Inteligência Artificial: Estudo para Predição de Doenças Cardíacas

Análise e Aplicação no Conjunto de Dados 'Heart Disease Prediction' com Random Forest

Alexandre Augusto
PUC Minas
Belo Horizonte, MG

André Mendes
PUC Minas
Belo Horizonte, MG

Arthur Martinho
PUC Minas
Belo Horizonte, MG

Caio Gomes
PUC Minas
Belo Horizonte, MG

Daniel Salgado
PUC Minas
Belo Horizonte, MG

Rafael Maluf
PUC Minas
Belo Horizonte, MG

ABSTRACT

This study investigates predictive modeling of heart disease and aims to improve early detection of heart disease by analyzing attributes of a database, such as age, sex, blood pressure, and cholesterol levels. The article demonstrates preprocessing techniques that were applied to prepare the data, such as Random Forest, Logistic Regression, and SVM, used to classify classes, predicting whether the patient has cancer or not. The predictive models use different bases to perform the analyses in a broad way, bringing good results to generate advances in medicine.

KEYWORDS

Heart Disease, Machine Learning, Data Preprocessing, Random Forest

1 INTRODUÇÃO

Os estudos e utilização da inteligência artificial estão avançando de forma que é permitido realizar várias pesquisas no campo da medicina. A temática de doenças cardíacas está presente neste campo, e é um dos maiores desafios da medicina atual, tendo em vista a complexidade e cuidado envolvendo o tratamento do coração. Sendo assim, o artigo a seguir contempla um estudo realizado para predição de doenças cardíacas, buscando aprimorar os conhecimentos voltados para essa temática da medicina.

O artigo se concentra na utilização do conjunto de dados "*Heart Disease Prediction Dataset*", disponível na plataforma *Kaggle*, para a construção de um modelo preditivo capaz de identificar a presença de doenças cardíacas em pacientes. A realização de uma análise detalhada desses dados melhora a compreensão dos fatores de risco associados às doenças cardíacas, e demonstra a eficácia de técnicas de pré-processamento, incluindo a limpeza de dados, a remoção de *outliers* e o balanceamento de classes, que são tópicos abordados ao decorrer do artigo.

2 ESTUDO DA BASE DE DADOS

A base de dados utilizada para realização do projeto foi a "*Heart Disease Predicition Dataset*", obtida através do site de base de dados "*Kaggle*" pelo senhor Krish Ujeniya. O conjunto de dados de predição de doenças cardíacas contém informações médicas utilizadas para prever a presença de doenças cardíacas em pacientes. Com um total de 303 registros e 14 atributos, esta base de dados inclui variáveis relevantes como idade, sexo, tipo de dor no peito, pressão arterial em repouso, níveis de colesterol, glicemia em jejum, resultados eletrocardiográficos em repouso, frequência cardíaca máxima alcançada, angina induzida por exercício, e depressão do segmento ST induzida por exercício em relação ao repouso. Os principais atributos contidos no conjunto de dados são:

- Idade: A idade dos pacientes varia de 29 a 77 anos.
- Sexo: A variável de sexo é binária, com 1 representando masculino e 0 feminino.
- Tipo de Dor no Peito (cp): Os tipos de dor no peito variam de 0 a 3.
- Pressão Arterial em Repouso (trestbps): A pressão arterial dos pacientes varia de 94 a 200 mm.
- Colesterol: Os níveis de colesterol variam entre 126 a 564 mg/dl.
- Glicemia em Jejum (fbs): A glicemia em jejum é superior a 120 mg/dl em 15% dos casos.
- Resultados Eletrocardiográficos (restecg): Esta variável varia de 0 a 2.
- Frequência Cardíaca Máxima (thalach): Os pacientes alcançam uma frequência cardíaca máxima que varia de 71 a 202 bpm.
- Angina Induzida por Exercício (exang): Aproximadamente 32,7% dos pacientes relataram angina induzida por exercício.
- Depressão ST (oldpeak): A depressão ST induzida por exercício varia de 0 a 6,2.
- Inclinação do Segmento ST (slope): A inclinação do segmento ST varia de 0 a 2.
- Número de Vessels (ca): O número de vasos sanguíneos coloridos varia de 0 a 4.
- Thalassemia (thal): Variando de 0 a 3.
- Alvo (*target*): Esta variável indica a presença (1) ou ausência (0) de doenças cardíacas, com uma média de 54,5% de pacientes apresentando a doença.

