UNIVERSIDADE FEDERAL DO CEARÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO DO MESTRADO PROFISSIONAL EM POLÍTICAS PÚBLICAS E GESTÃO DA EDUCAÇÃO SUPERIOR - POLEDUC

GERLÂNDIA ALVES DA SILVA

SOFTWARE: AMBIENTE DE COLETA DE DADOS E SISTEMA PARA IDENTIFICAÇÃO DE RENDIMENTO ACADÊMICO

Produto técnico oriundo dos resultados da dissertação intitulada de Retenção na Educação Superior: uma investigação baseada em mineração de dados na UFC - Campus de Crateús pertencente ao Mestrado Profissional em Políticas Públicas e gestão da Educação Superior - POLEDUC.

Orientadora: Profª. Drª. Adriana Castro Araújo
Coorientador: Prof. Dr. José Wellington Franco da Silva

FORTALEZA, 2026

SUMÁRIO

1	**INTRODUÇÃO**	4
2	**PÚBLICO-ALVO**……..……	4
3	**OBJETIVOS**……..	4
4	**SITUAÇÃO PROBLEMA**……	5
5	**ARQUITETURA E TECNOLOGIAS UTILIZADAS** ……..	6
6	**MÓDULO DE AVALIAÇÃO** ……	7
8	**CONCLUSÃO**……	19
9	**REFERÊNCIAS**……..	19

RESUMO

O estudo tem como objetivo geral desenvolver uma ferramenta computacional para o monitoramento do desempenho acadêmico dos estudantes, utilizando técnicas de Mineração de Dados Educacionais (MDE) para a verificação de casos de retenção, na Universidade Federal do Ceará (UFC) - Campus de Crateús. O acompanhamento do desempenho é realizado por meio de processos manuais de extração e análise de dados provenientes do Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA), o que torna a atividade morosa, pouco eficiente e limitadora da implementação de ações institucionais preventivas. O Ambiente de Coleta de Dados e Sistema para Identificação de Rendimento Acadêmico ou DEYSI, *Data Harvest and System for Yield,* tende a contribuir com a gestão universitária no que diz respeito a situações de retenção, que possam comprometer a permanência e o êxito estudantil. Com o sistema, espera-se a redução do retrabalho, a padronização das análises semestrais, o fortalecimento da tomada de decisão baseada em evidências e a implementação de ações institucionais de ordem pedagógica e acadêmica, buscando melhorar os indicadores de sucesso dos cursos, o desempenho dos estudantes, o bem-estar e a qualidade de vida no contexto universitário.

Palavras-chave: gestão universitária, permanência; mineração de dados; retenção discente.

1 INTRODUÇÃO

A UFC *campus* de Crateús, buscando acompanhar o desempenho dos estudantes, verifica o rendimento acadêmico dos matriculados nos cursos de graduação a cada semestre letivo. Os downloads dos documentos obtidos no SIGAA e a extração de dados são executados de forma manual pelos servidores que compõem a Divisão de Apoio Educacional (DAE), o que torna o processo exaustivo, lento e demorado.

Nesse contexto, considerando a situação vivenciada e demandada pela DAE, a pesquisa propõe-se a desenvolver uma ferramenta computacional, com base MDE, para auxiliar os gestores da universidade e os setores vinculados à gestão acadêmica e à assistência estudantil, no monitoramento e acompanhamento do desempenho dos discentes.

No âmbito educacional, o sistema DEYSI, ao disponibilizar uma base de dados confiável e segura, por eliminar falhas humanas, contribui para identificar situações que possam comprometer a permanência e o êxito estudantil em decorrência das dificuldades acadêmicas. O olhar sobre a trajetória percorrida pelos discentes, desde o ingresso até a conclusão do curso, colabora com a gestão nas tomadas de decisões das ações estratégicas de apoio institucional, acadêmico, pedagógico e psicológico, no enfrentamento da retenção.

