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

Francisco Willem Romão Moreira

Natal-RN
2025

RESUMO

Utilizando dados abertos do MEC através de técnicas de ciência de dados e aprendizado de máquina, foi desenvolvido um modelo preditivo que identifica se um determinado curso têm probabilidade de ser extinto. Dos oito algoritmos testados (KNN, Naive Bayes, MLP, Logistic Regression, SVM, Random Forest e XGBoost), o Random Forest apresentou o melhor desempenho com acurácia e F1-score de 85%. Os resultados deste projeto podem auxiliar instituições de ensino superior e órgãos governamentais na tomada de decisões estratégicas sobre a oferta de cursos.

Palavras-chave – aprendizado de máquina, classificação, extinção de cursos, random Forest, dados educacionais

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A adoção de técnicas de aprendizado de máquina na análise de dados educacionais tem se mostrado uma abordagem promissora para subsidiar a tomada de decisões estratégicas em instituições de ensino superior. Diante de um cenário em que diversos cursos de graduação enfrentam risco de descontinuidade, seja por baixa demanda, desempenho insuficiente ou reestruturações institucionais, torna-se relevante o desenvolvimento de sistemas preditivos capazes de identificar, com antecedência, padrões associados à extinção de cursos. Este trabalho propõe a construção de um modelo de classificação, utilizando dados abertos do Ministério da Educação (MEC, 2022), com o objetivo de prever o status de cursos superiores no Brasil — categorizando-os como "extinto" ou "não extinto".

O conjunto de dados utilizado é composto por mais de 900 mil registros e 18 atributos relacionados às características dos cursos, instituições e regiões. Para transformar esses dados brutos em insumos úteis para os modelos de aprendizado supervisionado, foi realizado um processo sistemático de análise exploratória, limpeza, codificação de variáveis categóricas, normalização de atributos numéricos e balanceamento das classes. A modelagem preditiva foi conduzida com a avaliação de oito algoritmos de classificação: K-Nearest Neighbors (KNN), Naive Bayes, Regressão Logística, Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Decision Tree, Random Forest e XGBoost.

A validação dos modelos foi realizada com uso de validação cruzada e otimização de hiperparâmetros por Random Search, priorizando métricas como acurácia, F1-score e AUC-ROC para aferir o desempenho preditivo. Entre os modelos testados, o Random Forest obteve o melhor resultado, com acurácia e F1-score próximos de 85%. Para facilitar o uso prático do modelo, foi desenvolvida uma API em FastAPI (Tiangolo, Sebastián Ramírez, 2025) e uma interface em Streamlit (Streamlit Inc., 2025) , permitindo a realização de inferências de forma interativa.

Este trabalho insere-se no campo emergente da Learning Analytics e demonstra o potencial da inteligência artificial aplicada à educação como ferramenta para diagnóstico institucional, otimização de recursos e políticas públicas voltadas ao ensino superior (SCHEFFEL et al., 2022).

2 METODOLOGIA

Este projeto foi desenvolvido com base em um fluxo estruturado de ciência de dados, dividido em quatro etapas principais: (1) análise exploratória e limpeza dos dados; (2) pré-processamento; (3) modelagem e avaliação; e (4) implantação do modelo. As implementações foram realizadas em *Python*, utilizando bibliotecas como *pandas* (pandas development team, 2025), *scikit-learn* (scikit-learn developers, 2025), *XGBoost* (XGBoost contributors, 2025), *FastAPI* (Tiangolo, Sebastián Ramírez, 2025), *Streamlit* (Streamlit Inc., 2025), *MLflow* (Databricks, 2025) para rastreamento dos experimentos e *DVC* (Iterative, 2025) para versionamento de dados.

2.1 Controle e Versionamento de Dados

Todo o fluxo de dados foi versionado com a ferramenta *DVC* (Data Version Control), que permitiu rastrear diferentes estágios do pipeline — incluindo os dados brutos, os dados após limpeza e os dados prontos para treinamento. Os arquivos em cada estágio foram armazenados e sincronizados com um bucket privado na nuvem *Amazon S3*, garantindo reprodutibilidade e controle de versões ao longo do desenvolvimento.

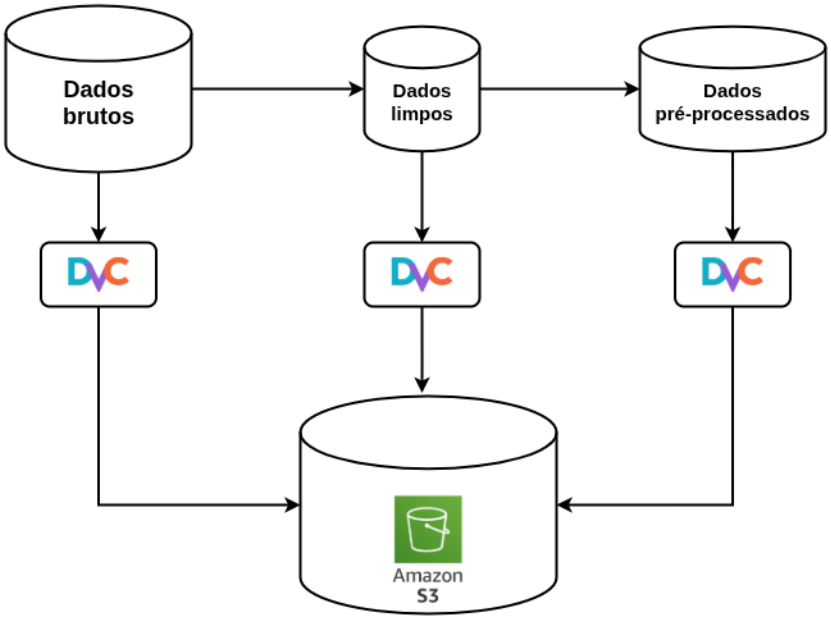


Figura 1 – Fluxograma do versionamento dos dados

2.2 Análise Exploratória e Limpeza

Os dados brutos foram carregados a partir de um arquivo CSV contendo 902.676 registros, abrangendo 18 variáveis relacionadas a cursos de graduação no Brasil [Figura 2].

