Machine Learning Aplicado ao Estudo da Diabetes

Felipe Campolina, Gabriel Vargas, Gabriela Colem, Nilson Deon e Saulo de Moura
ICEI - PUC Minas, Belo Horizonte, Minas Gerais, Brasil
1375450@sga.pucminas.br,1402785@sga.pucminas.br,1374375@sga.pucminas.br,
1369636@sga.pucminas.br,1400518@sga.pucminas.br

RESUMO

Este artigo se concentra na aplicação de modelos de *Machine Learning* para diagnosticar e analisar o aumento da incidência de diabetes nos Estados Unidos da América (EUA). Diante da complexidade do diagnóstico e devido à variedade de sintomas e de origens da doença, foram empregados os modelos: *Decision Tree*, *Naive Bayes*, *Random Forest*, *Bagging Classifier* e *Neural Network*. Uma ênfase especial foi colocada no pré-processamento da base de dados, visando aprimorar a qualidade das previsões. O objetivo central é contribuir para a predição de novos casos em escala global, fornecendo suporte tanto aos profissionais de saúde quanto aos órgãos responsáveis pela saúde pública. A análise minuciosa desses modelos não apenas visa determinar se um paciente tem diabetes, mas também identificar os fatores mais significativos relacionados ao surgimento da doença. Essa abordagem detalhada permitirá intervenções mais eficientes e informadas, enfrentando, de maneira mais assertiva, o desafio crescente da incidência de diabetes.

KEYWORDS

Decision Tree, Naïve Bayes, *Random Forest*, *Neural Network*, *Bagging Classifier*, *Machine Learning*, *Data Science*, *Public Health*, *Diabetes*

ACM Reference Format:
Felipe Campolina, Gabriel Vargas, Gabriela Colem, Nilson Deon e Saulo de Moura. 2023. *Machine Learning* Aplicado ao Estudo da Diabetes. In *TP - Inteligência Artificial, November 24, 2023, Belo Horizonte, MG.* , 8 pages.

1 INTRODUÇÃO

O corpo humano é responsável pela produção da insulina destinada ao próprio organismo. Isso ocorre por meio do pâncreas, glândula localizado atrás do estômago, que, após uma elevação da taxa de glicemia, utiliza as células beta para produzir a insulina [6]. Este hormônio é o mecanismo responsável por facilitar o aproveitamento da glicose pelas outras células do organismo, seja para atividades físicas, como correr e andar, seja para ser armazenada, em forma de lipídios, como fonte de reserva energética [7].

Entretanto, essa funcionalidade do corpo humano pode, às vezes, apresentar alguns problemas, ocasionando falhas no funcionamento do organismo. Nesse sentido, devido à inexistência ou à baixa produção de insulina, a saúde da pessoa pode ser grandemente comprometida. Assim, com problemas na secreção desse hormônio, o indivíduo sofre com o não aproveitamento da glicose no organismo, fazendo com que haja um forte acúmulo desse açúcar no sangue [7]. Tal cenário é chamado de hiperglicemia, que é um fator de grande relevância para o diagnóstico como *Diabetes mellitus*. Sob essa ótica, o quadro de Diabetes pode ser classificado em decorrência da origem do problema, como Tipo 1, Tipo 2 e Gestacional.

Nesse contexto, a Diabetes Tipo 1 tem origem autoimune, isto é, o próprio organismo do indivíduo destrói as células pancreáticas, afetando a produção da insulina. Isso gera uma insuficiência na produção do hormônio, o que faz com que o açúcar se acumule no sangue [7]. Em contrapartida, ao contrário do Tipo 1, a Diabetes Tipo 2 caracteriza-se pela existência da produção de insulina pelas células beta, mas, devido à pouca atividade física, obesidade e outros problemas secundários, tal produção torna-se insuficiente para as demandas do corpo, caracterizando o quadro de diabetes [6].

Além disso, no caso da Diabetes Gestacional, a mãe tende a aumentar a produção de diferentes hormônios por causa da gravidez para o desenvolvimento do feto no útero. Dentre eles, existe o lactogênio placentário humano (HPL), que é responsável pelo crescimento do bebê, porém tem a reação natural de diminuir a produção de insulina pelo seu organismo. Dessa forma, o pâncreas acaba ficando sobrecarregado para suprir a ausência do hormônio; todavia, não são todas as mulheres que conseguem compensar a perda pela gravidez, fazendo com que o nível de glicose no sangue aumente, de forma a causar diabetes [5].