2 PÚBLICO-ALVO

O produto técnico, a princípio, se destina ao campus da UFC em Crateús, em específico, à DAE, estendendo-se posteriormente à Coordenação de Programas Acadêmicos, às Coordenações de curso e à Secretaria Acadêmica desta Instituição Federal de Educação Superior (IFES), setores envolvidos com a gestão e/ou atividades administrativas de ordem acadêmica.

O DEYSI, uma vez implementado na UFC *campus* de Crateús, poderá ser ampliado para os órgãos de atividades específicas, a exemplo da Pró-Reitoria de Graduação (PROGRAD), da Pró-Reitoria de Assistência Estudantil (PRAE) e de outros *campi* do interior, podendo ser compartilhada com outras IES públicas.

3 OBJETIVOS

Reconhecendo que as inovações tecnológicas, com a aplicação de técnicas de MDE, podem auxiliar os gestores a buscar alternativas para reduzir os indicadores de

insucesso, retenção e evasão (Nunes, 2023), este trabalho tem como objetivo geral desenvolver uma ferramenta computacional para o monitoramento do desempenho acadêmico dos estudantes, utilizando técnicas de MDE para a verificação de casos de retenção, na Universidade Federal do Ceará (UFC) - Campus de Crateús. A fim de alcançar o objetivo proposto, foram elencados os seguintes objetivos específicos:

a) Definir os requisitos funcionais e não funcionais do sistema DEYSI;
b) Projetar a arquitetura para o DEYSI, utilizando algoritmos de Inteligência Artificial;
c) Implementar técnicas de MDE para identificar casos de retenção discente.

4 SITUAÇÃO PROBLEMA

Na UFC *campus* de Crateús, o acompanhamento do desempenho acadêmico e a identificação dos estudantes em situações de retenção estão condicionados, sobretudo, a fatores de ordem institucional e operacional. Destaca-se, nesse contexto, a dependência de processos manuais para a extração, a organização e a análise de dados referente ao percurso curricular dos alunos.

Essa dinâmica operacional limita a sistematização das informações e dificulta a análise integrada da trajetória acadêmica dos estudantes, tornando o monitoramento moroso e pouco eficiente. Outrossim, o caráter manual da atualização da base de dados e da consolidação das análises demanda um tempo elevado a cada semestre, o que reduz a disponibilidade dos setores envolvidos para a execução de ações institucionais preventivas e para o aprimoramento das estratégias já existentes voltadas para a melhoria do desempenho discente.

Como desdobramentos da implementação da ferramenta computacional, espera-se, em curto prazo,[1] diminuição do retrabalho, atualização mais frequente da base de dados e padronização das análises semestrais. No médio e longo prazo[2], almeja-se a priorização da oferta de disciplinas mediante o perfil do aluno com dificuldades acadêmicas, definição mais objetiva de ações institucionais, pedagógicas, e assistenciais, melhoria gradual dos indicadores acadêmicos e racionalização do uso de recursos institucionais (materiais e humano).

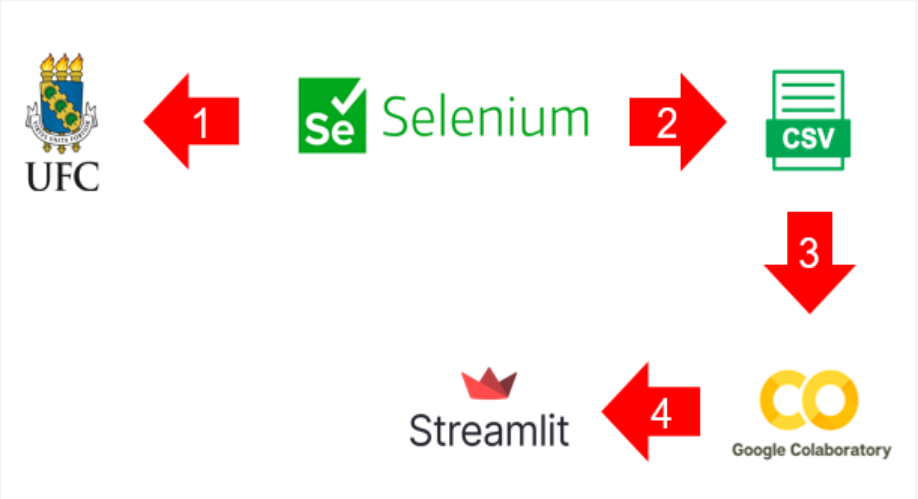
[1] Curto prazo: entre 1 e 2 anos
[2] Médio e longo prazo: acima de 2 anos