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 902676 entries, 0 to 902675
Data columns (total 18 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   CODIGO_IES                902676 non-null  int64
 1   NOME_IES                  902676 non-null  object
 2   CATEGORIA_ADMINISTRATIVA  902676 non-null  object
 3   ORGANIZACAO_ACADEMICA     902676 non-null  object
 4   CODIGO_CURSO              902676 non-null  int64
 5   NOME_CURSO                902676 non-null  object
 6   GRAU                      902676 non-null  object
 7   AREA_OCDE                 680577 non-null  object
 8   MODALIDADE                902676 non-null  object
 9   SITUACAO_CURSO            902676 non-null  object
 10  QT_VAGAS_AUTORIZADAS      902676 non-null  int64
 11  CARGA_HORARIA             902676 non-null  int64
 12  CODIGO_AREA_OCDE_CINE     902649 non-null  object
 13  AREA_OCDE_CINE            902649 non-null  object
 14  CODIGO_MUNICIPIO          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
```

Figura 2 – Informações gerais do dataset

Inicialmente, foram inspecionados os tipos de dados de cada coluna, bem como a presença de valores ausentes [Figura 3].

```
In [ ]: # Total de valores ausentes por coluna

        print(df.isna().sum())
```

```
CODIGO_IES                    0
NOME_IES                      0
CATEGORIA_ADMINISTRATIVA      0
ORGANIZACAO_ACADEMICA         0
CODIGO_CURSO                  0
NOME_CURSO                    0
GRAU                          0
AREA_OCDE                222099
MODALIDADE                    0
SITUACAO_CURSO                0
QT_VAGAS_AUTORIZADAS          0
CARGA_HORARIA                 0
CODIGO_AREA_OCDE_CINE        27
AREA_OCDE_CINE               27
CODIGO_MUNICIPIO              0
MUNICIPIO                     0
UF                            0
REGIAO                        0
dtype: int64
```

Figura 3 – Total de valores ausentes por coluna

As colunas que apresentavam valores ausentes e que, adicionalmente, não eram relevantes para a análise foram removidas.

Em seguida, procedeu-se à identificação e remoção de registros duplicados. No total, foram encontrados e excluídos 30 registros repetidos [Figura 4].

```
In [ ]: # Total de linhas duplicadas

        print(df.duplicated().sum())
```

```
30
```

Figura 4 – Total de registros duplicados

Posteriormente, foi realizada a detecção e o tratamento de outliers utilizando a técnica do intervalo interquartil (IQR). Os resultados podem ser observados nos boxplots a seguir [Figura 5].

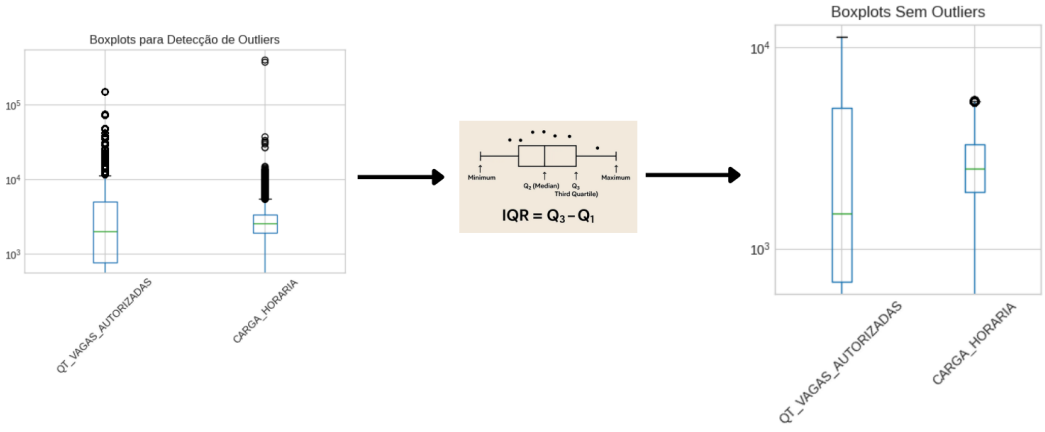


Figura 5 – Boxplots das variáveis Quantidade de Vagas Autorizadas e Carga Horária

A etapa final da limpeza consistiu na remoção de outras variáveis consideradas irrelevantes para os objetivos do projeto [Figura 6].

```
# Remover colunas não necessárias

columns_to_remove = ['CODIGO_IES', 'NOME_IES', 'CODIGO_CURSO', 'CODIGO_MUNICIPIO']