Tendo isso em vista, cria-se a extrema necessidade de se analisar aspectos e características do corpo de um indivíduo para auxiliar em possíveis diagnósticos de diabetes. Para tanto, é mister que um estudo na área de *Machine Learning*

TP - Inteligência Artificial, November 24, 2023, Belo Horizonte, MG

seja realizado com o intuito de prever e de diagnosticar possíveis pacientes de diabetes. Dessa forma, foi escolhida uma base de dados [13], publicada pelo *National Institute of Diabetes and Digestive and Kidney Diseases*, sobre a enfermidade nos EUA, na qual alguns atributos foram levados em consideração para classificar uma pessoa como diabética ou não.

2 DESCRIÇÃO DA BASE DE DADOS

Na Tabela 1, estão descritos os 9 atributos que fazem parte da base de dados selecionada, a qual possui 768 instâncias:

Tabela 1: Descrição dos atributos da base de dados

Atributo	Descrição	Valores
Pregnancies	Número de gestações do paciente	Numérico Máx = 13 Min = 0
Glucose	Glicose no sangue do paciente [mg/dL]	Numérico Máx = 197 Min = 44
Blood Pressure	Pressão sanguínea do paciente [mm Hg]	Numérico Máx = 104 Min = 44
Skin Thickness	Espessura da pele do paciente [mm]	Numérico Máx = 42 Min = 15
Insulin	Nível de insulina do paciente [$\mu U/mL$]	Numérico Máx = 245 Min = 16
BMI	Índice de Massa Corporal do paciente [kg/m²]	Numérico Máx = 49.6 Min = 18.2
Diabetes Pedigree Function	Probabilidade de o paciente ter diabetes dado histórico familiar	Numérico Máx = 1.191 Min = 0.078
Age	Idade do paciente [Anos]	Numérico Máx = 66 Min = 21
Outcome	Previsão para o paciente ter ou não diabetes	Categórico Não = 0 Sim = 1

3 ETAPAS DE PRÉ-PROCESSAMENTO

Inicialmente, antes da utilização dos algoritmos de *Machine Learning*, foram realizadas algumas etapas de pré-processamento da base de dados. Tal representação está ilustrada na Figura 1 e descrita textualmente em seguida, respeitando a cronologia das ações aplicadas.

Figura 1: Ordem das etapas de pré-processamento.

(1) **Verificação de Valores nulos:** Ao realizar uma análise da presença de valores ausentes ou nulos na base de dados, observou-se que todos os atributos estavam preenchidos para todas as instâncias. No entanto, uma inconsistência foi identificada nos atributos "Glucose", "Blood Pressure", "Skin Thickness", "Insulin"e "BMI". Embora não houvesse instâncias com valores nulos, 9,43% das instâncias continham valores zero, o que é incoerente, especialmente considerando que são atributos nos quais zeros não são permitidos.
Para abordar essa inconsistência, foi realizado um tratamento específico. Nos casos em que foram identificados valores zero, esses foram convertidos para *"NaN- Not a Number* -, possibilitando uma nova verificação de valores nulos. Essa abordagem revelou, de fato, a presença de valores ausentes na base de dados.

Diante da necessidade de lidar com valores ausentes, optou-se pelo uso do algoritmo KNNImputer. Esse algoritmo baseia-se no cálculo da distância entre o valor ausente e seus vizinhos mais próximos, proporcionando uma imputação de dados mais robusta e contribuindo para mitigar possíveis inconsistências [14]. A escolha do KNNImputer como método de imputação reflete a intenção de preservar a estrutura dos dados, levando em consideração a proximidade entre as instâncias na abordagem de imputação. Assim, foi utilizado o hiperparâmetro "*n_neighbors=1*".

(2) **Verificação de Instâncias Duplicadas:** foi necessário analisar a possível presença de instâncias duplicadas na base de dados, de forma a evitar que a revisão seja tendenciosa para determinada classe, bem como impedir que treino e teste sejam realizados com instâncias iguais. Para tanto, foi utilizado o método *"duplicated()"* da biblioteca *"pandas"*, no qual retornou que não havia instâncias duplicadas; logo, zero instâncias foram removidas nesta etapa.