Portanto, a proposta de desenvolvimento do sistema DEYSI articula-se diretamente com o objetivo geral deste trabalho, ao buscar apoiar a gestão na identificação de casos de retenção discente e no monitoramento sistemático do rendimento acadêmico dos estudantes. Conclui-se que, com a otimização dos processos de atualização e análise dos dados acadêmicos, o fortalecimento da tomada de decisão baseada em evidências e a qualificação das ações institucionais voltadas à permanência e ao êxito estudantil, a ferramenta tende a contribuir para melhorar os indicadores de sucesso desta IFES.

5 ARQUITETURA

O sistema foi implementado na linguagem Python 3.11 e Python 3.13, fazendo uso de diversas bibliotecas e ferramentas para cumprir suas funcionalidades. O DEYSI possui a arquitetura da Pipeline de Extração, Transformação e Carregamento do (ETL), adotando o *Apache Airflow* para a coordenação da ETL e o *Docker* para a conteinerização. Na Figura 1, visualiza-se as etapas pelas quais ocorreram a estruturação da ferramenta.

Figura 1 - Visão geral da arquitetura do sistema DEYSI



Fonte: Dados da pesquisa (2025)

A etapa 1 utilizou o Selenium WebDriver para automatizar a interação com o navegador *Web* com o portal do SIGAA da UFC. Nesse processo, o Selenium realiza as seguintes tarefas: faz o login no SIGAA, entra no Portal Coord. Graduação, seleciona o curso e na janela “aluno - consulta avançada” obtém os históricos no formato PDF, os arquivos são armazenados em um diretório local. Na etapa 2, o PyPDF2 executa a leitura dos textos e

tabelas dos arquivos em PDF, enquanto o Pandas recebe os textos, faz a limpeza dos dados e os organiza em planilhas no formato CSV.

Na etapa 3, o arquivo CSV é importado para o Google Colab, sendo feitas a limpeza e a transformação dos dados, buscando corrigir problemas como valores ausentes ou inconsistências. Em seguida, realiza-se a Análise Exploratória de Dados (EDA), o que consiste em examinar estatísticas descritivas, distribuições e correlações, além de gerar visualizações para compreender melhor o comportamento das variáveis. Com base nesses *insights*, desenvolve-se o modelo de Inteligência Artificial. Na etapa 4, ocorre a construção da Dashboard Interativa. Após o processamento, o mesmo arquivo CSV é utilizado no VS Code, com o auxílio da biblioteca Streamlit. Essa ferramenta permite criar uma dashboard online, na qual os resultados e métricas do modelo podem ser visualizados de forma clara e interativa.

6 MÓDULO DE AVALIAÇÃO

Na avaliação dos modelos, aplicou-se o processo de validação cruzada K-fold (k=10), integrado a um *holdout* de 80% para treinamento e 20% para teste. Foram avaliados algoritmos de diferentes paradigmas, tais como XGBoost, K-Nearest Neighbors (KNN), Logistic Regression, Random Forest e Decision Tree. Desses modelos, o que apresentou melhor desempenho foi o KNN, tendo como referência o resultado do Recall máximo (1.00), obtido por meio da Matriz de Confusão. A variabilidade de desempenho dos algoritmos foi verificada ao longo de 10 pastas de validação cruzada e o Teste de Postos Sinalizados de Wilcoxon foi aplicado sobre os resultados das 10 pastas da validação cruzada, a fim de verificar a superioridade numérica do KNN.

7 CÓDIGO DO MODELO E PARÂMETROS DE ENTRADA

Nos quadros abaixo, têm-se o código completo do modelo (Quadro 1), o que permite a qualquer pessoa copiá-lo e executá-lo (rodá-lo); e os parâmetros de entrada do modelo (Quadro 2).

