



Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital



RELATÓRIO DO PROJETO 2: Aplicação de Aprendizado de Máquina na Classificação de Extinção de Cursos de Graduação

Francisco Willem Romão Moreira
20220036966

Natal-RN
2024

RESUMO

Este relatório apresenta todas as etapas da construção de modelos de classificação para prever a extinção de cursos de graduação no Brasil. O relatório abrange desde o pré-processamento e a escolha dos algoritmos até os resultados obtidos dos modelos. Dos três algoritmos de classificação testados (Árvore de Decisão, Florestas Aleatórias e Perceptron de Múltiplas Camadas), o que apresentou o melhor desempenho foi o algoritmo de Florestas Aleatórias, com uma acurácia de 81%.

Palavras-chave: aprendizado de máquina, ciência de dados, pré-processamento, classificação, cursos de graduação brasil, risco de extinção.

LISTA DE FIGURAS

1	Atributos selecionados.	6
2	Atributos descartados.	6
3	Correção de dados inválidos.	6
4	Tratamento de dados ausentes.	7
5	Box plot da Quantidade de Vagas Autorizadas.	7
6	Box plot de Carga Horária.	8
7	Box plot da Carga Horária com IQR.	8
8	Box plot de Carga Horária com IQR.	9
9	Classes desbalanceadas.	9
10	Classes balanceadas.	10
11	Tipos dos dados.	10
12	Resultado da transformação.	11
13	Definição das features e do target.	12
14	Configuração da divisão dos dados de treino e teste.	12
15	Árvore de decisão.	13
16	Florestas aleatórias.	13
17	Rede neural perceptron de múltiplas camadas.	14
18	Métricas gerais para árvore de decisão.	16
19	Métricas gerais para florestas aleatórias.	16
20	Métricas gerais para perceptron de múltiplas camadas.	16
21	Matriz de confusão para árvore de decisão.	17
22	Matriz de confusão para florestas aleatórias.	18
23	Matriz de confusão para perceptron de múltiplas camadas.	18

LISTA DE TABELAS

1	Exemplo de matriz de confusão para duas classes.	17
---	--	----

SUMÁRIO

1	INTRODUÇÃO	5
2	METODOLOGIA	5
2.1	Ferramentas	5
2.2	Seleção de atributos (Feature selection)	6
2.3	Correção de dados inválidos	6
2.4	Tratamento de dados ausentes	7
2.5	Tratamento de ruídos e outliers	7
2.6	Balanceamento dos classes	9
2.7	Normalização dos dados numéricos	10
2.8	Codificação das variáveis categóricas	11
2.9	Seleção das features e do target	12
2.10	Divisão dos dados de treinamento e teste	12
2.11	Escolha dos algoritmos	12
2.12	Busca de hiperparâmetros	14
2.13	Limitações	14
3	RESULTADOS E DISCUSSÕES	15
3.1	Melhores parâmetros encontrados	15
3.2	Métricas gerais	15
3.3	Matrizes de confusão	17
3.4	Discussões	18
4	CONCLUSÃO	19
	REFERÊNCIAS	20

1 INTRODUÇÃO

Este relatório aborda todas as etapas da criação de modelos de classificação para a predição do risco de extinção de cursos, utilizando um conjunto de dados dos cursos de graduação no Brasil. Esses dados são extraídos do Portal de Dados Abertos do Ministério da Educação (MEC, 2022). Dada a enorme quantidade de cursos, ter uma compreensão das características que compõem os cursos em processo de extinção e já extintos ajuda as organizações acadêmicas a tomar melhores decisões.

O propósito deste relatório é detalhar como o autor chegou aos resultados alcançados, de modo que qualquer pessoa que deseje replicar meu experimento consiga fazê-lo e tenha uma base para, inclusive, realizar melhorias no processo. Assim, o principal problema que se pretende resolver é prever o "risco" de extinção de um curso, classificando-o em três categorias: *Em atividade*, *Em extinção* e *Extinto*.

Neste relatório, serão abordadas todas as etapas de pré-processamento, incluindo: seleção de atributos, correção de dados inválidos, tratamento de dados ausentes, tratamento de outliers, balanceamento, normalização e transformação. Também será abordada a construção dos modelos, incluindo a escolha dos algoritmos, seleção de atributos, divisão dos dados de treinamento e teste, configuração de hiperparâmetros e os resultados das métricas.

Este trabalho, além de contribuir com a educação superior no Brasil, também contribui para o campo de pesquisa chamado Learning Analytics (WIKIPEDIA,), onde é utilizado a área de ciência de dados para analisar e propor soluções para o campo educacional em geral.