(3) **Remoção de *Outliers*:** Durante a análise exploratória, identificou-se a presença de *outliers* por meio da construção de gráficos *BoxPlot*. Diante dessa constatação, tornou-se necessário realizar a filtragem das observações que excediam a faixa de corte inferior e superior, determinadas pela análise dos quartis Q1 (primeiro quartil) e Q3 (terceiro quartil). Para efetuar essa remoção, calculou-se a diferença entre Q3 e Q1, conhecida como Interquartil (IQR). Utilizando essa medida, os limites de corte para *outliers* foram estabelecidos, conforme a Equação 1. Os valores que ultrapassaram esses limites foram então excluídos da análise [3].
Essa abordagem resultou na remoção de 190 instâncias que continham valores considerados *outliers*. A eliminação dessas observações extremas teve como objetivo aprimorar a robustez e a representatividade da análise estatística, garantindo resultados mais confiáveis e condizentes com a distribuição dos dados. Essa estratégia contribui para a obtenção de insights mais consistentes e informados a partir do conjunto de dados.

Equação 1: Cálculo de Outliers.

$$IQR = Q3 - Q1$$

$$lower\ bound = Q1 - 1.5 * IQR$$

$$upper\ bound = Q3 - 1.5 * IQR$$

$$Outliers = valores < lower\ bound\ ||\ valores > upper\ bound$$

(4) **Análise de correlação:** Por meio da função "*corr()*", foram realizadas duas análises para verificar a correlação entre os atributos. Primeiramente, foi obtida a matriz de correlação de um atributo com os demais, no qual verificou que não haviam atributos correlacionados, pois a maior correlação foi de 54%. Em seguida, foi testado se algum atributo estava influenciando diretamente a classificação. Porém, também não existia, pois todos os valores estavam abaixo de 50%.

(5) **Normalização:** Foi executada a etapa de normalização de todos os atributos. Para realizar esse processo, optou-se pelo método *Min-Max Scaling*, utilizando o *MinMaxScaler()*. Essa técnica é empregada para transformar os valores dos atributos de modo que o valor mais elevado de cada atributo seja ajustado para 1.0, enquanto o valor mínimo é estabelecido em 0.0. Essa normalização garante que todos os valores fiquem dentro dessa faixa específica.
O método *MinMaxScaler* é uma abordagem comumente utilizada para escalar características de forma que fiquem compreendidas entre um intervalo específico, nesse caso, entre 0.0 e 1.0 [16]. Esse processo é essencial para evitar que a escala original dos atributos influencie de maneira desproporcional algoritmos sensíveis à escala, como muitos algoritmos de aprendizado de máquina. Portanto, ao aplicar a normalização, busca-se garantir a consistência e a adequação dos dados normalizados para análises posteriores ou para a entrada em modelos de aprendizado de máquina.

(6) **Verificação Sobre Codificar Atributos:** para uma abordagem nos algoritmos de *Machine Learning* com o *Python3*, seria necessário codificar os atributos nominais para o formato numérico, porém a base já possuía apenas dados numéricos, dispensando qualquer codificação adicional.

(7) **Divisão dos Conjuntos de Treinamento e de Teste:** o conjunto de dados foi dividido em dois subconjuntos de treinamento e de teste com a utilização do método *train_test_split*. Nele, foi ajustado os parâmetros, com o uso do *stratify*, para que seja mantida uma proporção de 60% para treinamento e de 40% para teste. Além disso, com a utilização do *random_state=42*, garante-se que, a cada nova execução, as instâncias selecionadas sejam as mesmas.