Quadro 1 - Código de modelo executável por qualquer usuário

```
# Configuração do Oversampling
# oversampler = SMOTE(random_state=42)  # Equilibra as classes
```

```
outer_kf = KFold(n_splits=10, shuffle=True, random_state=42)

melhores_modelos_por_rodada = []

metrics = {'accuracy': [], 'recall': [], 'precision': [], 'f1': []}

metricas_por_modelo_recall = {'Logistic Regression': [],'KNN':
[],'Random Forest': [], 'Decision Tree': [], 'XGBoost': []}

count = 1

for train_idx, test_idx in outer_kf.split(X):
    print('-------------------Fold: ', count)
    count += 1
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    print('++++++++++++++++++++++++++++++++++++')
    print('Divisão para treino', X_train.shape)
    print('Divisão para teste', X_test.shape)
    print('Porcentagem', len(X_test) / len(X))
    print('++++++++++++++++++++++++++++++++++++')

    X_train_Divided, X_val, y_train_Divided, y_val =
train_test_split(X_train, y_train, test_size=0.2, random_state=42)

    print('++++++++++++++++++++++++++++++++++++')
    print('Divisão para treino', X_train_Divided.shape)
    print('Divisão para validação', X_val.shape)
    print('Porcentagem', len(X_val) / len(X_train))
    print('++++++++++++++++++++++++++++++++++++')
```

```
    #oversampling
    # X_train_resampled, y_train_resampled =
oversampler.fit_resample(X_train_Divided, y_train_Divided)
    # print("Distribuição das classes após o OverSampling:")
    # print(pd.Series(y_train_resampled).value_counts())


    #normalização
    final_scaler = MinMaxScaler()
    X_train_Divided_resampled_scaled =
final_scaler.fit_transform(X_train_Divided)
    X_val_scaled = final_scaler.transform(X_val)


    final_scaler_f = MinMaxScaler()
    X_train_scaled = final_scaler_f.fit_transform(X_train)
    X_test_scaled = final_scaler_f.transform(X_test)


    acc_val_lr, recall_val_lr, precision_val_lr, f1_val_lr, par_lr
= [], [], [], [], []
    acc_val_knn, recall_val_knn, precision_val_knn, f1_val_knn,
par_knn = [], [], [], [], []
    acc_val_random, recall_val_random, precision_val_random,
f1_val_random, par_random = [], [], [], [], []
    acc_val_tree, recall_val_tree, precision_val_tree, f1_val_tree,
par_tree = [], [], [], [], []
    acc_val_xgb, recall_val_xgb, precision_val_xgb, f1_val_xgb,
par_xgb = [], [], [], [], []



    for params_lr in ParameterGrid(modelo_lr):
      try:
        #Modelo lr
        lr = LogisticRegression(C=params_lr['C'],
max_iter=params_lr['max_iter'], solver=params_lr['solver'],
penalty=params_lr['penalty'])
```

```
        lr.fit(X_train_Divided_resampled_scaled, y_train_Divided)
        y_pred = lr.predict(X_val_scaled)

        acc_lr = accuracy_score(y_val, y_pred)
        acc_val_lr.append(acc_lr)

        recall_lr = recall_score(y_val, y_pred,
zero_division=np.nan)
        recall_val_lr.append(recall_lr)

        precision_lr = precision_score(y_val, y_pred,
zero_division=np.nan)
        precision_val_lr.append(precision_lr)

        f1_lr = f1_score(y_val, y_pred, zero_division=np.nan)
        f1_val_lr.append(f1_lr)

        par_lr.append(params_lr)
      except Exception as e:
          print(f"Erro ao treinar LogisticRegression com params
{params_lr}: {e}")
          continue

    print('Logistc Regression_ACC: ',
par_lr[acc_val_lr.index(max(acc_val_lr))])
    print('Logistc Regression_RECALL: ',
par_lr[recall_val_lr.index(max(recall_val_lr))])
    print('Logistc Regression_PRECISION: ',
par_lr[precision_val_lr.index(max(precision_val_lr))])
    print('Logistc Regression_F1: ',
par_lr[f1_val_lr.index(max(f1_val_lr))])

    #Modelo knn
    for params_knn in ParameterGrid(modelo_knn):
```

```
      try:
        knn =
KNeighborsClassifier(n_neighbors=params_knn['n_neighbors'],
metric=params_knn['metric'])
        knn.fit(X_train_Divided_resampled_scaled, y_train_Divided)
        y_pred = knn.predict(X_val_scaled)

        acc_knn = accuracy_score(y_val, y_pred)
        acc_val_knn.append(acc_knn)

        recall_knn = recall_score(y_val, y_pred,
zero_division=np.nan)
        recall_val_knn.append(recall_knn)

        precision_knn = precision_score(y_val, y_pred,
zero_division=np.nan)
        precision_val_knn.append(precision_knn)