2 METODOLOGIA

2.1 Ferramentas

- Google Colab
- Python 3.10.12
- Pandas 2.1.4
- Numpy 1.26.4
- Matplotlib 3.7.1
- Seaborn 0.13.1
- os
- gdown 4.7.3
- Scikit-learn 1.3.2
- category-encoders 2.6.3
- Joblib 1.4.2

2.2 Seleção de atributos (Feature selection)

A primeira etapa foi extrair os atributos mais relevantes Figura [1] e descartar variáveis que não são importantes para o problema em questão Figura [2].

```
[ ] df.columns
In[ ]: Index(['CATEGORIA_ADMINISTRATIVA', 'ORGANIZACAO_ACADEMICA', 'GRAU',
             'MODALIDADE', 'SITUACAO_CURSO', 'QT_VAGAS_AUTORIZADAS', 'CARGA_HORARIA',
             'REGIAO'],
            dtype='object')
```

Figura 1 – Atributos selecionados.

```
[ ] df.drop(columns=['CODIGO_IES', 'NOME_IES',
                    'CODIGO_CURSO', 'NOME_CURSO',
                    'AREA_OCDE', 'CODIGO_AREA_OCDE_CINE',
                    'AREA_OCDE_CINE', 'CODIGO_MUNICIPIO',
                    'MUNICIPIO', 'UF'], inplace=True)
```

Figura 2 – Atributos descartados.

2.3 Correção de dados inválidos

Nesta etapa foi analisada quais categorias estão presentes em cada atributo, e em caso de dados inválidos, corrigir.

```
[ ] df['CATEGORIA_ADMINISTRATIVA'].unique()
In[ ]: array(['Privada com fins lucrativos', 'Privada sem fins lucrativos',
             'Pública Municipal', 'Pública Federal', 'Pública Estadual',
             'Especial'], dtype=object)

[ ] df['ORGANIZACAO_ACADEMICA'].unique()
In[ ]: array(['Centro Universitário', 'Universidade', 'Faculdade',
             'Instituto Federal de Educação, Ciência e Tecnologia',
             'Centro Federal de Educação Tecnológica',
             'Instituição Especialmente Credenciada para oferta de cursos lato sensu'],
            dtype=object)

[ ] df['GRAU'].unique()
In[ ]: array(['Bacharelado', 'Licenciatura', 'Tecnológico',
             'Área Básica de Ingresso (ABI)', 'Sequencial'], dtype=object)

[ ] df['MODALIDADE'].unique()
In[ ]: array(['Educação a Distância', 'Educação Presencial'], dtype=object)

[ ] df['SITUACAO_CURSO'].unique()
In[ ]: array(['Em atividade', 'Em extinção', 'Extinto'], dtype=object)

[ ] df['REGIAO'].unique()
In[ ]: array(['SUDESTE', 'NORDESTE', 'SUL', 'CENTRO-OESTE', 'NORTE',
             'IGNORADO/EXTERIOR'], dtype=object)
```


Figura 3 – Correção de dados inválidos.

Como visto na Figura [3] não existe categorias inválidas para o problema.

2.4 Tratamento de dados ausentes

Identificar e tratar dados faltantes é primordial no contexto de algoritmos de aprendizado de máquina, uma vez que pode afetar o desempenho e a precisão, desviando a capacidade do modelo capturar os padrões corretos. Além disso, tratar dados ausentes também garante que possamos utilizar uma gama mais ampla de algoritmos sem ter problemas técnicos.

```
[ ] df.isnull().sum()
```



	0
CATEGORIA_ADMINISTRATIVA	0
ORGANIZACAO_ACADEMICA	0
GRAU	0
MODALIDADE	0
SITUACAO_CURSO	0
QT_VAGAS_AUTORIZADAS	0
CARGA_HORARIA	0
REGIAO	0

dtype: int64

Figura 4 – Tratamento de dados ausentes.

Como visto na Figura [4], felizmente a qualidade dos dados em termos de dados ausentes está excelente, já que não apresenta eles.

2.5 Tratamento de ruídos e outliers

Como o conjunto de dados contém dados numéricos para dois atributos, é ideal identificá-los e tratá-los. Primeiramente, é importante identificar os outliers utilizando a visualização com o gráfico de box plot.

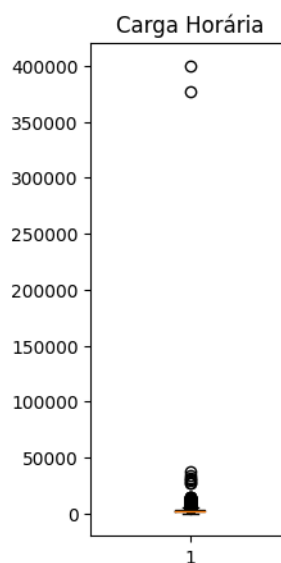


Figura 5 – Box plot da Quantidade de Vagas Autorizadas.

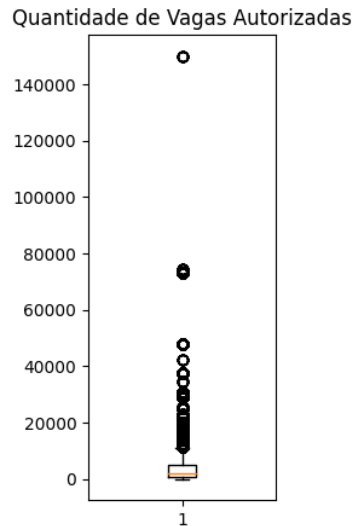


Figura 6 – Box plot de Carga Horária.