df = df.drop(columns=columns_to_remove)
```

```
df.columns
```

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

Figura 6 – Remoção de variáveis não necessárias

A seguir, algumas visualizações exploratórias que ajudam a compreender melhor a distribuição dos dados e suas principais características:

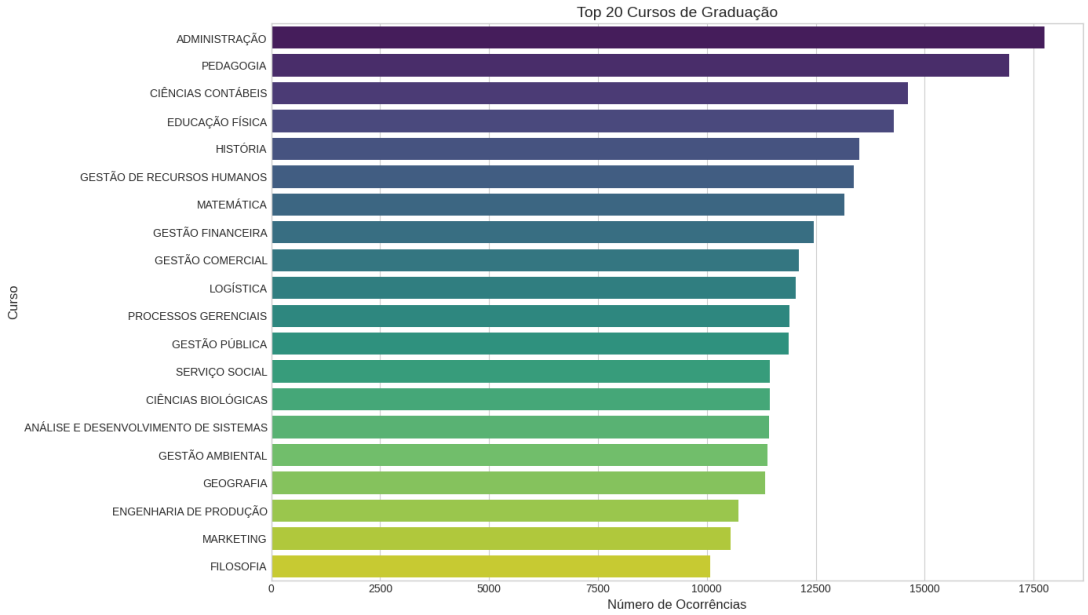


Figura 7 – Gráfico de barras Top 20 Cursos de Graduação

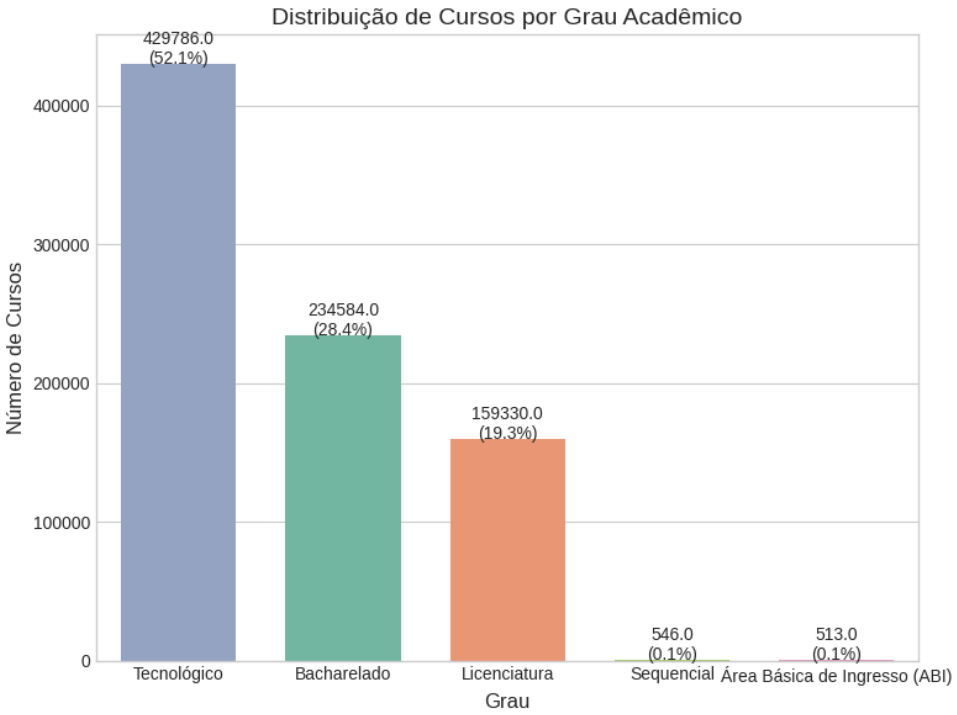


Figura 8 – Gráfico de barras Grau Acadêmico

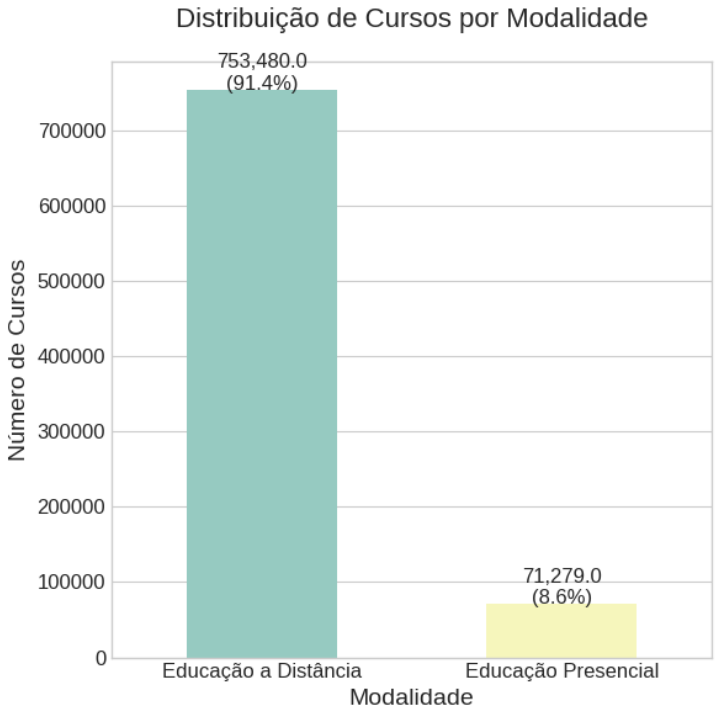


Figura 9 – Gráfico de barras Modalidade

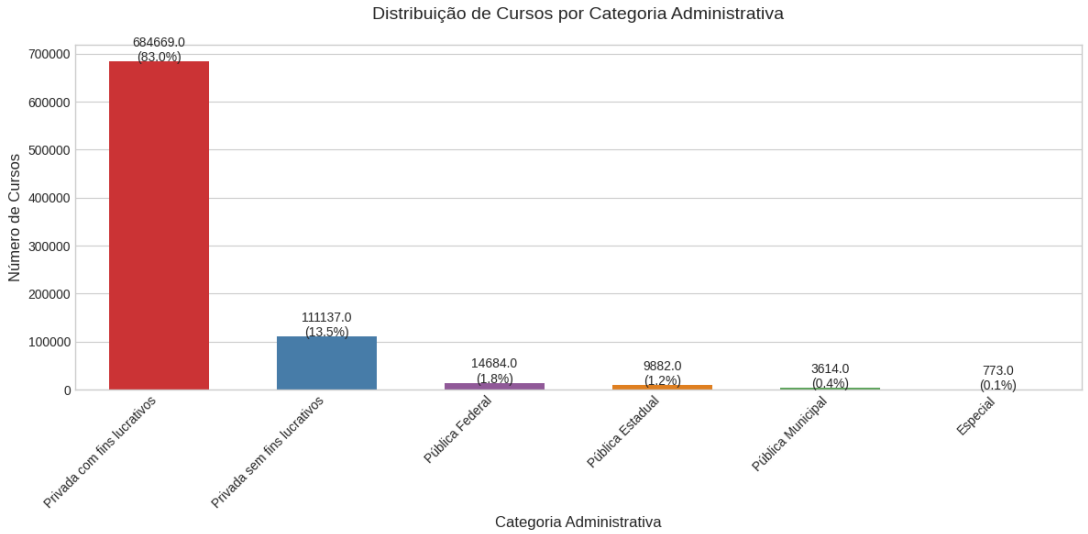


Figura 10 – Gráfico de barras Categoria Administrativa

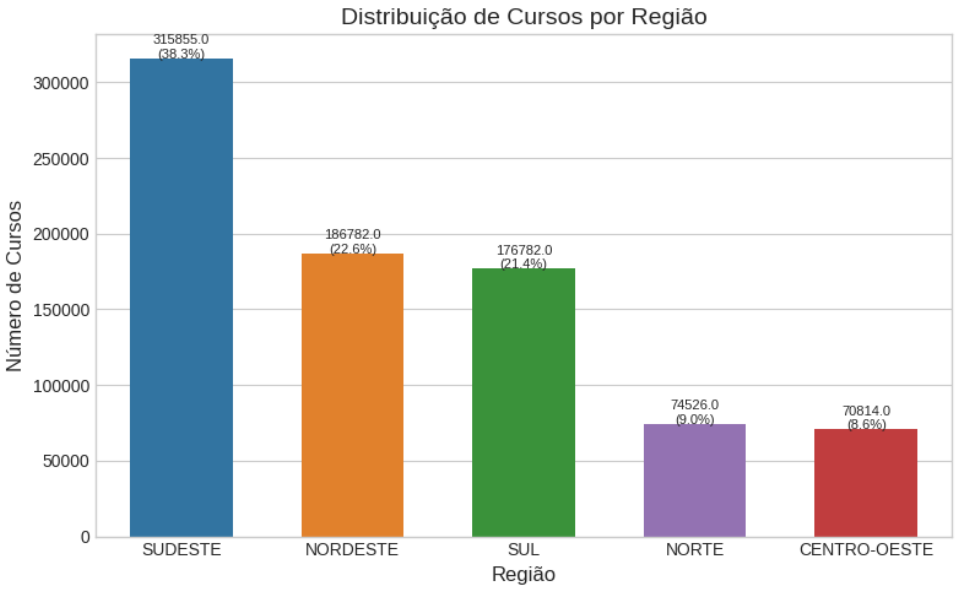


Figura 11 – Gráfico de barras Região

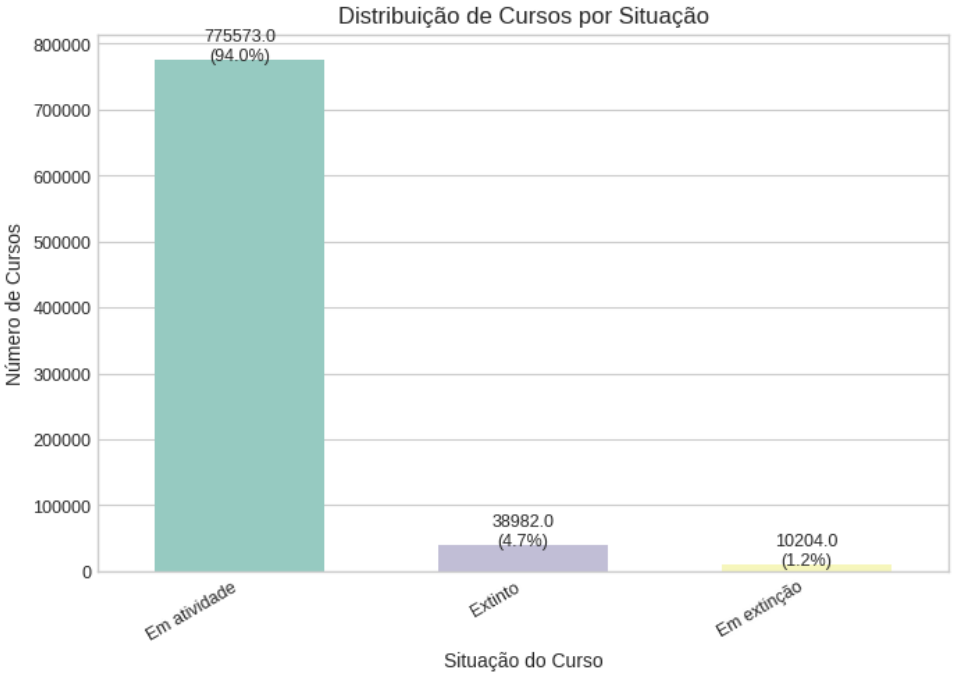