        f1_knn = f1_score(y_val, y_pred, zero_division=np.nan)
        f1_val_knn.append(f1_knn)

        par_knn.append(params_knn)
      except Exception as e:
          print(f"Erro ao treinar Knn com params {params_lr}: {e}")
          continue

    print('KNN_ACC: ',
par_knn[acc_val_knn.index(max(acc_val_knn))])
    print('KNN_RECALL: ',
par_knn[recall_val_knn.index(max(recall_val_knn))])
    print('KNN_PRECISION: ',
par_knn[precision_val_knn.index(max(precision_val_knn))])
    print('KNN_F!: ', par_knn[f1_val_knn.index(max(f1_val_knn))])
```

```
    #Modelo RandomForestClassifier
    for params_random in ParameterGrid(modelo_random):
      try:
        random =
RandomForestClassifier(n_estimators=params_random['n_estimators'],
criterion=params_random['criterion'],
max_depth=params_random['max_depth'],
min_samples_split=params_random['min_samples_split'])
        random.fit(X_train_Divided_resampled_scaled,
y_train_Divided)
        y_pred = random.predict(X_val_scaled)


        acc_dtc = accuracy_score(y_val, y_pred)
        acc_val_random.append(acc_dtc)


        recall_dtc = recall_score(y_val, y_pred)
        recall_val_random.append(recall_dtc)


        precision_dtc = precision_score(y_val, y_pred)
        precision_val_random.append(precision_dtc)


        f1_dtc = f1_score(y_val, y_pred)
        f1_val_random.append(f1_dtc)


        par_random.append(params_random)
      except Exception as e:
          print(f"Erro ao treinar Random Forest com params
{params_lr}: {e}")
          continue


    print('RandomForestClassifier_ACC',
par_random[acc_val_random.index(max(acc_val_random))])
    print('RandomForestClassifier_RECALL',
par_random[recall_val_random.index(max(recall_val_random))])
```

```
    print('RandomForestClassifier_PRECISION',
par_random[precision_val_random.index(max(precision_val_random))])
    print('RandomForestClassifier_F1',
par_random[f1_val_random.index(max(f1_val_random))])


    #Modelo DecisionTreeClassifier
    for params_tree in ParameterGrid(modelo_arvore):
      try:
        tree =
DecisionTreeClassifier(criterion=params_tree['criterion'],
splitter=params_tree['splitter'],
max_depth=params_tree['max_depth'],
min_samples_split=params_tree['min_samples_split'])
        tree.fit(X_train_Divided_resampled_scaled, y_train_Divided)
        y_pred = tree.predict(X_val_scaled)


        acc_tre = accuracy_score(y_val, y_pred)
        acc_val_tree.append(acc_tre)


        recall_tre = recall_score(y_val, y_pred)
        recall_val_tree.append(recall_tre)


        precision_tre = precision_score(y_val, y_pred)
        precision_val_tree.append(precision_tre)


        f1_tre = f1_score(y_val, y_pred)
        f1_val_tree.append(f1_tre)


        par_tree.append(params_tree)


      except Exception as e:
          print(f"Erro ao treinar Arvore de decisão com params
{params_lr}: {e}")
          continue
```

```
    print('DecisionTreeClassifier_ACC',
par_tree[acc_val_tree.index(max(acc_val_tree))])
    print('DecisionTreeClassifier_RECALL',
par_tree[recall_val_tree.index(max(recall_val_tree))])
    print('DecisionTreeClassifier_PRECISION',
par_tree[precision_val_tree.index(max(precision_val_tree))])
    print('DecisionTreeClassifier_F1',
par_tree[f1_val_tree.index(max(f1_val_tree))])


    #Modelo XGBoost
    for params_xgb in ParameterGrid(modelo_xgb):
      try:
        xgb = XGBClassifier(max_depth=params_xgb['max_depth'],
booster=params_xgb['booster'])
        xgb.fit(X_train_Divided_resampled_scaled, y_train_Divided)
        y_pred = xgb.predict(X_val_scaled)


        acc_xgb = accuracy_score(y_val, y_pred)
        acc_val_xgb.append(acc_xgb)


        recall_xgb = recall_score(y_val, y_pred)
        recall_val_xgb.append(recall_xgb)


        precision_xgb = precision_score(y_val, y_pred)
        precision_val_xgb.append(precision_xgb)


        f1_xgb = f1_score(y_val, y_pred)
        f1_val_xgb.append(f1_xgb)


        par_xgb.append(params_xgb)


      except Exception as e:
```

```
            print(f"Erro ao treinar XGBoost com params {params_lr}:
{e}")
            continue


    print('XGBoost_ACC',
par_xgb[acc_val_xgb.index(max(acc_val_xgb))])
