MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL RURAL DA AMAZÔNIA

CAMPUS CAPITÃO POÇO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

Capitão poço – PA

2025

Relatório: Funcionamento do Código e Entendimento sobre Rede Neural

1. Objetivo do Código
   O código implementa um sistema de diagnóstico médico utilizando uma Rede Neural Artificial (RNA) para prever se um paciente está saudável ou doente, com base em sintomas fornecidos pelo usuário. Ele também classifica a gravidade da condição e fornece possíveis causas para doenças graves.

2. Funcionamento do Código
   Entrada de Dados:
   O usuário insere os seguintes sintomas:
   - Temperatura Corporal
   - Dor de Cabeça
   - Fadiga

3. Dados Simulados
   O código utiliza um conjunto de dados simulados (X e Y) para treinar a rede neural:
   - X: Contém as características (temperatura, dor de cabeça, fadiga).
   - Y: Contém os rótulos (0 = Saudável, 1 = Doente).

4. Diagnóstico
   O modelo prevê se o paciente está saudável ou doente com base nos sintomas fornecidos. A gravidade da condição é classificada com a função classificar_gravidade, que considera: Temperatura corporal, Presença de dor de cabeça e Presença de fadiga.

5. Exibição dos Resultados
   Diagnóstico: Se o paciente for saudável, uma mensagem de sucesso é exibida. Se o paciente estiver doente, a gravidade é exibida. Para casos de "Doença Grave", o código exibe um alerta recomendando que o paciente procure um médico e lista possíveis causas.

6. Motivos do Diagnóstico: O código exibe os sintomas que levaram ao diagnóstico. Probabilidades: As probabilidades de o paciente estar saudável ou doente são exibidas (se o modelo suportar predict_proba). Avaliação do Modelo: A matriz de confusão e a acurácia do modelo são exibidas para avaliar o desempenho.

7. Entendimento sobre Rede Neural
   Uma Rede Neural Artificial é um modelo computacional inspirado no funcionamento do cérebro humano. Ela é composta por neurônios artificiais organizados em camadas, que processam informações de forma interconectada.

8. Componentes da Rede Neural no Código
   Camada de Entrada: Recebe os dados normalizados (temperatura, dor de cabeça, fadiga). Camada Oculta: Contém 5 neurônios que processam os dados e aprendem padrões complexos. Camada de Saída: Produz a previsão final (0 = Saudável, 1 = Doente).
9. Treinamento da Rede Neural
   Durante o treinamento, a rede ajusta os pesos e vieses para minimizar o erro entre as previsões e os rótulos reais. O algoritmo de retropropagação é usado para atualizar os pesos com base no gradiente do erro.

10. Vantagens das Redes Neurais
    Capacidade de aprender padrões complexos. Aplicável a uma ampla variedade de problemas, como classificação, regressão e reconhecimento de padrões.

11. Métricas de Avaliação
    Matriz de Confusão: Mostra o desempenho do modelo em prever corretamente as classes (Saudável ou Doente). Acurácia: Mede a proporção de previsões corretas feitas pelo modelo.