(8) **Balanceamento de Classes:** a desigualdade na distribuição das classes foi uma grande preocupação, visto que, originalmente, 500 amostras pertencentes à classe "Sem Diabetes"e 268 à "Com Diabetes". Para evitar essa

discrepância, utilizou-se o método *Synthetic Minority Oversampling Technique (SMOTE)*, com o hiperparâmetro *sampling_strategy*, no qual ajustou-se para que a classe minoritária para o treino tenha uma proporção de 75% diante das amostras da classe majoritária. Além disso, ao se utilizar *random_state=42*, garante-se que, a cada nova execução, as instâncias criadas sejam as mesmas, evitando erros durante a análise das métricas. Na Tabela 2, são apresentadas as quantidades de instâncias de cada classe da base original. Em seguida, tem-se a coluna com a quantidade de instâncias após todas as etapas de pré-processamento, na qual foi buscado um balanceamento entre as classes. Por fim, nas últimas duas colunas, é mostrado a quantidade de amostras que foram para o treino e as que foram para o teste. Vale ressaltar que a divisão para o teste, por ter sido realizada antes do balanceamento *SMOTE*, segue a proporção da base original.

Tabela 2: Divisão das instâncias em treino e teste

	Base Original	Base Tratada	Treino	Teste
Não	500	390	233	157
Sim	268	249	174	75
Total	768	639	407	232

4 DESCRIÇÃO DOS MÉTODOS UTILIZADOS

Ambiente de Desenvolvimento

O projeto foi desenvolvido no ambiente do *Google Colaboratory*, uma plataforma baseada em nuvem que oferece acesso a recursos de computação e que permite a colaboração entre os membros da equipe em tempo real. O código foi implementado em *Python3*.

Decision Tree

O algoritmo *Decision Tree* foi escolhido para criar o primeiro modelo de classificação para o estudo. A árvore de decisão é uma técnica supervisionada amplamente utilizada em tarefas de classificação devido à interpretabilidade e à capacidade de capturar relacionamentos não lineares nos dados [10]. A ideia básica do algoritmo se dá diante de dividir o problema mais complexo em problemas mais simples, cujas soluções, ao serem combinadas na forma de uma árvore, resolvem a questão como um todo [10].

Ajuste de Hiperparâmetros. Para otimizar o desempenho do modelo, optou-se por utilizar um ajuste de hiperparâmetros, que foi conduzido por meio da técnica de *Random Search* [10]. Esta permite explorar diferentes combinações, visando encontrar a melhor configuração para a árvore.

Tabela 3: Configuração de Hiperparâmetros para *Decision Tree*

Hiperparâmetro	Range de Busca	Selecionado
criterion	'gini' e 'entropy'	'gini'
splitter	'best' e 'random'	'best'
max_depth	None list(np.arange(1, 21))	7
min_samples_split	list(np.arange(2, 21))	6
min_samples_leaf	list(np.arange(2, 21))	5
max_features	'auto', 'sqrt', 'log2' None	'auto'

Naïve Bayes

Para contrastar com o método simbólico anteriormente utilizado, optou-se por um algoritmo de classificação probabilística: o *Naïve Bayes* Gaussian. Enquanto o método simbólico era baseado em regras lógicas e representações, o *Naïve Bayes* emprega uma abordagem estatística. Este algoritmo pressupõe que os recursos de entrada são representados por contagens discretas na forma percentual e são independentes entre si; isto é, dado as classes, são ignoras todas as dependências entre os atributos [12]. Durante o treinamento, o modelo calcula as probabilidades condicionais de cada valor discreto para cada classe, de forma que os atributos possuam uma probabilidade diferente de ocorrer dada a classe pertencente. Sendo assim, ele faz previsões, utilizando essas probabilidades para calcular a chance de determinada instância pertencer à classe "Diabetes"ou à "Não Diabetes". Em seguida, faz-se a resposta final do algoritmo é a classe que possuir maior probabilidade.

Bagging

O "*Bagging*", ou *"Bootstrap Aggregating"*, é uma poderosa técnica *ensemble* usada para melhorar a estabilidade e a precisão de algoritmos de *Machine Learning*, visando principalmente mitigar o *overfitting*. A abordagem fundamental do *Bagging* é treinar vários modelos, geralmente árvores de decisão no contexto mencionado, em diferentes subconjuntos aleatórios do conjunto de treinamento original [10]. Essa diversificação dos conjuntos de dados de treinamento ajuda a reduzir a variância dos modelos individuais.