É claramente perceptível a presença de outliers e ruídos nesses atributos (Figuras [5] [6]). Para lidar com isso, será aplicado o método do Intervalo Interquartil (Interquartile Range, em inglês), mais conhecido como IQR. O IQR é uma técnica estatística utilizada para medir a dispersão central de um conjunto de dados, calculando a diferença entre o terceiro quartil e o primeiro quartil. Ao representar a faixa em que se concentram os 50% centrais dos dados, ele é resistente a outliers (ORACLE,).

Com isso, os ruídos e outliers presentes no conjunto de dados foram removidos, resultando numa perda de 8.6%, o que é uma perda proporcionalmente pequena de dados.

Após a aplicação da técnica, será possível observar como a distribuição dos dados melhorou (Figuras [7][8]).

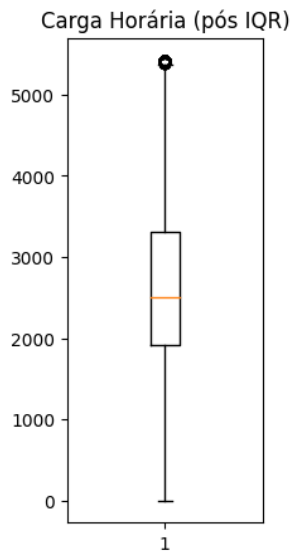


Figura 7 – Box plot da Carga Horária com IQR.

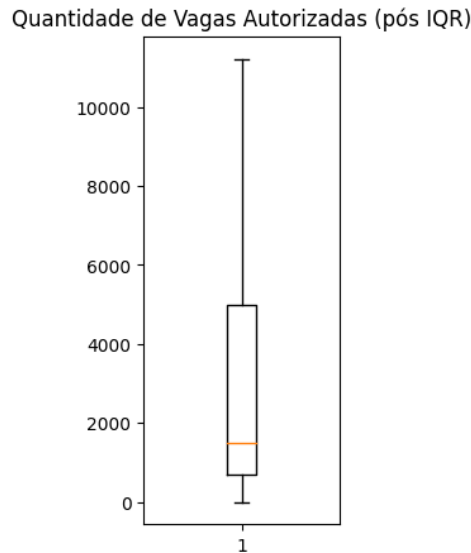


Figura 8 – Box plot de Carga Horária com IQR.

2.6 Balanceamento dos classes

Nesta etapa, que é de extrema importância para algoritmos de classificação, ocorre o balanceamento dos dados.

O balanceamento dos dados garante que o modelo consiga aprender a diferenciar corretamente as classes, especialmente quando elas são desproporcionais. Por exemplo, em caso de classes desproporcionalmente altas, o modelo pode tender a prever essa classe majoritária em relação às outras, ignorando as classes minoritárias.

As técnicas mais comuns incluem **oversampling** (aumentar a classe minoritária), **undersampling** (diminuir a classe majoritária) e o **equilíbrio de pesos**, usando algoritmos que adotam pesos para cada classe.

Nesse contexto, foi escolhido o uso de undersampling para aproveitar a redução do tamanho do dataset e, conseqüentemente, diminuir o tempo de treinamento dos algoritmos de classificação.

```
[ ] df['SITUACAO_CURSO'].value_counts()
```


	count
SITUACAO_CURSO	
Em atividade	776097
Extinto	39017
Em extinção	10207

dtype: int64

Figura 9 – Classes desbalanceadas.

Na Figura [9], é mostrada a frequência de cada classe, onde a classe *Em atividade* é a majoritária e *Em extinção* é a classe minoritária. Após a aplicação do undersampling, o conjunto de dados terá pouco mais de 30 mil registros (Figura [10]).

```
[ ] df['SITUACAO_CURSO'].value_counts()
```



	count
SITUACAO_CURSO	
Em atividade	10207
Em extinção	10207
Extinto	10207

dtype: int64

Figura 10 – Classes balanceadas.


2.7 Normalização dos dados numéricos

É interessante normalizar os atributos numéricos para garantir que todas as características contribuam igualmente para o treinamento do modelo, independentemente de suas escalas originais. Muitos algoritmos de classificação calculam distâncias entre dados ou usam funções que podem ser sensíveis à escala das variáveis. Se as características não forem normalizadas, aquelas com maiores magnitudes podem dominar o cálculo de distâncias e influenciar desproporcionalmente as decisões do modelo, levando a resultados imprecisos.

Duas das técnicas mais comuns são **Min-Max Scaling** e **Z-score normalization** (padronização ou standardization). O Min-Max Scaling transforma os dados para que cada característica seja escalada entre um intervalo específico, geralmente entre 0 e 1. Por outro lado, a Z-score normalization transforma os dados para que cada característica tenha uma média de 0 e um desvio padrão de 1.