Código em python:

```
import streamlit as st

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import confusion_matrix

from sklearn.preprocessing import StandardScaler

import pandas as pd



st.set_page_config(page_title="Diagnóstico Médico", page_icon="🧠")

st.title("🩺 Diagnóstico Médico com Inteligência Artificial")



# Dados simulados

X = np.array([

   [36.5, 0, 0],

   [38.7, 1, 1],
```

```
    [37.2, 0, 0],
    [39.5, 1, 1],
    [37.8, 1, 0],
    [40.0, 1, 1],
    [36.9, 0, 0],
    [39.0, 1, 1]
])
y = np.array([0, 1, 0, 1, 1, 1, 0, 1])  # 0 = Saudável, 1 = Doente

# Treinamento
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Normalizar os dados
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

clf = MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000, random_state=42)
clf.fit(X_train, y_train)

# Entradas
st.markdown("Preencha os sintomas abaixo:")
temperatura = st.slider("Temperatura Corporal (°C)", 35.0, 41.0, 36.5, 0.1)
dor_de_cabeca = st.selectbox("Dor de cabeça?", ["Não", "Sim"])
fadiga = st.selectbox("Fadiga?", ["Não", "Sim"])

entrada = np.array([[temperatura, int(dor_de_cabeca == "Sim"), int(fadiga == "Sim")]])
entrada = scaler.transform(entrada)
```

```
predicao_modelo = clf.predict(entrada)[0]
proba = clf.predict_proba(entrada)[0]

# Classificações médicas reais
def classificar_temperatura(temp):
    if temp < 35.4:
        return "Hipotermia"
    elif 35.4 <= temp <= 37.4:
        return "Normal"
    elif 37.5 <= temp <= 37.8:
        return "Febrícula"
    elif 37.9 <= temp <= 39.0:
        return "Febre"
    else:
        return "Febre Alta"

def classificar_gravidade(temp, dor, fadiga):
    if temp < 37.5 and not dor and not fadiga:
        return "Saudável"
    elif 37.5 <= temp < 38.0:
        return "Doença Leve"
    elif 38.0 <= temp <= 39.0 and dor and fadiga:
        return "Doença Moderada"
    elif temp > 39.0 and dor and fadiga:
        return "Doença Grave"
    else:
        return "Doença Leve"
```

```
nivel_temp = classificar_temperatura(temperatura)
gravidade = classificar_gravidade(temperatura, dor_de_cabeca == "Sim", fadiga == "Sim")

# Decisão final com lógica reforçada
if gravidade != "Saudável":
    predicao_final = 1
else:
    predicao_final = predicao_modelo

# Resultado
st.subheader("🔬 Resultado do Diagnóstico:")
if predicao_final == 0:
    st.success("**Diagnóstico Previsto: Saudável**")
    st.write("✅ Nenhum sinal grave detectado.")
else:
    st.error("**Diagnóstico Previsto: Doente**")
    st.write(f"🧬 Gravidade da condição: **{gravidade}**")
    if gravidade == "Doença Grave":
        st.warning("⚠️ **Comunicado Importante:** Procure um médico imediatamente para avaliação e tratamento adequado.")

        # Possíveis causas de Doença Grave
        st.write("### Possíveis causas para Doença Grave:")
        st.write("- **Infecções graves**: Pneumonia, meningite, septicemia.")
        st.write("- **Doenças inflamatórias**: Febre reumática, doença de Kawasaki.")
        st.write("- **Condições metabólicas**: Crise tireotóxica, desidratação severa.")
```

```
    st.write("- **Outras causas**: Choque séptico, complicações de doenças crônicas.")

# Motivos
st.write("### Motivos para o diagnóstico:")
st.write(f"- **Temperatura Corporal:** {temperatura:.1f} °C ({nivel_temp})")
if dor_de_cabeca == "Sim":
  st.write("- **Dor de Cabeça:** Presente")
else:
  st.write("- **Dor de Cabeça:** Ausente")
if fadiga == "Sim":
  st.write("- **Fadiga:** Presente")
else:
  st.write("- **Fadiga:** Ausente")

# Probabilidade
st.write("### Probabilidades do Modelo:")
if hasattr(clf, "predict_proba"):
  proba = clf.predict_proba(entrada)[0]
  st.write(f"- Saudável: {proba[0] * 100:.1f}%")
  st.write(f"- Doente: {proba[1] * 100:.1f}%")
else:
  st.warning("O modelo não suporta cálculo de probabilidades.")

# Avaliação
y_pred = clf.predict(X_test)
acuracia = np.mean(y_pred == y_test) * 100
```

```
# Gerar a matriz de confusão com todas as classes
cm = confusion_matrix(y_test, y_pred, labels=[0, 1])  # 0 = Saudável, 1 = Doente
labels = ["Saudável", "Doente"]

# Exibir a matriz de confusão
st.subheader("📊 Avaliação do Modelo")
st.write("### Matriz de Confusão:")
st.dataframe(pd.DataFrame(cm, index=labels, columns=labels))

# Exibir a acurácia
st.write(f"- **Acurácia:** {acuracia:.2f}%")
```

Relatório: Diagnóstico de Gripe com Redes Bayesianas

1. Importação das Bibliotecas
   O código utiliza as seguintes bibliotecas para implementar e visualizar a Rede Bayesiana:
   - DiscreteBayesianNetwork: Cria a estrutura da Rede Bayesiana, definindo as relações entre as variáveis.
   - TabularCPD: Define as Probabilidades Condicionais (CPDs) para cada variável da rede.
   - VariableElimination: Permite realizar inferências na rede, calculando probabilidades condicionais.
   - matplotlib.pyplot: Utilizado para criar gráficos de barras horizontais que visualizam as probabilidades calculadas.
   - streamlit: Cria uma interface interativa para o usuário inserir sintomas e visualizar os resultados.

2. Definição da Estrutura da Rede
   A Rede Bayesiana modela a relação entre a gripe e os sintomas associados:
   Estrutura da Rede:
   - Gripe → Febre
   - Gripe → Dor
   - Gripe → Fadiga
   - Gripe → Tosse

Interpretação: A presença de gripe influencia diretamente a probabilidade de apresentar febre, dor de cabeça, fadiga e tosse.

3. Definição das Probabilidades Condicionais (CPDs)
   As CPDs definem as probabilidades de cada variável, considerando as relações estabelecidas na rede.