No caso específico mencionado, além de utilizar árvores de decisão, a técnica *Bagging* foi estendida para incorporar *Support Vector Machines* (SVM) no processo de *ensemble* [15]. Ao empregar o SVM no *Bagging*, busca-se obter benefícios adicionais em termos de desempenho e generalização. O SVM é conhecido por ser eficaz em problemas de classificação e, ao integrá-lo no contexto do *Bagging*, a combinação de múltiplos modelos SVM treinados em subconjuntos distintos contribui para um modelo composto mais robusto e confiável.

Os seguintes hiperparâmetros foram considerados para o *Random Search*:

Tabela 4: Configuração de Hiperparâmetros para o *Bagging*

Hiperparâmetro	Range de Busca	Selecionado
n_estimators	10, 30, 50, 100, 200	30
max_samples	0.5, 0.7, 0.9, 1.0	0.5
max_features	0.5, 0.7, 0.9, 1.0	1.0

Random Forest

O *Random Forest* é um algoritmo *ensemble* que opera construindo múltiplas árvores de decisão e combinando os resultados chegando em um único resultado. Além disso, baseia-se no sistema *Bootstrap Aggregation* (*Bagging*), o qual é capaz de possibilitar que cada uma das árvores sejam diferentes uma das outras, de forma a buscar reduzir a taxa de erro e melhorar as métricas, resultado em um modelo mais robusto e completo [10]. Um dos benefícios de se usar esse algoritmo é que ele retorna, de uma maneira muito mais compreensiva, a importância atribuída para cada variável na hora de realizar as predições. Com isso, é possível medir o impacto de cada questão no resultado.

Ajuste de Hiperparâmetros. Tal como na *Decision Tree*, foi utilizado o *Random Search* para otimizar e ajustar os hiperparâmetros.

Para este algoritmo, o *Random Search* encontrou o seguinte ajuste nos hiperparâmetros:

Tabela 5: Configuração de Hiperparâmetros para *Random Forest*

Hiperparâmetro	Range de Busca	Selecionado
n_estimators	50, 100, 150, 200	200
max_depth	None, 10, 20, 30, 50	None
min_samples_split	2, 5, 10	5
min_samples_leaf	1, 2, 4, 8	8
criterion	'gini', 'entropy' e 'log_loss'	'entropy'
max_features	'sqrt', 'log' e 5	5

Rede Neural Artificial - *Multilayer Perceptron*

As Redes Neurais Artificiais (RNAs), ou Redes Neurais Simuladas, são modelos matemáticos baseados na lógica da estrutura neural de seres inteligentes, ou seja, baseiam-se nas sinapses realizadas entre os neurônios destes seres para simular a sua lógica de aprendizado por experiência [17]. Essas células, unidades fundamentais do Sistema Nervoso Humano, desempenham um papel relevante na determinação do funcionamento e do comportamento do organismo, bem como no processo de raciocínio. Os neurônios consistem em três partes principais: os dendritos, que atuam como terminais de entrada; o corpo central; e os axônios, que são extensões longas utilizadas para transmitir informações para outras células [17].

Sob essa ótica, as RNAs são compostas neurônios artificiais (Perceptrons) que se conectam, de forma a interligar cada entrada a um respectivo peso associado, contendo as camadas de entrada (instâncias) e uma camada de saída (resposta do algoritmo). Assim, baseando-se no aprendizado por experiência, após o processamento de determinada instância, a Rede Neural determina se houve erro ou acerto em relação à saída desejada. Tal algoritmo busca melhorar sua precisão ao longo do tempo, de forma que a cada iteração os pesos são reajustados para aumentar o acerto para a base de dados [4].

Para o problema, foi utilizado um tipo de Rede Neural: o *Multilayer Perceptron* (MLP). Este foi desenvolvido para resolver problemas não linearmente separáveis - o que é uma limitação do Perceptron simples -, a alternativa mais utilizada é adicionar uma ou mais camadas intermediárias [10]. Tal como o Perceptron simples, algoritmo de *Multilayer Perceptron* continua possuindo as camadas de entrada e de saída com neurônios interconectados; porém, diferentemente do Perceptron Simples que utiliza somente uma função de ativação para impor um limite, como ReLU ou sigmóide, o Perceptron Multicamada pode escolher arbitrariamente a função que gerar o melhor resultado em cada camada[10].