Figura 12 – Gráfico de barras Situação Curso

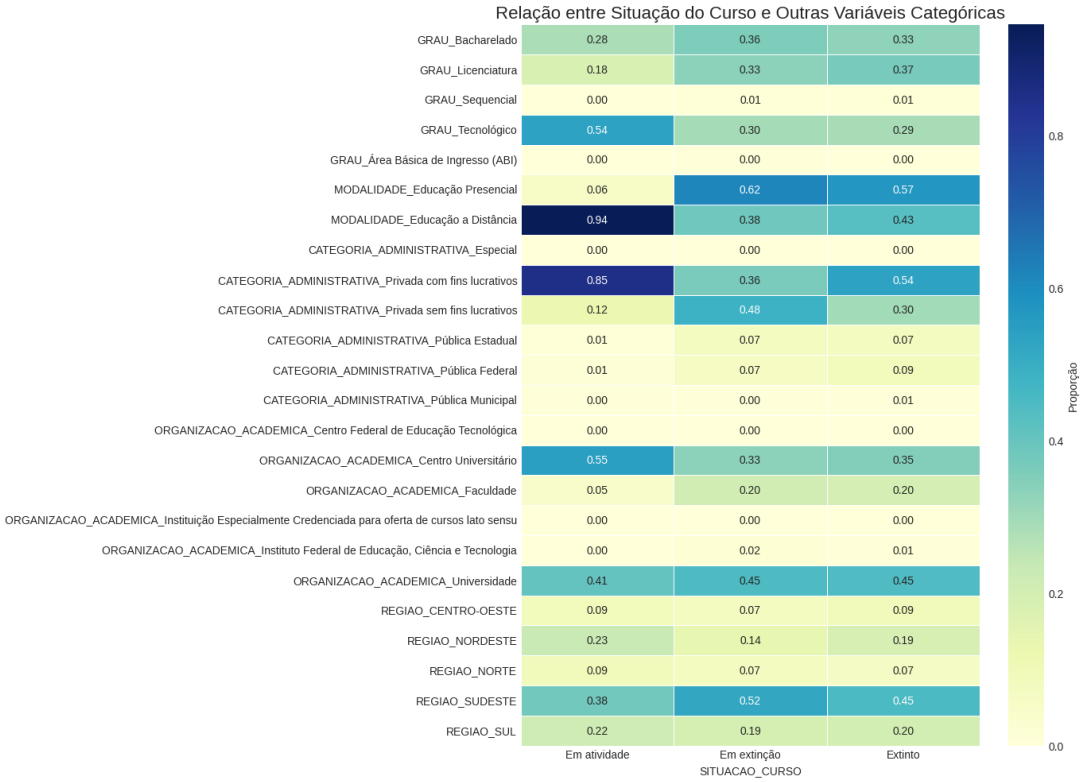


Figura 13 – Gráfico Heatmap Situação Curso em relação a outras variáveis categóricas

2.3 Pré-processamento dos Dados

O processo de pré-processamento envolveu diversas etapas: transformação da variável alvo (*SITUAÇÃO CURSO*), discretização de variáveis numéricas, balanceamento das classes, além da codificação de variáveis categóricas nominais e ordinais.

A transformação da variável alvo consistiu em converter o problema original de classificação multiclasse em um problema binário. As categorias "Em Atividade", "Em Extinção"e "Extinto"foram reorganizadas em dois grupos, conforme mostrado na [Figura 14]. Essa simplificação permitiu o uso de algoritmos de classificação binária, facilitando a modelagem do problema.

