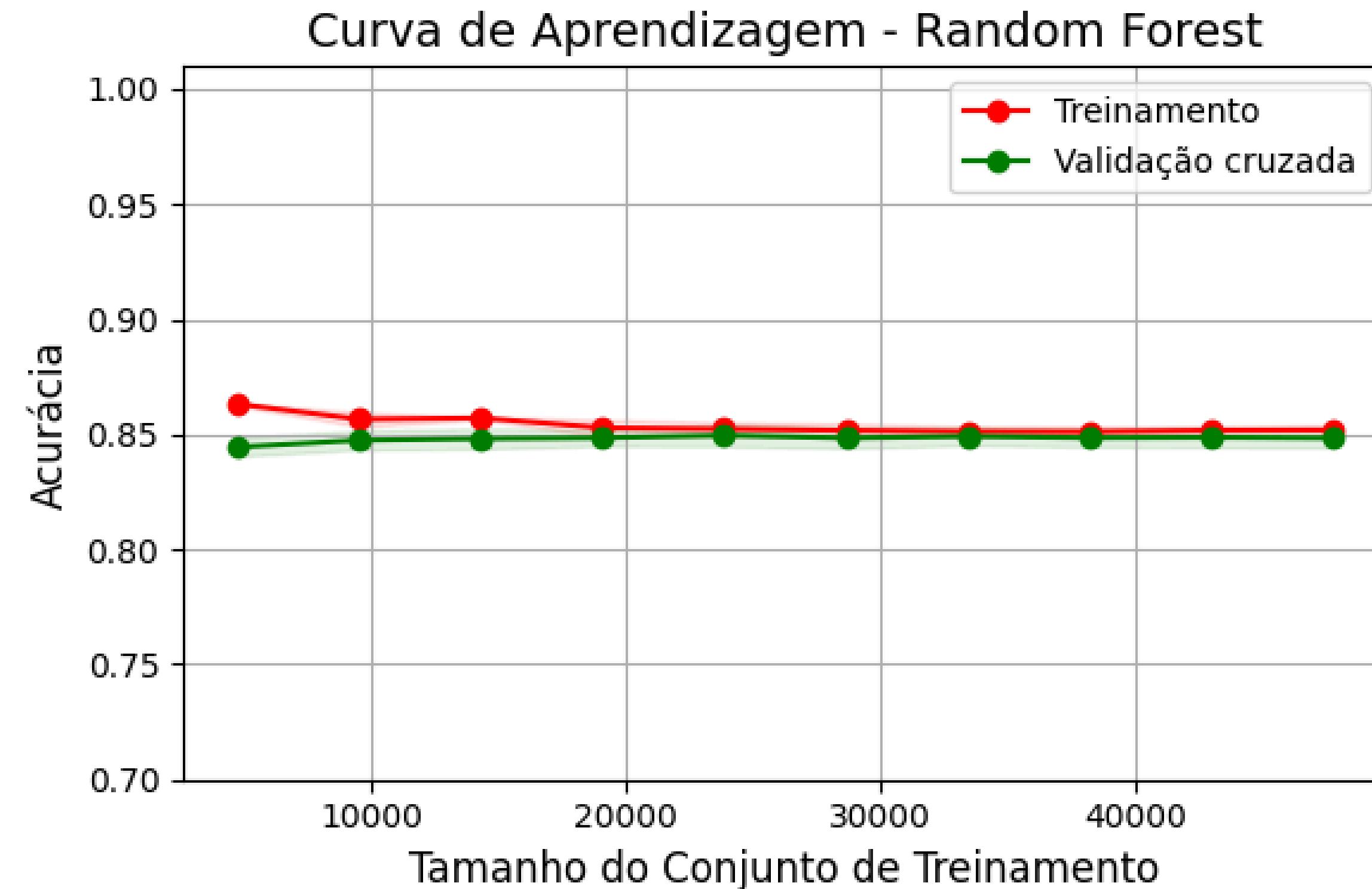
Matrizes de Confusão de Todos os Modelos



RESULTADOS - Curva ROC



RESULTADOS - Curva de aprendizagem



DEPLOY