No dataset analisado, há dois atributos numéricos: QT_VAGAS_AUTORIZADAS e CARGA_HORARIA (Figura [11]). A normalização será aplicada a esses atributos, utilizando a técnica de Min-Max Scaling.

```
[ ] df.dtypes
```



	0
CATEGORIA_ADMINISTRATIVA	object
ORGANIZACAO_ACADEMICA	object
GRAU	object
MODALIDADE	object
SITUACAO_CURSO	object
QT_VAGAS_AUTORIZADAS	int64
CARGA_HORARIA	int64
REGIAO	object

Figura 11 – Tipos dos dados.

2.8 Codificação das variáveis categóricas

Por fim, a última etapa do pré-processamento é a codificação das variáveis categóricas. Essa etapa é fundamental em algoritmos de classificação porque a maioria deles opera com dados numéricos. As variáveis categóricas precisam ser convertidas em um formato numérico para serem utilizadas de forma efetiva; caso contrário, essas variáveis não poderão ser interpretadas adequadamente pelo algoritmo, resultando tanto em erros quanto em desempenho insatisfatório.

Existem muitas técnicas de codificação. O **Label Encoding**, por exemplo, atribui um número inteiro único a cada categoria. É simples e eficiente, mas pode introduzir uma ordem indesejada entre as categorias. É adequado quando as categorias têm uma ordem intrínseca, como tamanhos ("pequeno", "médio", "grande"). Já o **One-Hot Encoding** cria uma nova coluna binária (0 ou 1) para cada categoria, representando a presença ou ausência dessa categoria. É ideal para variáveis sem ordem intrínseca e funciona bem com algoritmos que não assumem relacionamentos ordinais, como redes neurais e regressão logística.

Outras técnicas bastante utilizadas incluem o **Ordinal Encoding**, que é similar ao Label Encoding, mas os números atribuídos refletem uma ordem específica das categorias. É útil para variáveis categóricas com uma ordem implícita, como classificações ("ruim", "médio", "bom"). Há também o **Binary Encoding**, que converte categorias em números binários e depois transforma cada bit em uma nova coluna. Isso reduz a dimensionalidade em comparação com o One-Hot Encoding, tornando-o útil para conjuntos de dados com muitas categorias, evitando a explosão dimensional.

Com base nos motivos supracitados, será utilizado o Binary Encoding para as features e o Label Encoding para o target (SITUACAO_CURSO).

CATEGORIA_ADMINISTRATIVA_0	int64
CATEGORIA_ADMINISTRATIVA_1	int64
CATEGORIA_ADMINISTRATIVA_2	int64
ORGANIZACAO_ACADEMICA_0	int64
ORGANIZACAO_ACADEMICA_1	int64
ORGANIZACAO_ACADEMICA_2	int64
GRAU_0	int64
GRAU_1	int64
GRAU_2	int64
MODALIDADE_0	int64
MODALIDADE_1	int64
REGIAO_0	int64
REGIAO_1	int64
REGIAO_2	int64
SITUACAO_CURSO_encoded	int64
QT_VAGAS_AUTORIZADAS	float64
CARGA_HORARIA	float64

Figura 12 – Resultado da transformação.

2.9 Seleção das features e do target

Finalmente, após toda a etapa de pré-processamento, tem-se a garantia de que os dados estão adequados para os algoritmos de classificação. Agora, é necessário definir quais são as features e qual é o target (Figura [13]).

```
[ ] features = ['CATEGORIA_ADMINISTRATIVA_0', 'CATEGORIA_ADMINISTRATIVA_1', 'CATEGORIA_ADMINISTRATIVA_2',
                'ORGANIZACAO_ACADEMICA_0', 'ORGANIZACAO_ACADEMICA_1', 'ORGANIZACAO_ACADEMICA_2',
                'GRAU_0', 'GRAU_1', 'GRAU_2',
                'MODALIDADE_0', 'MODALIDADE_1',
                'REGIAO_0', 'REGIAO_1', 'REGIAO_2',
                'QT_VAGAS_AUTORIZADAS',
                'CARGA_HORARIA']

target = 'SITUACAO_CURSO_encoded'

X = df_transformed[features]
y = df_transformed[target]
```

Figura 13 – Definição das features e do target.

2.10 Divisão dos dados de treinamento e teste

Agora, deve-se dividir o conjunto de dados em dados de treinamento e dados de teste (Figura [14]).

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figura 14 – Configuração da divisão dos dados de treino e teste.