3 ETAPAS DE PRÉ-PROCESSAMENTO

O pré-processamento dos dados foi dividido em 4 seções para que todos os dados fossem treinados e testados de forma que o algoritmo estivesse apto para realizar os cálculos da melhor forma possível.

(1) **Limpeza de Dados**
Na primeira etapa do processo, é realizada a limpeza dos dados, fundamental para garantir a qualidade e a integridade das informações antes de qualquer análise. O conjunto de dados é carregado utilizando a biblioteca `pandas`, que facilita o uso de estruturas de dados. O método `dropna()` é utilizado para remover qualquer linha que contenha dados ausentes, evitando grandes distorções nos resultados das análises e dos modelos preditivos. Em seguida, `drop_duplicates()` elimina valores duplicados no conjunto de dados, garantindo que cada registro seja único. A limpeza de dados foi escolhida

como a primeira etapa do processo, pois dados imprecisos ou redundantes podem levar a conclusões erradas, reduzindo a eficácia dos modelos de *machine learning*.

```
df = pd.read_csv('/content/heart-disease.csv')

rows_with_na = df[df.isnull().any(axis=1)]

df_cleaned = df.dropna()

duplicated_rows = df_cleaned[df_cleaned.duplicated
    ()]

df_cleaned = df_cleaned.drop_duplicates()
```

Listing 1: Código Limpeza de dados

A etapa de limpeza de dados é essencial em qualquer pipeline de análise para garantir a qualidade e integridade do conjunto de dados antes da aplicação de modelos preditivos. Neste trabalho, a limpeza consistiu na remoção de valores ausentes e duplicatas utilizando as funções `dropna()` e `drop_duplicates()`. O resultado foi a exclusão de apenas uma linha duplicada, reduzindo o dataset de 303 para 302 registros, o que evidencia a consistência inicial da base de dados. Apesar do impacto mínimo, esta etapa é fundamental para evitar distorções nos resultados e garantir maior confiabilidade nas análises posteriores.

(2) **Identificação e Remoção de Outliers usando IQR**
Após a limpeza de dados abordamos a identificação e remoção de *outliers* (valores extremos que podem influenciar indevidamente a análise) usando o método do Intervalo Interquartil (IQR). O IQR é uma técnica estatística que ajuda a determinar a dispersão dos dados.
A função `remove_outliers_iqr()` foi definida para calcular o primeiro quartil (Q1) e o terceiro quartil (Q3) de uma coluna específica, a partir dos quais calculamos o intervalo interquartil (IQR). Os limites inferior e superior são então definidos como $Q1 - 1.5 \times IQR$ e $Q3 + 1.5 \times IQR$, respectivamente. A função remove os registros fora desses limites, o que ajuda a refinar o conjunto de dados. As colunas selecionadas para análise incluem `age` (idade), `trestbps` (pressão arterial em repouso), `chol` (colesterol), `thalach` (frequência cardíaca máxima) e `oldpeak` (depressão ST induzida por exercício). Após a remoção de *outliers*, o conjunto de dados tratado é salvo em um novo arquivo CSV.

```
def remove_outliers_iqr(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    data = data[(data[column] >= lower_bound) & (
     data[column] <= upper_bound)]
    return data

columns_to_check = ['age', 'trestbps', 'chol', '
    thalach', 'oldpeak']

for column in columns_to_check:
    df = remove_outliers_iqr(df, column)

df.to_csv('/content/heart-disease-treated.csv',
    index=False)
```