   CPD 1: Probabilidade de Gripe

   A variável "Gripe" pode ter dois estados:

   - 0 = Não Gripado
   - 1 = Gripado

   Probabilidades:

   - 60% de chance de não estar gripado (P(Gripe=0) = 0.6).
   - 40% de chance de estar gripado (P(Gripe=1) = 0.4).

   CPD 2: Probabilidade de Febre Condicionada à Gripe

   - A variável "Febre" depende da presença ou ausência de gripe.

   Matriz de Probabilidades se não está gripado:

   - 80% de chance de não ter febre, 20% de chance de ter febre.
   - Se está gripado: 30% de chance de não ter febre, 70% de chance de ter febre.

   CPD 3: Probabilidade de Dor Condicionada à Gripe

   - A variável "Dor" depende da presença ou ausência de gripe.

   Matriz de Probabilidades:

   - Se não está gripado: 70% de chance de não ter dor, 30% de chance de ter dor.
   - Se está gripado: 40% de chance de não ter dor, 60% de chance de ter dor.

   CPD 4: Probabilidade de Fadiga Condicionada à Gripe

   - A variável "Fadiga" depende da presença ou ausência de gripe.

   Matriz de Probabilidades:

   - Se não está gripado: 60% de chance de não ter fadiga, 40% de chance de ter fadiga.
   - Se está gripado: 20% de chance de não ter fadiga, 80% de chance de ter fadiga.

   CPD 5: Probabilidade de Tosse Condicionada à Gripe

- A variável "Tosse" depende da presença ou ausência de gripe.

Matriz de Probabilidades:

- Se não está gripado: 90% de chance de não ter tosse, 10% de chance de ter tosse.
- Se está gripado: 40% de chance de não ter tosse, 60% de chance de ter tosse.

4. Adicionando as CPDs ao Modelo
   As CPDs são adicionadas ao modelo com o método add_cpds. Isso permite que a rede calcule probabilidades condicionais com base nas relações definidas.

5. Verificação do Modelo
   O método check_model verifica se as CPDs estão consistentes e se a rede é matematicamente válida. Caso contrário, o código exibe uma mensagem de erro.

6. Inferência na Rede Bayesiana
   A inferência é realizada com o método VariableElimination, que permite calcular probabilidades condicionais com base em evidências fornecidas pelo usuário. O usuário insere os seguintes sintomas:
   - Febre: "Sim" ou "Não".
   - Dor de Cabeça: "Sim" ou "Não".
   - Fadiga: "Sim" ou "Não".
   - Tosse: "Sim" ou "Não".

   Essas entradas são convertidas para valores binários:

   - 1 = Sim
   - 0 = Não

   Consulta à Rede

   A consulta calcula a probabilidade de estar gripado (P(Gripe=1)) e não gripado (P(Gripe=0), dado os sintomas fornecidos.

7. Visualização dos Resultados
   Os resultados são exibidos de forma textual e gráfica.
   - Probabilidade de não estar gripado (P(Gripe=0)).
   - Probabilidade de estar gripado (P(Gripe=1)).
   - Mensagens de alerta baseadas na probabilidade de gripe:
   - Alta chance de gripe: Se P(Gripe=1) > 70%.

- Possível gripe: Se 40% < P(Gripe=1) ≤ 70%.
- Baixa chance de gripe: Se P(Gripe=1) ≤ 40%.

Gráfico de Barras Horizontais

Um gráfico de barras horizontais exibe as probabilidades de estar gripado e não gripado. Configuração do Gráfico:

- Categorias: "Gripado" e "Não Gripado".
- Valores: Probabilidades calculadas (P(Gripe=1) e P(Gripe=0)).
- Cores: Vermelho para "Gripado" e verde para "Não Gripado"

8. Exemplo de Saída
   Entrada do Usuário
   - Febre: "Sim"
   - Dor de Cabeça: "Sim"
   - Fadiga: "Não"
   - Tosse: "Sim"

   Saída do Modelo

   - Probabilidade de NÃO estar gripado: 35.0%
   - Probabilidade de estar gripado: 65.0%

   Mensagem: "🟠 Possível gripe detectada. Fique atento aos sintomas."

   Gráfico

   Um gráfico de barras horizontais mostrando:

   - "Gripado": 65.0%
   - "Não Gripado": 35.0%

Código em python:

```
import streamlit as st

from pgmpy.models import DiscreteBayesianNetwork

from pgmpy.factors.discrete import TabularCPD

from pgmpy.inference import VariableElimination

import matplotlib.pyplot as plt


# Configuração da página
```

```
st.set_page_config(page_title="Diagnóstico de Gripe - Rede Bayesiana",
page_icon="🤧")

st.title("🤧 Diagnóstico de Gripe com Redes Bayesianas")

st.markdown("Preencha os sintomas abaixo para obter uma probabilidade de estar
gripado:")


# Entrada do usuário

febre = st.selectbox("Está com febre?", ["Não", "Sim"])

dor = st.selectbox("Está com dor de cabeça?", ["Não", "Sim"])

fadiga = st.selectbox("Está com fadiga?", ["Não", "Sim"])

tosse = st.selectbox("Está com tosse?", ["Não", "Sim"])


# Modelo Bayesiano

model = DiscreteBayesianNetwork([

  ('Gripe', 'Febre'),

  ('Gripe', 'Dor'),

  ('Gripe', 'Fadiga'),

  ('Gripe', 'Tosse')

])


# CPDs

cpd_gripe = TabularCPD('Gripe', 2, [[0.6], [0.4]])


cpd_febre = TabularCPD('Febre', 2, [[0.8, 0.3], [0.2, 0.7]], evidence=['Gripe'],
evidence_card=[2])

cpd_dor = TabularCPD('Dor', 2, [[0.7, 0.4], [0.3, 0.6]], evidence=['Gripe'],
evidence_card=[2])

cpd_fadiga = TabularCPD('Fadiga', 2, [[0.6, 0.2], [0.4, 0.8]], evidence=['Gripe'],
evidence_card=[2])
```

```
cpd_tosse = TabularCPD('Tosse', 2, [[0.9, 0.4], [0.1, 0.6]], evidence=['Gripe'],
evidence_card=[2])


model.add_cpds(cpd_gripe, cpd_febre, cpd_dor, cpd_fadiga, cpd_tosse)


# Inferência
if not model.check_model():
    st.error("Erro na rede bayesiana.")
else:
    infer = VariableElimination(model)
    evidencias = {
        'Febre': 1 if febre == "Sim" else 0,
        'Dor': 1 if dor == "Sim" else 0,
        'Fadiga': 1 if fadiga == "Sim" else 0,
        'Tosse': 1 if tosse == "Sim" else 0
    }
    resultado = infer.query(variables=['Gripe'], evidence=evidencias)

    prob_nao_gripado = round(resultado.values[0] * 100, 2)
    prob_gripado = round(resultado.values[1] * 100, 2)

    # Resultado
    st.subheader("🔍 Resultado do Diagnóstico:")
    st.write(f"- **Probabilidade de NÃO estar gripado:** {prob_nao_gripado}%")
    st.write(f"- **Probabilidade de estar GRIPADO:** {prob_gripado}%")

    if prob_gripado > 70:
        st.error("⚠️ Alta chance de estar gripado! Procure atendimento médico.")
```

```
    elif prob_gripado > 40:
        st.warning("🟠 Possível gripe detectada. Fique atento aos sintomas.")
    else:
        st.success("🟢 Provavelmente não está gripado.")

    # Gráfico horizontal
    st.subheader("📊 Visualização das Probabilidades")
    fig, ax = plt.subplots(figsize=(6, 3.5))
    categorias = ['Gripado', 'Não Gripado']
    valores = [prob_gripado, prob_nao_gripado]
    cores = ['red', 'green']

    bars = ax.barh(categorias, valores, color=cores)
    ax.set_xlim(0, 100)
    ax.set_xlabel("Probabilidade (%)")
    ax.set_title("Diagnóstico de Gripe")

    for bar in bars:
        largura = bar.get_width()
        ax.text(largura + 1, bar.get_y() + bar.get_height()/2,
                f'{largura:.1f}%', va='center', fontweight='bold')
st.pyplot(fig)
```

Quadro 1 – Distribuição dos artigos de acordo com os autores, ano de publicação, título, método e conclusão.

Autores	Ano	Título	Método	Conclusão

Juan Manuel Adán Coello, Renan de Oliveira Yamaguti	2016	Algoritmo híbrido de recomendaçã o utilizando algoritmo genético	Combinação de dois algoritmos de filtragem colaborativa cujas saídas são combinadas utilizando um algoritmo genético.	O algoritmo híbrido proposto apresentou uma precisão preditiva superior à de cada um dos algoritmos de filtragem colaborativa isoladamente, demonstrando a eficácia da combinação com algoritmo genético.
Amanda Gomes Sampaio	2020	Análise de sentimentos	Estudo sobre as redes sociais como fonte de dados para análise de sentimentos, apresentando métodos envolvendo IA.	A análise de sentimentos em redes sociais, utilizando técnicas de inteligência artificial, é crucial para compreender opiniões públicas e tem aplicações significativas em marketing, saúde pública e política.
Waldemar Bonventi Júnior	2021	Lógica “Fuzzy”: fundamentos e aplicabilidad e	Discussão sobre a lógica fuzzy e sua aplicação na modelagem e controle de sistemas complexos, com exemplo prático.	A lógica fuzzy é eficaz na modelagem e controle de sistemas complexos, oferecendo soluções viáveis para sistemas estáticos e destacando sua aplicabilidade em diversas áreas.