5 RESULTADOS E DISCUSSÕES

Métricas

No âmbito da classificação de pacientes diabéticos, é crucial priorizar a identificação precisa dos casos verdadeiramente positivos. Classificar erroneamente um paciente como portador de diabetes, quando não é o caso, pode gerar preocupações temporárias, porém, em última análise, causar danos menores. Por outro lado, a falha em diagnosticar corretamente um paciente com diabetes pode acarretar sérias consequências. Nesse cenário, a métrica de *Recall*, que reflete a taxa de verdadeiros positivos, ganha importância na avaliação de algoritmos, uma vez que seu aumento garante a identificação eficaz dos pacientes que realmente necessitam de tratamento e monitoramento contínuos.

Tabela 6: Comparação de Métricas Entre os Algoritmos para a Classe Não

Classe	Método	Métricas		
		Precisão	*Recall*	*F1-Score*
Não	***Decision Tree***	0.85	0.76	0.80
	Naïve Bayes	0.88	0.86	0.87
	Random Forest	0.88	0.84	0.86
	Bagging	0.83	0.87	0.85
	Rede Neural	0.79	0.80	0.79

Tabela 7: Comparação de Métricas Entre os Algoritmos para a Classe Sim

Classe	Método	Métricas		
		Precisão	*Recall*	*F1-Score*
Sim	***Decision Tree***	0.59	0.71	0.64
	Naïve Bayes	0.72	0.75	0.73
	Random Forest	0.70	0.76	0.73
	Bagging	0.70	0.64	0.67
	Rede Neural	0.57	0.55	0.56

Nas Tabelas 6 e 7, são apresentadas as métricas de Precisão, *Recall* e *F1-Score* para cada um dos algoritmos utilizados. Como o objetivo é majoritariamente analisar o *Recall* da classe de pessoas com diabetes, observa-se que o *Bagging* se destaca ao se tratar da Classe Não (0), já o *Random Forest* apresentou melhor desempenho diante da Classe Sim (1). Nesse sentido, o *Random Forest* foi o modelo que atendeu melhor ao problema, indicando uma capacidade significativa do modelo em identificar e acertar dentro da própria classe, resultando em uma alta taxa de verdadeiros positivos.

Validação Cruzada

A validação cruzada (CV) é uma técnica essencial no campo de aprendizado de máquina para avaliar de maneira robusta o desempenho de um modelo, prevenindo o *overfitting*. Ao examinar diversas configurações de hiperparâmetros, como o parâmetro C em uma Máquina de Vetores de Suporte (SVM), há o perigo de adaptação excessiva aos dados de teste. A CV aborda esse problema ao reter um conjunto de teste para a avaliação final, eliminando, assim, a necessidade de um conjunto de validação [10]. Por meio dela, é possível que o modelo de *Machine Learning* gerado seja testado com diferentes instâncias, afim de uma melhor análise dos resultados.

Após o treinamento dos modelos, optou-se por empregar uma abordagem de validação cruzada com 5 subconjuntos para cada um deles, observando uma média significativamente menor em comparação com a métrica de *Recall* previamente utilizada. Essa escolha reflete a preferência por uma técnica robusta de avaliação, dividindo o conjunto de treinamento em cinco partes distintas. A métrica de média inferior sugere que, em média, o modelo pode estar capturando uma porcentagem menor de instâncias positivas em relação ao conjunto de dados.

Tabela 8: Recalls Médios dos Modelos, usando Cross-Validation

Modelo	Recall Médio (CV)
Decision Tree	0.51
Naive Bayes	0.60
Random Forest	0.57
Bagging	0.47
Rede Neural	- - -

Apesar das métricas mais baixas observadas utilizando a validação cruzada, é importante ressaltar que essas métricas estão mais robustas e oferecem um espaço significativo para melhorias no desempenho do modelo. A validação cruzada, ao dividir o conjunto de treinamento em subconjuntos distintos, proporciona uma avaliação mais abrangente, evitando decisões baseadas em uma única divisão arbitrária de dados. Isso resulta em métricas mais realistas e menos tendenciosas causadas por escolhas específicas dos conjuntos de treinamento e teste.