Listing 2: Código Identificação e remoção dos outliers

Na etapa 1 é realizada a limpeza de dados e identificação e remoção de *outliers* usando IQR. As duas são realizadas na primeira etapa, pois são feitas na mesma base inicial. Após a limpeza de dados, é possível verificar que não existem dados ausentes, e existe apenas um dado duplicado, presente na linha 164 da base de dados, fazendo com que a base, de antes 303 instâncias, ficasse com 302 instâncias. Realizada a limpeza dos dados, nós executamos o código para identificação e remoção dos *outliers* a partir do método IQR. Com a base passando por este processo, garantimos uma melhoria na precisão e uma facilidade maior para visualização de dados. Ajustando os limites possíveis da tabela a partir dos valores dos quadrantes de Q1 e Q3, é possível verificar a redução na quantidade de instâncias, que passa de 302 instâncias para 284 instâncias, sofrendo uma redução de 18 instâncias. Com a finalização de ambos os processos, nós criamos uma nova base de dados que recebe essas novas alterações, e possui o nome de *heart-disease-treated*. Isto foi feito para que não seja necessário realizar o tratamento de dados em cada etapa, e a visualização dos dados alterados seja vista mais facilmente.

(3) **Normalização e Padronização**
A segunda e a terceira etapa contam com métodos de transformação dos dados presentes na nova base de dados, fornecida após a etapa 1. A segunda etapa, de normalização, é realizada para geração da base de dados que será utilizada no algoritmo de *RandomForest*. O tratamento pela normalização ajusta os dados para que fiquem em um intervalo fixo entre 0 e 1, reduzindo os valores para uma escala uniforme. No código abaixo, é possível verificar que recebemos a base de dados tratadas na etapa 1 e realizamos a normalização sem incluir a variável *target*. Ao fim da normalização, a variável *target* é adicionada novamente e os dados são salvos em uma nova base de dados, chamada de *normalized-heart-disease*, usada nas próximas etapas.

```
data = pd.read_csv('/content/heart-disease-treated.
    csv')

features = data.drop(columns=['target'])

scaler = MinMaxScaler()
normalized_features = pd.DataFrame(scaler.
    fit_transform(features), columns=features.
    columns)

normalized_data = pd.concat([normalized_features,
    data['target']], axis=1)

normalized_data.to_csv('normalized-heart-disease.
    csv', index=False)
```

Listing 3: Código Normalização

A terceira etapa, de padronização, é realizada para geração da base de dados que será utilizada nos algoritmos de Regressão Logística e *Support Vector Machine*. O tratamento

pela padronização ajusta os dados para que sejam centralizados em um média de 0 e um desvio-padrão de 1, facilitando a utilização de algoritmos. No código abaixo, é possível verificar que recebemos a base de dados tratadas na etapa 1 e realizamos a padronização sem incluir a variável *target*. Ao fim da padronização, a variável *target* é adicionada novamente e os dados são salvos em uma nova base de dados, chamada de *standardized-heart-disease*.

```
data = pd.read_csv('/content/heart-disease-treated.csv')

features = data.drop(columns=['target'])

scaler = StandardScaler()
standardized_features = pd.DataFrame(scaler.fit_transform(features), columns=features.columns)

standardized_data = pd.concat([standardized_features, data['target']], axis=1)

standardized_data.to_csv('standardized_heart_disease.csv', index=False)