    print('XGBoost_RECALL',
par_xgb[recall_val_xgb.index(max(recall_val_xgb))])
    print('XGBoost_PRECISION',
par_xgb[precision_val_xgb.index(max(precision_val_xgb))])
    print('XGBoost_F1', par_xgb[f1_val_xgb.index(max(f1_val_xgb))])




    best_lr = LogisticRegression(C =
par_lr[recall_val_lr.index(max(recall_val_lr))]['C'],
                                 max_iter =
par_lr[recall_val_lr.index(max(recall_val_lr))]['max_iter'],
                                 solver =
par_lr[recall_val_lr.index(max(recall_val_lr))]['solver'],
                                 penalty =
par_lr[recall_val_lr.index(max(recall_val_lr))]['penalty'])


    best_knn = KNeighborsClassifier(n_neighbors =
par_knn[recall_val_knn.index(max(recall_val_knn))]['n_neighbors'],
                                    metric =
par_knn[recall_val_knn.index(max(recall_val_knn))]['metric'])


    best_random = RandomForestClassifier(n_estimators =
par_random[recall_val_random.index(max(recall_val_random))]['n_esti
mators'],
                                         criterion =
par_random[recall_val_random.index(max(recall_val_random))]['criter
ion'],
                                         max_depth =
par_random[recall_val_random.index(max(recall_val_random))]['max_de
pth'],
```

```
                                        min_samples_split =
par_random[recall_val_random.index(max(recall_val_random))]['min_sa
mples_split'])


    best_tree =
DecisionTreeClassifier(criterion=par_tree[recall_val_tree.index(max
(recall_val_tree))]['criterion'],

splitter=par_tree[recall_val_tree.index(max(recall_val_tree))]['spl
itter'],

max_depth=par_tree[recall_val_tree.index(max(recall_val_tree))]['ma
x_depth'],

min_samples_split=par_tree[recall_val_tree.index(max(recall_val_tre
e))]['min_samples_split'])


    best_xgb =
XGBClassifier(max_depth=par_xgb[recall_val_xgb.index(max(recall_val
_xgb))]['max_depth'],

booster=par_xgb[recall_val_xgb.index(max(recall_val_xgb))]['booster
'])




    best_recall_lr = max(recall_val_lr)

    best_recall_knn = max(recall_val_knn)

    best_recall_random = max(recall_val_random)

    best_recall_tree = max(recall_val_tree)

    best_recall_xgb = max(recall_val_xgb)




    models_recall = {'Logistic Regression': (best_lr,
best_recall_lr),
                     'KNN': (best_knn, best_recall_knn),
                     'Random Forest': (best_random,
best_recall_random),
```

```
                    'Decision Tree': (best_tree,
best_recall_tree),
                    'XGBoost': (best_xgb, best_recall_xgb)
                   }


    # modelo com o melhor recall
    best_model_name = max(models_recall, key=lambda x:
models_recall[x][1])#recebe nome do modelo com maior recall
    best_model = models_recall[best_model_name][0]#recebe os
melhores hiperparametros desse modelo




    # nome do melhor modelo e seus hiperparâmetros
    print(f'Melhor modelo: {best_model_name}')
    print(f'Hiperparâmetros: {best_model.get_params()}')


    best_model.fit(X_train_Divided_resampled_scaled,
y_train_Divided)
    y_pred_final = best_model.predict(X_test_scaled)


    conf_matrix = confusion_matrix(y_test, y_pred_final)
    ConfusionMatrixDisplay(conf_matrix).plot(cmap='Blues')
    plt.title('Matriz de Confusão')
    plt.show()


    acc_final = accuracy_score(y_test, y_pred_final)
    print('Acuracia: ',acc_final)
    metrics['accuracy'].append(acc_final)


    recall_final = recall_score(y_test, y_pred_final)
    print('Recall: ',recall_final)
    metrics['recall'].append(recall_final)
```

```
    precision_final = precision_score(y_test, y_pred_final)
    print('Precisão: ',precision_final)
    metrics['precision'].append(precision_final)

    f1_final = f1_score(y_test, y_pred_final)
    print('F1_score: ',f1_final)
    metrics['f1'].append(f1_final)

    melhores_modelos_por_rodada.append({
        'model_name': best_model_name,
        'model': best_model,
        'recall': models_recall[best_model_name][1],
        'accuracy': acc_final,
        'precision': precision_final,
        'f1': f1_final
    })

    for model_name, (model, recall) in models_recall.items():
        metricas_por_modelo_recall[model_name].append(recall)
```