A verificação das métricas mais baixas, uma principal hipótese pode ser considerada. Primeiramente, o *Recall* é sensível à capacidade do modelo em identificar instâncias positivas, o que pode ser influenciado por fatores como a natureza do conjunto de dados e a prevalência das classes. Aparentemente, o que levou a esse problema pode estar relacionado com o desequilíbrio significativo entre as classes.

Teste T

O chamado teste t é apropriado para comparar dois conjuntos de dados quantitativos, em termos de seus valores médios [3]. Dessa forma, a hipotése de um modelo ser melhor que outro poderá ser validada de forma mais eficiente. Para tanto, os dois modelos determinados pelo *Cross Validation*, *Random Forest* e *Naive Bayes*, foram selecionados para o teste.

Ao comparar o valor de t com o valor crítico de t para o nível de significância escolhido (0.05). Foi-se obtido que **não há diferença significativa** entre os dois modelos, já que o valor de t (calculado igual a 0.38) foi maior do que o valor crítico.

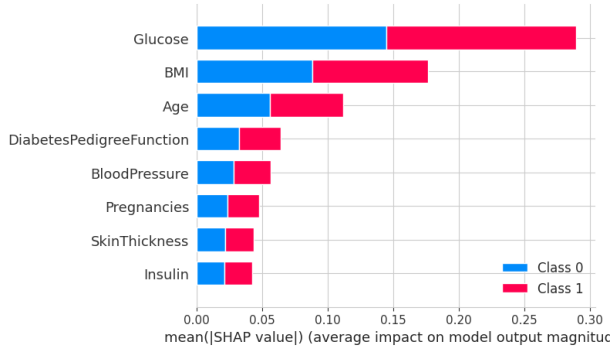
Interpretabilidade e Importância dos Atributos

A interpretabilidade é uma consideração crucial ao avaliar modelos de *Machine Learning*, especialmente para aplicações em saúde. Modelos do tipo "Caixa Preta", como *Random*

Forest e *Naïve Bayes*, podem apresentar desafios de transparência, tornando difícil entender as regras e decisões tomadas pelo algoritmo [1]. Isso levanta preocupações éticas, pois decisões com viés preconceituosos podem ser tomadas. Para suavizar esse problema, utilizou-se a biblioteca *SHapley Additive exPlanations (SHAP)* para avaliar a contribuição de cada atributo na predição do modelo.

O *SHAP* é uma biblioteca que utiliza a Teoria dos Jogos, um modelo matemático que analisa a contribuição de cada jogador para o resultado final de uma partida [11]. No contexto de *Machine Learning*, o *SHAP* avalia como cada atributo da base de dados afeta a predição final do modelo, de forma a comparar a importância de cada atributo na determinação se uma instância pertence a uma determinada classe [2]. Dessa forma, o *SHAP* foi aplicado para avaliar a importância dos atributos da base de dados e determinar quais foram os mais relevantes para rotular se um indivíduo possui ou não diabetes.

Figura 2: Resultado *SHAP* para o algoritmo *Random Forest*



Implicações para Profissionais da Saúde

Na Figura 2, é apresentada a saída do *SHAP* para a *Random Forest* para compreender melhor como este modelo obteve um ótimo resultado. Foi percebido que a glicose é o atributo mais determinante para classificar o indivíduo quanto à Diabetes. Todavia, para o *Decision Tree* e para o Naïve Bayes, por exemplo, a glicose também foi a característica mais relevante para a determinação da saúde do paciente. Assim, faz-se necessário interpretar outros atributos.

Sob essa ótica, em segundo lugar, tem-se o Índice de Massa Corporal (IMC), calculado com o uso da massa corporal do indivíduo, como um fator determinante para classificar um paciente com um possível quadro de Diabetes. Assim, como o IMC possui uma correlação linear com a massa do paciente, entende-se que, com o aumento dela, o corpo tende a precisar de mais nutrientes e, por conseguinte, aumenta-se a necessidade de produção de insulina. Dessa forma, o tecido adiposo tende a crescer, de tal modo que a necessidade do hormônio no corpo seja cada vez maior e mais urgente [9]. Tal cenário caótico é o início da Diabetes Tipo 2, marcada pela insuficiência da insulina para as demandas do corpo, aumentando o açúcar no corpo do paciente [6].