```

Listing 4: Código Padronização

(4) **Balanceamento de Dados com SMOTE**
A quarta etapa concentra-se no balanceamento do conjunto de dados utilizando a técnica de *oversampling* conhecida como SMOTE (*Synthetic Minority Over-sampling Technique*). Essa técnica é utilizada para lidar com a distribuição desbalanceada das classes no conjunto de dados, que pode comprometer o desempenho dos modelos de predição. Os conjuntos de dados, *normalizado* e o *padronizado* , são divididos em variáveis independentes (X ou Xp = df . drop (' target ', axis =1)) e a variável dependente (y ou yp = df [' target ']), que indica a presença ou ausência de doenças cardíacas. Os conjuntos de dados são divididos em subconjuntos de treino e teste usando a função `train_test_split()`. Sendo 1/5 o conjunto de teste e 4/5 o conjunto de train, o SMOTE é então aplicado ao conjunto de treino para gerar exemplos sintéticos da classe minoritária, equilibrando assim as classes. Após a aplicação do SMOTE, é possível perceber as mudanças realizadas. Recebemos 128 instâncias na classe 1 e 99 instâncias na classe 0. Isso significa que das 227 instâncias de teste, 128 delas tem doença cardíaca, e 99 não tem. Com o SMOTE sendo aplicado, são geradas instâncias sintéticas para balancear a quantidade de amostras, fazendo com que tanto a classe de possuir doença cardíaca quanto a classe de não possuir, tenham 128 instâncias, totalizando 256 instâncias finais.
Após o término do SMOTE, a nova distribuição das classes é verificada e os conjuntos de dados balanceados são salvos em novos arquivos CSV, denominados de *normalizedbalanced_heart_disease* e *standardizedbalanced_heart_disease*.

```
df = pd.read_csv('/content/normalized_heart_disease.csv')
dfp = pd.read_csv('/content/standardized-heart-disease.csv')

X = df.drop('target', axis=1)
y = df['target']

Xp = dfp.drop('target', axis=1)
yp = dfp['target']

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

Xp_train, Xp_test, yp_train, yp_test = train_test_split(Xp, yp, test_size=0.2, random_state=42)

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

Xp_resampled, yp_resampled = smote.fit_resample(Xp_train, yp_train)

balanced_df = pd.DataFrame(data=X_resampled, columns=X.columns)
balanced_df['target'] = y_resampled

balanced_dfp = pd.DataFrame(data=Xp_resampled, columns=Xp.columns)
balanced_dfp['target'] = yp_resampled

balanced_df.to_csv('/content/normalizedbalanced_heart_disease.csv', index=False)
balanced_dfp.to_csv('/content/standardizedbalanced_heart_disease.csv', index=False)

```

Listing 5: Código balanceamento por SMOTE

4 ALGORITMOS UTILIZADOS

A quinta e sexta etapa consistem nos algoritmos utilizados para análise dos dados treinados. Na quinta etapa é usado o algoritmo de Regressão Logística e *Support Vector Machine*(SVM), que utilização da base de dados de padronização para executarem as funções.

O algoritmo de Regressão Logística é um modelo de classificação estatístico usado para tarefas de classificação binária através de uma análise de dados, isso permite que seja possível analisar os dados estimando a probabilidade associada à ocorrência de determinado evento. Optamos por usar este algoritmo como teste devido à sua capacidade de lidar com problemas de classificação binária através da variável *target* de forma eficiente e intuitiva, além de ser útil para problemas em que é necessário interpretar probabilidades. Na base de dados usada, é de grande ajuda para a análise da presença ou ausência de uma doença, que possibilita não apenas a classificação, mas também uma estimativa da probabilidade associada a cada caso, o que pode fornecer informações valiosas para a tomada de decisões médicas e estratégias de intervenção.

O algoritmo de *Support Vector Machine* é um algoritmo supervisionado de classificação que tenta encontrar o hiperplano ótimo para separar classes no espaço de entrada. O algoritmo procura uma linha de separação entre essas classes analisando a cada dois pontos, um de cada grupo, mais próximos da outra classe. Quando a reta for descoberta, o programa consegue predizer a qual classe