2.11 Escolha dos algoritmos

Para o problema em questão, foi escolhida a utilização de três algoritmos de classificação:

- **Árvore de Decisão** (Decision Tree)
- **Florestas Aleatórias** (Random Forest)
- **Rede Neural Perceptron de Múltiplas Camadas** (Multi-Layer Perceptron)

O algoritmo de **Árvore de Decisão** é uma técnica de aprendizado de máquina que cria um modelo em forma de árvore para tomar decisões. Cada nó na árvore representa uma pergunta sobre um atributo dos dados, cada ramo representa o resultado da pergunta, e cada folha representa uma decisão final ou uma previsão. O modelo é construído dividindo os dados em grupos com base em critérios que maximizam a pureza dos grupos. Árvores de decisão são fáceis de interpretar e podem lidar com dados numéricos e categóricos. No entanto, podem sofrer de overfitting e são instáveis, o que pode ser mitigado por técnicas como a poda e o uso de métodos de ensemble (técnica de aprendizado de máquina que combina múltiplos modelos para melhorar a performance geral), como Random Forests (LEARN, a). A escolha de utilizar a Árvore de Decisão deve-se ao fato de ela ser eficaz para dados categóricos e quando é necessário um modelo interpretável.

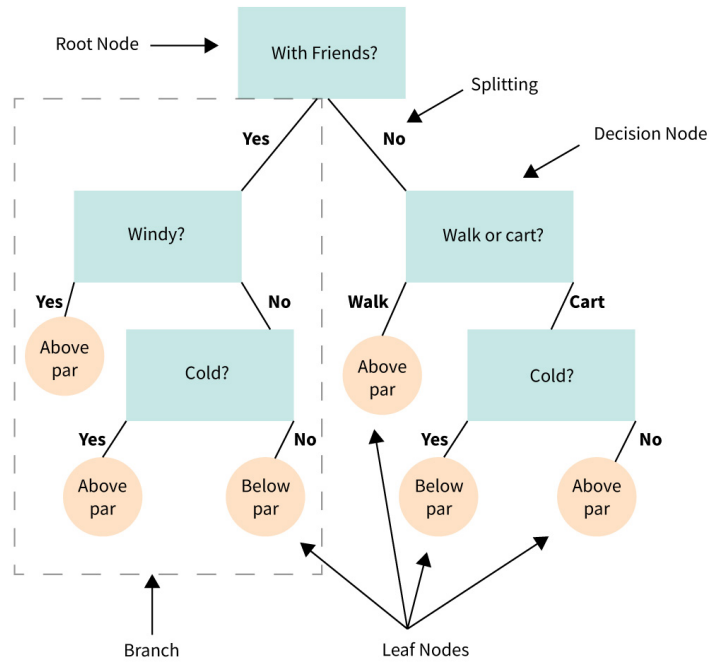


Figura 15 – Árvore de decisão.

O **Florestas Aleatórias** é um algoritmo de aprendizado de máquina baseado em ensemble que utiliza múltiplas árvores de decisão para fazer previsões. Em vez de construir uma única árvore, o Random Forest cria várias árvores de decisão durante o treinamento, cada uma baseada em um subconjunto diferente dos dados e características. A decisão final é obtida por votação majoritária para classificação ou média para regressão entre as previsões de todas as árvores. Esse método reduz o risco de overfitting e melhora a precisão do modelo, tornando-o mais robusto e estável em comparação com uma única árvore de decisão (LEARN, c). A opção pelo uso das Florestas Aleatórias é devido à sua robustez contra overfitting e à sua eficácia em problemas complexos e com grandes conjuntos de dados.

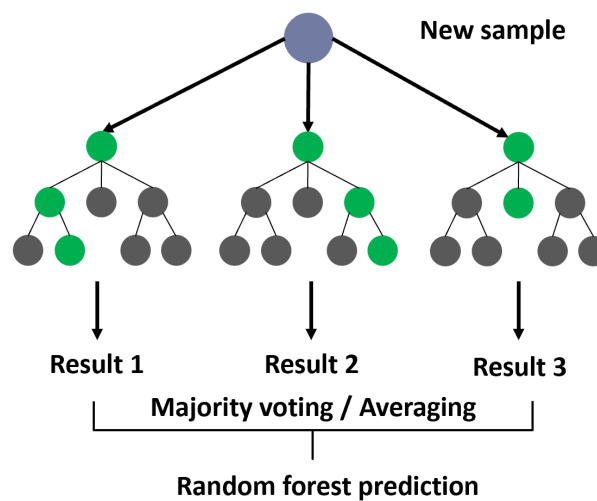


Figura 16 – Florestas aleatórias.

Um **Perceptron de Múltiplas Camadas** (da sigla MLP em inglês) é uma rede neural artificial composta por várias camadas de neurônios. Ele tem uma camada de entrada, uma ou mais camadas ocultas, e uma camada de saída. Cada neurônio em uma camada está conectado a todos os neurônios da camada seguinte, e usa funções de ativação não lineares para capturar padrões complexos nos dados. O MLP é treinado usando algoritmos de retropropagação para ajustar os pesos das conexões e minimizar o erro de previsão (LEARN, b). A Rede Neural Perceptron de Múltiplas Camadas foi escolhida por sua capacidade de capturar relações complexas e não lineares nos dados.