```
# 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']
```

Figura 14 – Processo de transformação da variável alvo

Na etapa seguinte, foram discretizadas as variáveis numéricas *Carga Horária* e *Quantidade de Vagas Autorizadas*, como demonstrado na [Figura 15]. Essa transformação visou facilitar a interpretação dos dados e permitir o uso de técnicas baseadas em categorias.

```
# 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
)
```

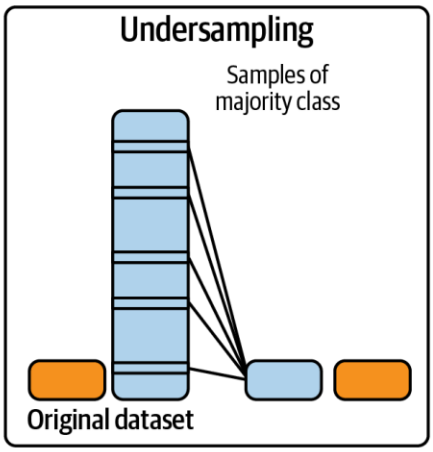
```
# Discretização de QT_VAGAS_AUTORIZADAS

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
)
```

Figura 15 – Discretização das variáveis numéricas

Para lidar com o desbalanceamento das classes, foi aplicada a técnica de *undersampling*, resultando na distribuição apresentada na [Figura 16].




```
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})
```

Figura 16 – Balanceamento das classes

Posteriormente, as colunas *NOME_ CURSO*, *UF* e *MUNICÍPIO* foram removidas por não agregarem valor à modelagem e por representarem atributos altamente específicos e potencialmente redundantes.

A [Figura 17] apresenta uma visão geral do conjunto de dados após as etapas descritas.

```
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
----------------------------------------
```

Figura 17 – Visão geral do dataset

Por fim, realizou-se a codificação das variáveis categóricas: foi aplicado *One-Hot Encoding* para variáveis nominais e *Ordinal Encoding* para variáveis com hierarquia definida. O resultado é ilustrado na [Figura 18].

```python
# 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
)

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

Figura 18 – Codificação das variáveis categóricas

2.4 Modelagem e Avaliação

A etapa de modelagem iniciou-se com o carregamento dos dados previamente pré-processados. Em seguida, o conjunto de dados foi particionado em treino e teste, reservando-se 20% dos dados para o conjunto de teste. Utilizou-se o parâmetro `stratify` para garantir que a distribuição das classes da variável alvo fosse preservada em ambas as divisões [Figura 19].

```python
# 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
)
```

Figura 19 – Carregamento e preparação dos dados

Para a busca de hiperparâmetros, optou-se pela técnica de *Randomized Search*, por sua capacidade de explorar eficientemente uma amostra representativa do espaço de busca, reduzindo o tempo computacional em comparação ao *Grid Search*.

Os modelos selecionados para o treinamento estão listados na [Figura 20]. A escolha contemplou algoritmos com diferentes abordagens e complexidades, com o objetivo de comparar seu desempenho no contexto do problema tratado.