pertence o novo dado. Neste caso, existem duas classes, já referenciadas anteriormente como a classse de possuir ou não possuir doença cardíaca, sendo assim, o algoritmo verificará o resultado da variável *target* de cada instância.

```
data = pd.read_csv('/content/balanced_heart_disease.csv')

X = data.drop(columns=['target'])
y = data['target']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42)

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)

svm = SVC(probability=True, random_state=42)
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)

def evaluate_model(y_true, y_pred):
    return {
        'Accuracy': accuracy_score(y_true, y_pred),
        'Precision': precision_score(y_true, y_pred),
        'Recall': recall_score(y_true, y_pred),
        'F1-Score': f1_score(y_true, y_pred),
        'AUC': roc_auc_score(y_true, y_pred)
    }

log_reg_metrics = evaluate_model(y_test, y_pred_log_reg)

svm_metrics = evaluate_model(y_test, y_pred_svm)

metrics_table = pd.DataFrame({
    'Model': ['Logistic Regression', 'SVM'],
    'Accuracy': [log_reg_metrics['Accuracy'], svm_metrics
    ['Accuracy']],
    'Precision': [log_reg_metrics['Precision'],
    svm_metrics['Precision']],
    'Recall': [log_reg_metrics['Recall'], svm_metrics['
    Recall']],
    'F1-Score': [log_reg_metrics['F1-Score'], svm_metrics
    ['F1-Score']],
    'AUC':[log_reg_metrics['AUC'], svm_metrics['AUC']]
})
```

Listing 6: Código de Regressão Logística e SVM

O Treinamento do Modelo *Random Forest* é a sexta e última seção do algoritmo completo e foca no treinamento de um modelo de classificação utilizando o algoritmo *Random Forest*, por causa da sua robustez e capacidade de lidar com conjuntos de dados complexos. O arquivo CSV balanceado é carregado, e as variáveis independentes são separadas da variável dependente. O conjunto de dados é então dividido em subconjuntos de treino e teste. Para melhorar o desempenho do modelo, normalizamos os dados utilizando a classe *StandardScaler*, que transforma as características para que tenham uma média de 0 e um desvio padrão de 1, o que é especialmente importante para algoritmos baseados em distância. O modelo *Random Forest* é então instanciado e treinado com o conjunto de dados de treino. Após o treinamento, previsões são realizadas no conjunto de teste e o desempenho do modelo é avaliado usando métricas como acurácia, matriz de confusão e relatório de classificação. Esses resultados fornecem uma visão clara da eficácia do modelo em prever a presença de doenças cardíacas.

```
X = data.drop(columns=['target'])
y = data['target']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42)

rf = RandomForestClassifier(n_estimators=100,
    random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

def evaluate_model(y_true, y_pred):
    return {
        'Accuracy': accuracy_score(y_true, y_pred),
        'Precision': precision_score(y_true, y_pred),
        'Recall': recall_score(y_true, y_pred),
        'F1-Score': f1_score(y_true, y_pred),
        'AUC': roc_auc_score(y_true, y_pred)
    }

rf_metrics = evaluate_model(y_test, y_pred_rf)

metrics_table = pd.DataFrame({
    'Model': ['Random Forest'],
    'Accuracy': [rf_metrics['Accuracy']],
    'Precision': [rf_metrics['Precision']],
    'Recall': [rf_metrics['Recall']],
    'F1-Score': [rf_metrics['F1-Score']],
    'AUC': [rf_metrics['AUC']]
})
```

Listing 7: Código RandomForest

5 RESULTADOS DA ANÁLISE COM ALGORITMOS DE APRENDIZADO DE MÁQUINA

	Model	Accuracy	Precision	Recall	F1-Score	AUC
0	Random Forest	0.818182	0.782609	0.900	0.837209	0.814865
1	SVM	0.883117	0.829787	0.975	0.896552	0.879392
2	Logistic Regression	0.818182	0.795455	0.875	0.833333	0.815878

Figure 1: Resultados banco normalizado

	Model	Accuracy	Precision	Recall	F1-Score	AUC
0	Random Forest	0.831169	0.787234	0.925	0.850575	0.827365
1	SVM	0.857143	0.808511	0.950	0.873563	0.853378
2	Logistic Regression	0.844156	0.791667	0.950	0.863636	0.839865

Figure 2: Resultados banco padronizado

5.1 Introdução

O objetivo do experimento foi avaliar o desempenho de três algoritmos de aprendizado de máquina — Random Forest, SVM e Regressão Logística — na predição de predisposição a doenças cardíacas. Foram realizadas análises em dois cenários diferentes: uma base de dados padronizada (com média 0 e desvio padrão 1) e uma base normalizada (escalonamento dos valores entre 0 e 1). As métricas utilizadas para avaliação foram Acurácia, Precisião, *Recall*, *F1-Score* e AUC (Área Sob a Curva).