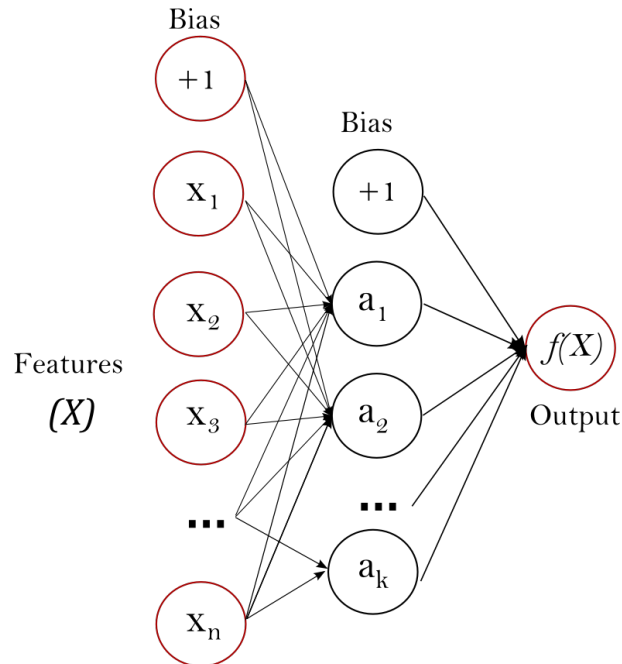


Figura 17 – Rede neural perceptron de múltiplas camadas.

2.12 Busca de hiperparâmetros

Ainda é necessário definir a técnica para encontrar a melhor combinação de hiperparâmetros para o modelo. Existem duas técnicas muito conhecidas: **Grid Search** e **Random Search**. O Grid Search explora de forma exaustiva todas as combinações possíveis de hiperparâmetros especificados em uma grade predefinida. A vantagem é que garante encontrar a melhor combinação dentro do espaço definido, mas pode ser computacionalmente caro e demorado, especialmente com muitos hiperparâmetros e valores.

O Random Search, por outro lado, amostra aleatoriamente combinações de hiperparâmetros dentro de um intervalo definido. Geralmente, é mais eficiente porque não testa todas as combinações, permitindo encontrar boas configurações mais rapidamente e com menor custo computacional. Embora não garanta encontrar a melhor combinação, pode ser mais prático e eficaz para grandes espaços de busca.

Com base nisso, foi decidido optar pela técnica Random Search, pois o custo de processamento tem um peso maior na escolha do autor.

2.13 Limitações

A principal limitação da metodologia adotada foi optar por técnicas de menor custo computacional, a fim de minimizar o tempo de busca pelo melhor conjunto de hiperparâmetros.

3 RESULTADOS E DISCUSSÕES

3.1 Melhores parâmetros encontrados

Árvore de decisão:

```
min_samples_split: 2,  
min_samples_leaf: 1,  
max_depth: 30,  
criterion: gini
```

Florestas aleatórias:

```
n_estimators: 300,  
min_samples_split: 2,  
min_samples_leaf: 1,  
max_features: sqrt,  
max_depth: 40,  
bootstrap: True
```

Perceptron de múltiplas camadas:

```
solver: adam,  
learning_rate: adaptive,  
hidden_layer_sizes: (100, 50),  
batch_size: 64,  
alpha: 0.0001,  
activation: tanh
```

3.2 Métricas gerais

Este trabalho utiliza algumas métricas para avaliar o desempenho dos modelos de classificação. A **acurácia** (accuracy) é a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. A **precisão** (precision) é a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas pelo modelo. O **recall** (revocação ou sensibilidade) é a proporção de verdadeiros positivos identificados corretamente em relação ao total de positivos reais. O **F1-score** é a média harmônica entre a precisão e o recall, proporcionando um equilíbrio entre essas duas métricas, especialmente útil quando há uma distribuição desigual entre as classes. Por fim, o **suporte** (support) refere-se ao número de ocorrências de cada classe no conjunto de dados, indicando o tamanho da classe e ajudando a entender o equilíbrio das classes no dataset.


```

Acurácia: 0.8029387755102041
Matriz de Confusão:
[[1841  86 110]
 [  54 1624 375]
 [  94  488 1453]]
Relatório de Classificação:
      precision    recall  f1-score   support

      0         0.93      0.90      0.91      2037
      1         0.74      0.79      0.76      2053
      2         0.75      0.71      0.73      2035

 accuracy          0.80          0.80          0.80          6125
 macro avg         0.80          0.80          0.80          6125
 weighted avg      0.80          0.80          0.80          6125

```

Figura 18 – Métricas gerais para árvore de decisão.

```

Acurácia: 0.8070204081632653
Matriz de Confusão:
[[1839  82 116]
 [  36 1605 412]
 [  80  456 1499]]
Relatório de Classificação:
      precision    recall  f1-score   support

      0         0.94      0.90      0.92      2037
      1         0.75      0.78      0.77      2053
      2         0.74      0.74      0.74      2035

 accuracy          0.81          0.81          0.81          6125
 macro avg         0.81          0.81          0.81          6125
 weighted avg      0.81          0.81          0.81          6125

```

Figura 19 – Métricas gerais para florestas aleatórias.

```

Acurácia: 0.7324081632653061
Matriz de Confusão:
[[1713 117 207]
 [  79 1407 567]
 [ 183  486 1366]]
Relatório de Classificação:
      precision    recall  f1-score   support

      0         0.87      0.84      0.85      2037
      1         0.70      0.69      0.69      2053
      2         0.64      0.67      0.65      2035

 accuracy          0.73          0.73          0.73          6125
 macro avg         0.74          0.73          0.73          6125
 weighted avg      0.74          0.73          0.73          6125

```

Figura 20 – Métricas gerais para perceptron de múltiplas camadas.