Fonte: Dados da pesquisa (2025)

Quadro 2 - Parâmetros de entrada do modelo

```
modelo_lr = {
    'C': [0.01, 0.1, 1, 10],
    'max_iter': [1000],
    'solver': ['liblinear', 'lbfgs'],
    'penalty': ['l1', 'l2']
}
modelo_knn = {
    'n_neighbors': [3, 5, 7, 9, 11],
        'metric':  ['euclidean',  'manhattan',  'chebyshev',
'minkowski']
```

```
}
modelo_random = {
    'n_estimators': [100, 200, 300],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 3, 5, 7, 10],
    'min_samples_split': [2, 5, 10]
}

modelo_arvore = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 3, 5, 7, 10],
    'min_samples_split': [2, 5, 10]
}

modelo_xgb = {
    'max_depth': [3, 5, 7, 10],
    'booster': ['gbtree', 'gblinear', 'dart'],
    'n_estimators': [100, 200],
    'learning_rate': [0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
```

Fonte: Dados da pesquisa (2025)

8 CONCLUSÃO

A ferramenta computacional desenvolvida, além de apresentar uma inovação metodológica para a UFC *campus* de Crateús, possui impacto no âmbito educacional, ao contribuir para a melhoria do processo de acompanhamento do desempenho acadêmico dos estudantes, tornando-se uma proposta que inova no campo da gestão acadêmica desta IFES.

Considerando que a ferramenta, em fase de implementação, precisa de melhorias e ser avaliado pelos usuários a quem se destina a princípio, registra-se as ações futuras, quais

sejam: aplicação em outras unidades acadêmicas, integração com o SIGAA, utilização de outros algoritmos de Inteligência Artificial e, por fim, o uso de aprendizagem profunda.

REFERÊNCIAS

NUNES, Hélder Antero Amaral. **Mineração de dados socioeconômicos e educacionais de discentes para predição de evasão e retenção escolar**. 2023. Dissertação (Mestrado Profissional em Tecnologia Educacional) - Instituto UFC Virtual, Universidade Federal do Ceará, Fortaleza, 2023. Disponível em: https://repositorio.ufc.br/handle/riufc/75061. Acesso em: 15 dez. 2025.