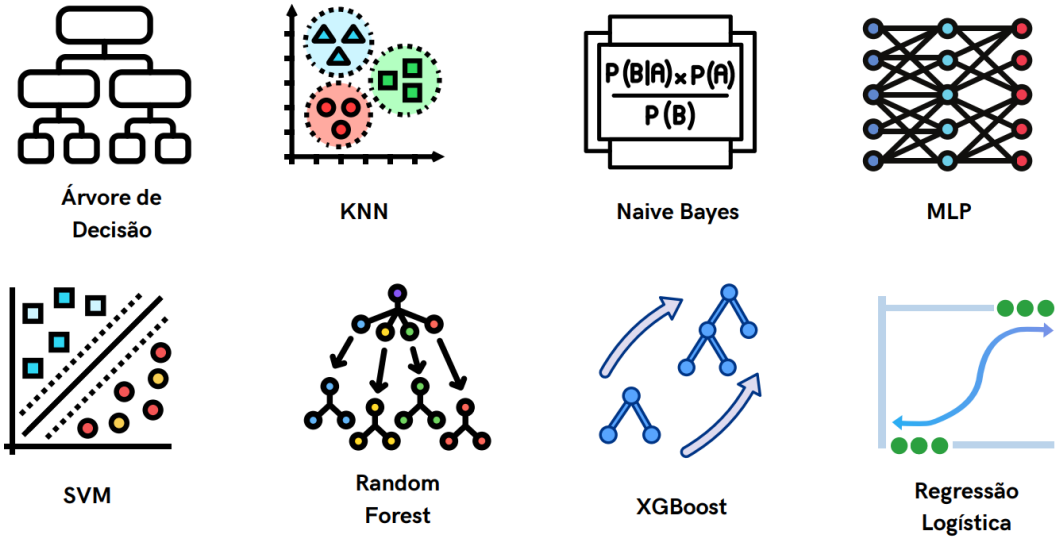


Figura 20 – Modelos escolhidos

Antes de iniciar o processo de treinamento, foram definidos os parâmetros globais, tais como o número de iterações do *RandomizedSearchCV*, o número de divisões para validação cruzada (*folds*) e o número de núcleos de CPU utilizados para paralelismo. Esses parâmetros estão detalhados na [Figura 21].

```
# 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
```

Figura 21 – Parâmetros globais de treinamento

As métricas utilizadas para avaliação e comparação dos modelos foram: acurácia, precisão, sensibilidade (recall), F1-score, AUC, AUC com validação cruzada (CV AUC) e tempo de execução.

Para garantir rastreabilidade, reprodutibilidade e organização dos experimentos realizados durante o processo de modelagem, utilizou-se o framework *MLflow*. Essa ferramenta possibilitou o registro automático de métricas, parâmetros e artefatos de cada execução [Figura 22].

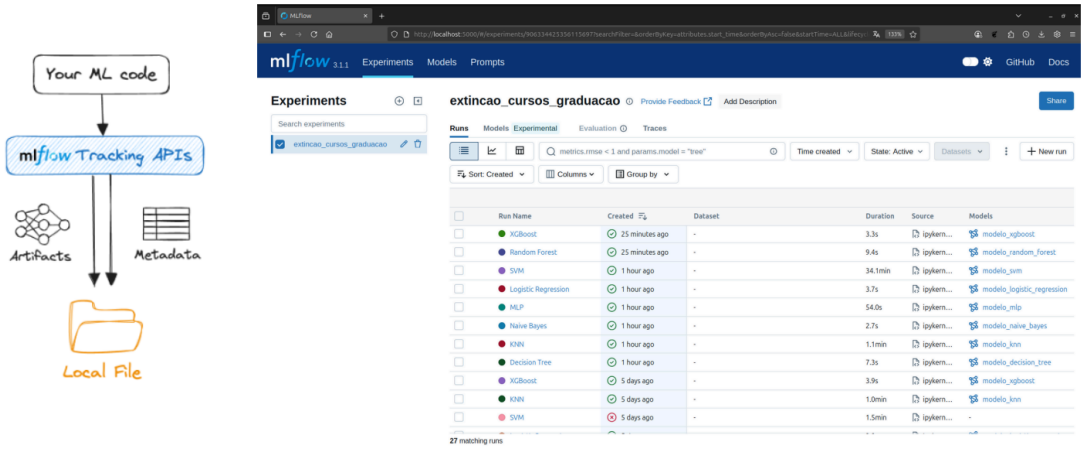


Figura 22 – Rastreamento dos experimentos

2.5 Implantação

A implantação foi realizada por meio de uma API desenvolvida com *FastAPI*, expondo o modelo para inferência via requisições HTTP. Além disso, foi construído um *frontend* interativo com *Streamlit*, permitindo a entrada de dados por usuários e a exibição dos resultados de forma visual e acessível. Todo o ambiente foi containerizado utilizando *Docker*, o que garante portabilidade, facilidade de *deploy* e consistência na execução entre diferentes ambientes [Figura 23]. O código-fonte completo do projeto está disponível no repositório público: <https://github.com/willemromao/cursos_graduacao_brasil>, onde também se encontram, no arquivo `README`, as instruções detalhadas para execução local da aplicação.

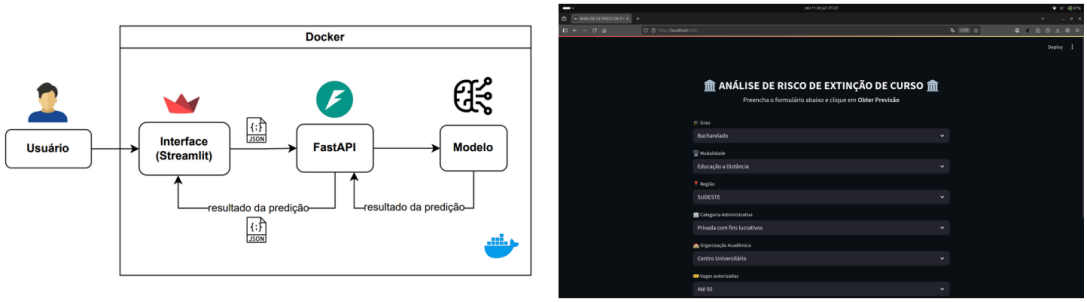


Figura 23 – Fluxograma e tela interativa do modelo

3 RESULTADOS E DISCUSSÕES

3.1 Resultados

A [Figura 24] apresenta uma tabela comparativa contendo os resultados obtidos para cada modelo.