3.3 Matrizes de confusão

A Matriz de Confusão é uma tabela que descreve o desempenho do modelo de classificação. Ela mostra o número de previsões corretas e incorretas agrupadas por classe. Principais componentes da matriz de confusão:

- **Verdadeiros Positivos (TP)**: Previsões corretas para a classe positiva.
- **Falsos Positivos (FP)**: Previsões incorretas para a classe positiva.
- **Verdadeiros Negativos (TN)**: Previsões corretas para a classe negativa.
- **Falsos Negativos (FN)**: Previsões incorretas para a classe negativa.

	Predito Classe 1	Predito Classe 2
Verdadeiro Classe 1	TP	FP
Verdadeiro Classe 2	FN	TN

Tabela 1 – Exemplo de matriz de confusão para duas classes.

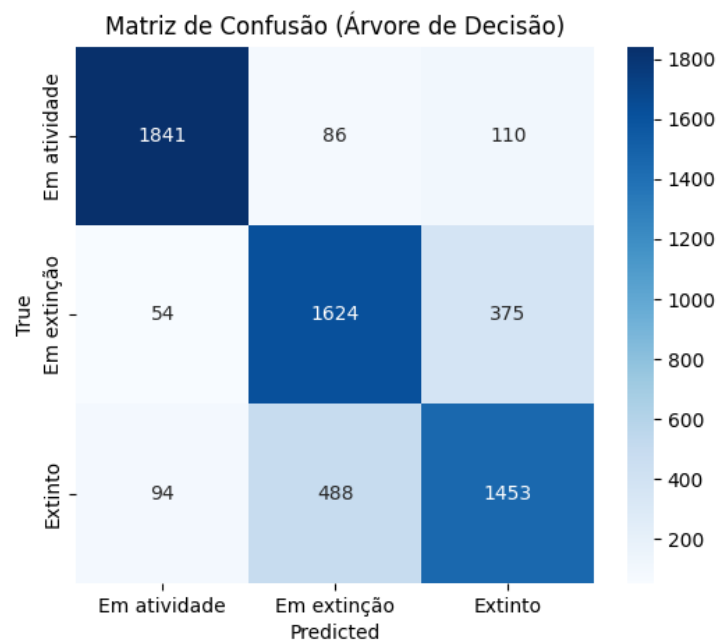


Figura 21 – Matriz de confusão para árvore de decisão.

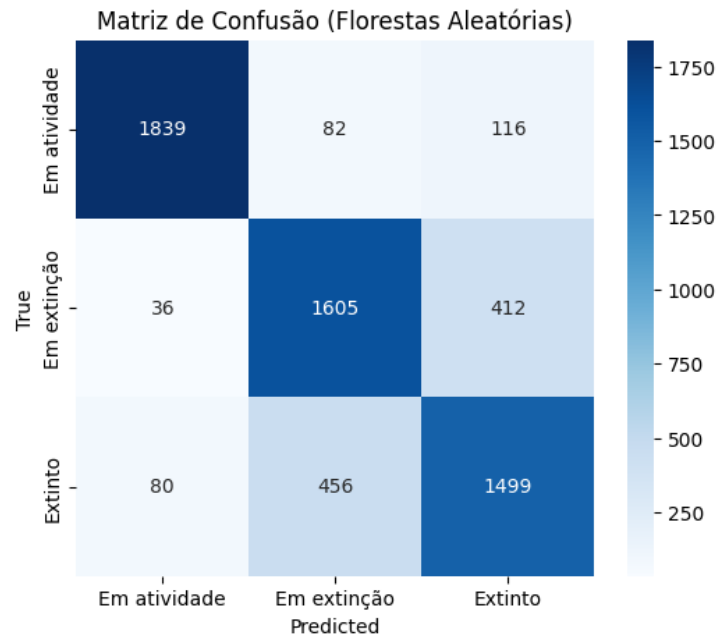


Figura 22 – Matriz de confusão para florestas aleatórias.