5.2 Resultados com a Base Padronizada

Na análise com a base padronizada, os resultados obtidos foram os seguintes:

(1) Random Forest: Obteve uma acurácia de 0.831169 e um *F1-Score* de 0.850575, com um *recall* relativamente alto (0.925), mas menor precisão (0.787234).
(2) SVM: Apresentou o melhor desempenho geral, com acurácia de 0.857143, precisão de 0.808511, *recall* de 0.950 e *F1-Score* de 0.873563.
(3) Regressão Logística: Teve desempenho intermediário, com acurácia de 0.844156, precisão de 0.791667, *recall* de 0.950 e F1-Score de 0.863636.

5.3 Resultados com a Base Normalizada

Na análise com a base normalizada, os resultados foram:

(1) *Random Forest*: Mostrou uma leve redução de desempenho, com acurácia de 0.818182, *recall* de 0.900, mas uma precisão similar de 0.782609 e *F1-Score* de 0.837209.
(2) SVM: Continuou como o melhor algoritmo, apresentando acurácia de 0.883117, precisão de 0.829787, *recall* de 0.975 e *F1-Score* de 0.896552.
(3) Regressão Logística: Apresentou acurácia e *F1-Score* de 0.818182, com precisão de 0.795455 e *recall* de 0.875.

5.4 Discussão

O algoritmo SVM apresentou o melhor desempenho em ambos os cenários, destacando-se em termos de *F1-Score* e AUC, o que indica sua eficácia em balancear a precisão e o *recall*. A Regressão Logística apresentou desempenho consistente nos dois cenários, mas ficou atrás do SVM em métricas importantes como acurácia e F1-Score. O *Random Forest*, apesar de ter um recall alto, mostrou uma precisão mais baixa, o que pode indicar uma tendência para falsos positivos. A comparação entre os cenários revela que a normalização beneficiou o SVM e teve um impacto ligeiramente negativo no desempenho do Random Forest. Esse comportamento pode estar relacionado com a sensibilidade dos algoritmos ao escalonamento dos dados, sendo que o SVM se beneficia mais da normalização.

5.5 Conclusão

Os resultados mostram que a escolha do algoritmo e do método de pré-processamento impacta significativamente o desempenho. Para este conjunto de dados, o SVM se mostrou a melhor opção, principalmente em cenários em que recall é uma prioridade, como na predição de doenças cardíacas.

6 UTILIZAÇÃO DO GPT

Foi usado o GPT através do ChatGPT, tendo em vista que é o chatbot mais conhecido pelo grupo. O mesmo foi utilizado para revisão dos textos que nós escrevemos e otimização dos códigos utilizados.

Para a revisão dos textos, utilizamos ele para verificação de possíveis erros relacionados à estrutura do texto, verificando a coerência do mesmo baseado no que escrevemos, mantendo a segurança de um texto limpo e bem interpretável.

Na otimização dos códigos, verificamos se existia algum erro em relação às classes utilizadas, que poderiam acarretar em problemas na verificação dos resultados, que mesmo com mudanças nas variáveis de treino e teste, não seriam perceptíveis para verificação do erro.

7 CÓDIGO DESENVOLVIDO

O código foi todo desenvolvido em Python através da plataforma *Google Colab*, recomendada em sala de aula, e o mesmo encontra-se citado nas referências.