Desse modo, entende-se que existe uma alta correlação entre os dois principais atributos considerados pela *Random Forest*. O indivíduo que apresenta massa corporal elevada, podendo incluir obesidade, por exemplo, tende a ter uma carência da insulina no corpo, de tal forma que a glicose não é metabolizada corretamente. Sendo assim, o nível de açúcar no corpo é extremamente elevado, tendendo a ter Diabetes.

Tendo isso, é possível compreender a forte influência da obesidade e da pouca atividade física no quadro da doença. Portanto, um indivíduo que apresenta tal sintoma, somado à taxa de glicose alterada, tende a possuir a doença, sendo necessárias ações para mitigar tais aspectos.

Profissionais de saúde podem usar essas descobertas para direcionar estratégias preventivas, por ter maior consciência de quais características estão diretamente relacionadas ao diagnóstico de diabetes. A promoção de um estilo de vida saudável, principalmente para pessoas de maior idade, incluindo dieta balanceada e atividade física regular, pode ser fundamental na prevenção da diabetes. Identificar indivíduos com sinais de obesidade e níveis elevados de glicose pode orientar intervenções precoces e estratégias de gerenciamento [8].

6 CÓDIGO DESENVOLVIDO

Link para o código.

REFERÊNCIAS

[1] Sprint PrograMaria Inteligência Artificial. 2020. Algoritmos de Inteligência Artificial (IA) e Vieses: uma reflexão sobre ética e justiça. https://www.programaria.org/algoritmos-de-inteligencia-artificial-e-vieses-uma-reflexao-sobre-etica-e-justica/

[2] Abid Ali Awan. 2023. An Introduction to SHAP Values and Machine Learning Interpretability. https://www.datacamp.com/tutorial/introduction-to-shap-values-machine-learning-interpretability#rdl

[3] Marcelo Menezes; BORNIA Antonio Cezar BARBETTA, Pedro Alberto; REIS. 2010. *Estatística: para cursos de engenharia e informática.* São Paulo: Atlas - E-book.

[4] IBM Cloud. 2021. O que são Redes Neurais? https://www.ibm.com/br-pt/topics/neural-networks

[5] Sociedade Brasileira de Diabetes. [n.d.]. Diabetes gestacional exige cuidados. https://diabetes.org.br/diabetes-gestacional-exige-cuidados/

[6] Sociedade Brasileira de Diabetes. [n.d.]. Tipos de Diabetes. https://diabetes.org.br/tipos-de-diabetes/

[7] Sociedade Brasileira de Endocrinologia e Metabiologia. 2007. O que é diabetes. https://www.endocrino.org.br/o-que-e-diabetes/

[8] Secretaria de Saúde do Estado do Paraná. [n.d.]. O que é diabetes. https://www.saude.pr.gov.br/Pagina/Diabetes-diabetes-mellitus

[9] Fernanda de Almeida Escobar. 2017. Relação entre Obesidade e Diabete Mellitus Tipo II em Adultos. *Cadernos UniFOA* 4, 11 (mar. 2017), 69–72. https://doi.org/10.47385/cadunifoa.v4.n11.1004

[10] Katti FACELI. 2021. *Inteligência artificial: uma abordagem de aprendizado de máquina.* LTC - E-book.

[11] Dicionário Financeiro. [n.d.]. Teoria dos Jogos: definição e exemplos. https://www.dicionariofinanceiro.com/teoria-dos-jogos/

[12] Liangxiao Jiang, Shasha Wang, Chaoqun Li, and Lungan Zhang. 2016. Structure extended multinomial naive Bayes. *Information Sciences* 329 (2016), 346–356. https://doi.org/10.1016/j.ins.2015.09.037 Special issue on Discovery Science.

[13] Akshay Dattatray Khare. 2022. Diabetes Dataset. https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset

[14] Scikit Learn. [n.d.]. 6.4. Imputação de valores faltantes. https://scikit-learn.org/stable/modules/impute.html#knnimpute

[15] Scikit Learn. [n.d.]. C-Support Vector Classification. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[16] Scikit Learn. [n.d.]. Normalização. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[17] USP. [n.d.]. Redes Neurais Artificiais. https://sites.icmc.usp.br/andre/research/neural/