	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

Figura 24 – Tabela de comparação dos modelos

A tabela está ordenada pela desempenho, indicando que o modelo *Random Forest* apresentou o melhor resultado geral. O *MLP* (Perceptron Multicamadas) apresentou desempenho próximo, mas, por ser consideravelmente mais complexo e demandar maior tempo de treinamento, não foi considerado a melhor escolha para implantação.

O modelo *SVM* mostrou-se computacionalmente custoso, exigindo mais de 30 minutos para treinamento, o que compromete sua viabilidade, especialmente em cenários de produção ou retraining.

Por outro lado, o *Naive Bayes* apresentou o pior desempenho entre os modelos avaliados.

A comparação visual entre os modelos pode ser observada no gráfico de barras da [Figura 25].

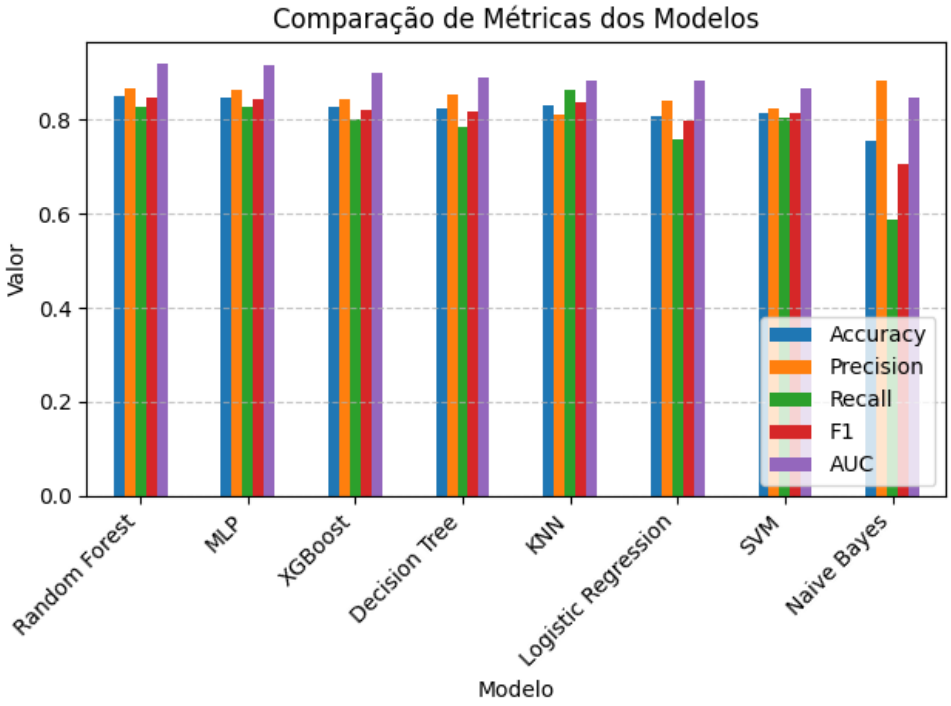


Figura 25 – Gráfico de barras comparativo dos modelos

A performance dos modelos também foi analisada por meio das matrizes de confusão, permitindo a identificação de padrões de erro específicos na classificação de cada classe [Figura 26].

Figura 26 – Matrizes de confusão de todos os modelos

As curvas ROC apresentadas na [Figura 27] fornecem uma visão mais detalhada da sensibilidade e especificidade dos modelos, permitindo a comparação da capacidade de discriminação entre as classes.

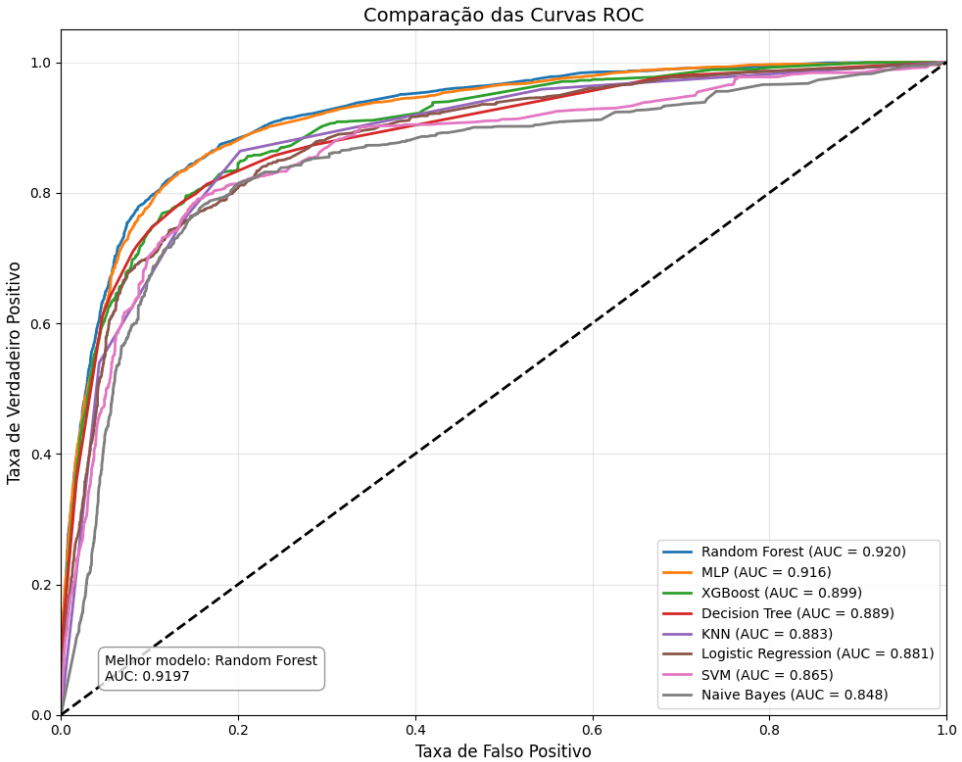


Figura 27 – Curvas ROC comparativo

Por fim, foi analisada a curva de aprendizagem para o modelo *Random Forest*, com o objetivo de avaliar seu comportamento durante o treinamento e a validação. A curva permite verificar a presença de overfitting ou underfitting e entender a dinâmica de generalização do modelo [Figura 28].