REFERÊNCIAS

[1] [n. d.]. O que são SVMs? IBM. https://www.ibm.com/br-pt/topics/support-vector-machine#:~:text=O%20algoritmo%20SVM%20%C3%A9%20amplamente,para%20permitir%20a%20separa%C3%A7%C3%A3o%20linear. Publicado em: 27 de Dezembro de 2023.
[2] Arthur Martinho Caio Gomes Daniel Salgado Rafael Maluf Alexandre Augusto, André Mendes. 2024. Inteligência Artificial: Estudo para Predição de Doenças Cardíacas. Google Colab. https://colab.research.google.com/drive/18YnBIOUPVSlhjSLq8W1UeuekHDtgqm7x?usp=sharing Publicado em: 29 de Setembro de 2024.
[3] AWS. [n. d.]. O que é regressão logística? AWS. https://aws.amazon.com/pt/what-is/logistic-regression/#:~:text=A%20regress%C3%A3o%20log%C3%ADstica%20%C3%A9%20uma,resultados%2C%20como%20sim%20ou%20n%C3%A3o.
[4] Roman Tkachenko Nataliya Shakhovska Bohdan Ilchyshyn Krishna Kant Ivan Izonin, ORCID. [n. d.]. A Two-Step Data Normalization Approach for Improving Classification Accuracy in the Medical Diagnosis Domain. Lviv Polytechnic National University. https://www.mdpi.com/2227-7390/10/11/1942 Publicado em: 6 de Junho de 2022.
[5] ZEBA TUSNIA TOWSHI MANAL A. OTHMAN MD ABDUS SAMAD KHALED MAHMUD SUJON, ROHAYANTI BINTI HASSAN and KWONHUE CHOI. [n. d.]. When to Use Standardization and Normalization: Empirical Evidence From Machine Learning Models and XAI. IEEE Access. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10681438 Publicado em: 17 de Setembro de 2024.
[6] Cristiane Neri Nobre. 2024. Ensembles Learning ou Aprendizagem em Conjunto. Slide presentes nos arquivos do Canvas. Publicado em: 29 de Setembro de 2024.
[7] Cristiane Neri Nobre. 2024. ETAPAS DE PRÉ-PROCESSAMENTO BALANCEAMENTO DA BASE DE DADOS. Slide presentes nos arquivos do Canvas. Publicado em: 29 de Setembro de 2024.
[8] Cristiane Neri Nobre. 2024. ETAPAS DE PRÉ-PROCESSAMENTO DADOS INCONSISTENTES E REDUNDANTES. Slide presentes nos arquivos do Canvas. Publicado em: 29 de Setembtro de 2024.
[9] Cristiane Neri Nobre. 2024. ETAPAS DE PRÉ-PROCESSAMENTO TRATAMENTO DE DADOS AUSENTES. Slide presentes nos arquivos do Canvas. Publicado em: 29 de Setembro de 2024.
[10] Cristiane Neri Nobre. 2024. TRANSFORMAÇÃO DE DADOS REDUÇÃO DE DIMENSIONALIDADE. Slide presentes nos arquivos do Canvas. Publicado em: 29 de Setembro de 2024.
[11] Vinícius de Paula Pilan. [n. d.]. TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA DIAGNÓSTICODE ACIDENTE VASCULAR CEREBRAL ATRAVÉS DE IMAGENS E DADOS TEXTUAIS SOBRE POSSÍVEIS VÍTIMAS. Faculdade de Ciências, Campus Bauru.. https://repositorio.unesp.br/server/api/core/bitstreams/82212b3a-8485-4a88-835b6968b69de544/content Publicado em: Janeiro de 2023.
[12] David Sontag. [n. d.]. Support vector machines (SVMs) Lecture 2. New York University. https://people.csail.mit.edu/dsontag/courses/ml14/slides/lecture2.pdf
[13] USP. [n. d.]. REGRESSÃO LOGÍSTICA. USP. https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf

[2] [6] [9] [10] [7] [8] [11] [4] [5] [1] [12] [3] [13]