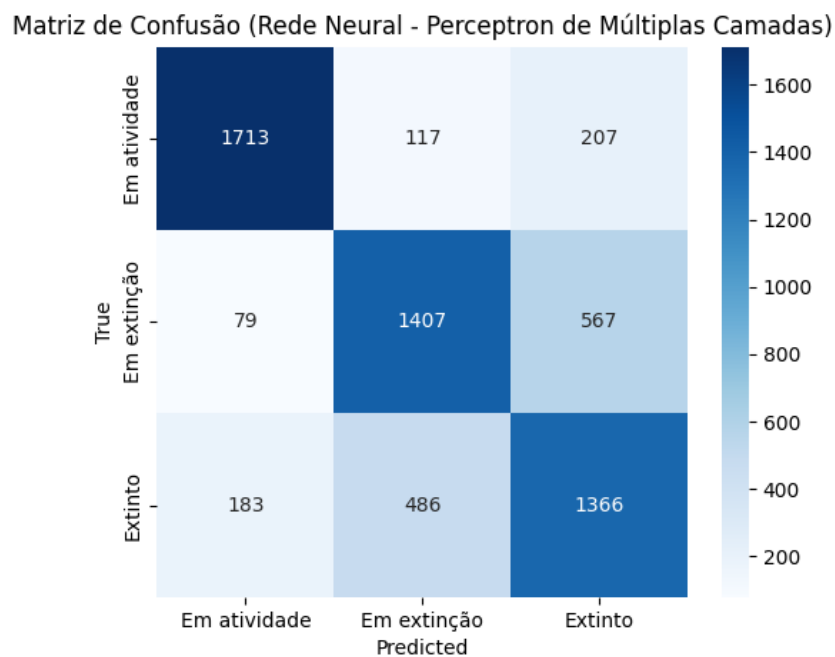


Figura 23 – Matriz de confusão para perceptron de múltiplas camadas.

3.4 Discussões

A maior acurácia foi obtida pelo algoritmo de florestas aleatórias (Figura [19]), que também apresentou as melhores métricas para a classe principal do problema, que são os cursos extintos. A rede neural MLP (Figura [20]) teve a menor acurácia. Essa acurácia poderia ser melhorada através de técnicas de aumento de dados, equilíbrio de pesos e pesquisa em grade. No entanto, conforme mencionado anteriormente, os fatores tempo e custo computacional eram primordiais, por isso foram escolhidos métodos de menor custo computacional.

É interessante perceber que nas matrizes de confusão da (Seção [3.3]), os algoritmos cometeram menos erros para cursos *Em atividade* e tiveram maior dificuldade em diferenciar entre as classes *Em extinção* e *Extinto*.

Embora uma acurácia de 81% seja considerada boa, é provável que as features do conjunto de dados não sejam suficientes para classificar com exatidão nosso alvo.

4 CONCLUSÃO

O principal resultado foi a obtenção de um modelo com 81% de acurácia usando o algoritmo de florestas aleatórias. Esse resultado reflete o objetivo do trabalho, que era desenvolver uma ferramenta de predição de risco de extinção de um curso de graduação no Brasil. Esse achado tem uma aplicação prática em que representantes superiores de organizações acadêmicas podem se beneficiar. As limitações desse projeto estão relacionadas ao próprio conjunto de dados, que carece de mais features para realizar classificações mais precisas. Além disso, poderia-se usar a técnica de Grid Search e validação cruzada para analisar como os algoritmos se comportariam diante dessas técnicas. O próximo passo do trabalho é desenvolver uma ferramenta que utilize uma API para se comunicar com o modelo disponibilizado na web. Assim, este projeto é de extrema relevância para a educação superior no Brasil.

Em relação a considerações finais do trabalho, foi interessante observar como o tempo de treinamento é bastante diferente entre uma árvore de decisão simples e uma rede neural, como o perceptron de múltiplas camadas. O projeto contribuiu para o aprimoramento das habilidades de construção de modelos com scikit-learn e métodos de pré-processamento em geral, bem como na comunicação dos resultados, desde a escrita do relatório até a apresentação dos resultados em sala de aula. Por isso, é importante agradecer aos professores da disciplina, Prof. Dr. Daniel Sabino e Prof. Dr. Heitor Florêncio, ambos do Instituto Metrópole Digital da Universidade Federal do Rio Grande do Norte, que serviram como guias sobre quais tópicos aprender e se concentrar para desenvolver um projeto como este.

REFERÊNCIAS

LEARN scikit. Decision trees. Disponível em: <<https://scikit-learn.org/stable/modules/tree.html#>>.

LEARN scikit. Multi-layer perceptron. Disponível em: <https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron>.

LEARN scikit. Random forests. Disponível em: <<https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles>>.

MEC. Cursos de graduação do brasil. 2022. Disponível em: <<https://dadosabertos.mec.gov.br/indicadores-sobre-ensino-superior/item/183-cursos-de-graduacao-do-brasil>>.

ORACLE. Intervalo entre quartis. *Oracle Help Center*. Disponível em: <https://docs.oracle.com/cloud/help/pt_BR/pbcs_common/PFUSU/insights_metrics_IQR.htm#PFUSU-GUID-CF37CAEA-730B-4346-801E-64612719FF6B>.

WIKIPEDIA. Learning analytics. Disponível em: <https://en.wikipedia.org/wiki/Learning_analytics>.