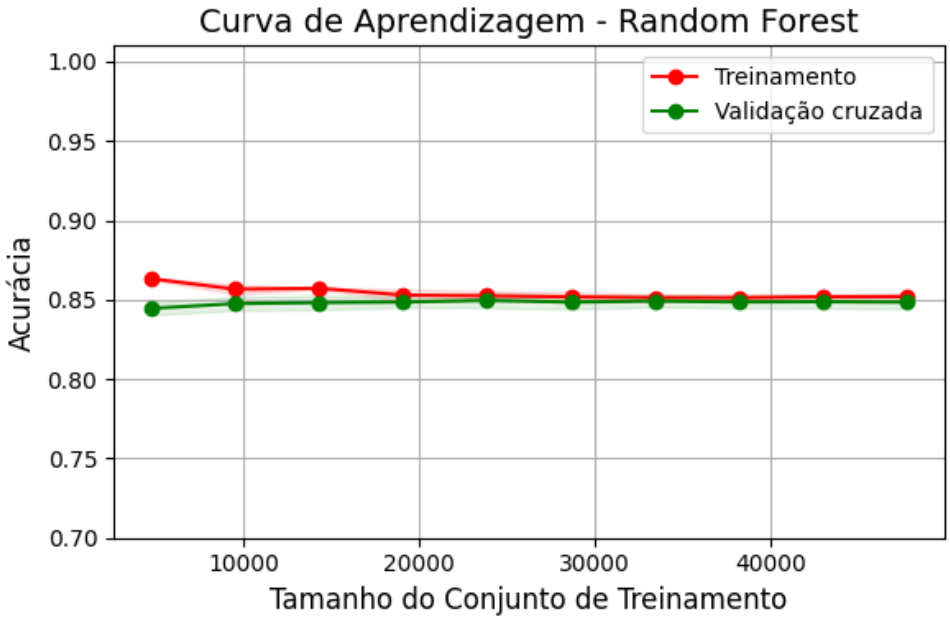


Figura 28 – Curva de aprendizagem para o modelo com Random Forest

3.2 Discussões

Embora o modelo Random Forest tenha apresentado alto desempenho preditivo, seu uso em contextos institucionais exige atenção quanto à interpretabilidade, sendo recomendável o uso de ferramentas explicativas como SHAP. Modelos como Decision Tree, embora menos eficazes em termos absolutos, podem ser mais adequados quando a transparência das decisões for um requisito. Além disso, a ausência de variáveis que identifiquem claramente as áreas do conhecimento dos cursos limita a capacidade explicativa do modelo e pode ocultar padrões relevantes. A extinção de cursos pode também refletir desigualdades regionais e institucionais historicamente enraizadas, o que levanta a necessidade de cuidado na interpretação dos resultados para não reforçar vieses sistêmicos. Por fim, o custo computacional elevado em alguns modelos — como SVM — e a necessidade de retreinamento periódico reforçam a importância de considerar viabilidade operacional no uso real do sistema.

4 CONCLUSÃO

Este trabalho demonstrou o potencial do uso de aprendizado de máquina para prever a extinção de cursos de graduação com base em dados públicos do Ministério da Educação. O processo envolveu etapas de preparação dos dados, avaliação de múltiplos algoritmos e implementação de um sistema interativo acessível a usuários finais. O modelo Random Forest se destacou pelo desempenho consistente, apontando que há padrões informativos relevantes para a tarefa. A aplicação prática da solução, por meio de uma API e interface gráfica, oferece uma ferramenta promissora para gestores e tomadores de decisão. Como próximos passos, destaca-se a necessidade de incorporar atributos mais

informativos, ampliar a busca por hiperparâmetros e validar a solução em contextos reais, com foco em apoio à gestão educacional estratégica.

REFERÊNCIAS

Databricks. *MLflow Documentation*. 2025. <https://mlflow.org/docs/latest/index.html>. Acesso em: Jul. 2025.

Iterative. *DVC Documentation*. 2025. <https://dvc.org/doc>. Acesso em: Jul. 2025.

MEC. *Sistema e-MEC – Cursos de Graduação do Brasil – Dados Abertos*. 2022. <https://dadosabertos.mec.gov.br/indicadores-sobre-ensino-superior/item/183-cursos-de-graduacao-do-brasil>. Acesso em: Jul. 2025.

pandas development team. *Pandas Documentation*. 2025. <https://pandas.pydata.org/docs/>. Acesso em: Jul. 2025.

SCHEFFEL, M. et al. Chapter 23: Learning analytics policies. In: *Handbook of Learning Analytics*. [S.l.: s.n.], 2022. Discusses the policy frameworks and adoption of Learning Analytics in higher education.

scikit-learn developers. *Scikit-learn Documentation*. 2025. <https://scikit-learn.org/stable/documentation.html>. Acesso em: Jul. 2025.

Streamlit Inc. *Streamlit Documentation*. 2025. <https://docs.streamlit.io/>. Acesso em: Jul. 2025.

Tiangolo, Sebastián Ramírez. *FastAPI Documentation*. 2025. <https://fastapi.tiangolo.com/>. Acesso em: Jul. 2025.

XGBoost contributors. *XGBoost Documentation*. 2025. <https://xgboost.readthedocs.io/>. Acesso em: Jul. 2025.