Relatório sobre Análise de Diabetes com Machine Learning

Gabriel Andrade do Nascimento

November 20, 2024

Abstract

Este relatório descreve o desenvolvimento de um sistema de previsão de diabetes utilizando o conjunto de dados de diabetes e técnicas de aprendizado de máquina. O sistema é baseado em um modelo de classificação Random Forest, com pré-processamento dos dados, treinamento do modelo e avaliação da acurácia. Também é implementada uma função interativa para prever o risco de diabetes com base em entradas do usuário.

1 Introdução

Este trabalho utiliza o conjunto de dados de diabetes disponibilizado no Kaggle para construir um modelo de aprendizado de máquina capaz de prever se uma pessoa tem risco de desenvolver diabetes com base em variáveis clínicas. O modelo escolhido é o Random Forest Classifier, uma técnica de ensemble que se destaca pela sua robustez e eficiência na classificação de dados complexos.

2 Objetivo

O objetivo deste trabalho é construir e avaliar um modelo de aprendizado de máquina para a previsão de diabetes, utilizando um conjunto de dados com informações de fatores clínicos como níveis de glicose, pressão arterial, índice de massa corporal (BMI), entre outros.

3 Metodologia

O processo de construção do modelo segue as etapas descritas abaixo:

1. **Importação e preparação dos dados:** O dataset é baixado do Kaggle e carregado em um DataFrame do Pandas.
2. **Pré-processamento:** As variáveis são normalizadas, e os dados são divididos em conjuntos de treino e teste.

3. **Treinamento do modelo:** Um modelo de Random Forest é treinado com os dados de treino.
4. **Avaliação do modelo:** A acurácia do modelo é avaliada com os dados de teste.
5. **Predição interativa:** O modelo é utilizado para realizar predições com base em entradas fornecidas pelo usuário.

4 Código

Abaixo estão os dois códigos utilizados para realizar a análise.

4.1 Código 1 - Importação do Dataset

```
import pandas as pd
import kagglehub

def importa_dataset_diabetes():
    """
    Faz o download do dataset de diabetes do Kaggle e retorna um DataFrame do Pandas.

    Returns:
        pd.DataFrame: DataFrame contendo os dados do dataset.
    """
    # Faz o download da versão mais nova
    path = kagglehub.dataset_download("akshaydattatraykhare/diabetes-dataset")
    print("Path to dataset files:", path)

    # Carrega a tabela em formato Pandas Dataframe
    file_path = f"{path}/diabetes.csv"
    df = pd.read_csv(file_path)
    return df
```

4.2 Código 2 - Pré-processamento, Treinamento e Predição

```
from importacao_insumos import importa_dataset_diabetes
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import numpy as np

def pre_processamento_modelo(df):
    # Separar features e o alvo
    X = df.drop("Outcome", axis=1)
```

```
    y = df["Outcome"]

    # Normalizar as features
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    # Dividir os dados em treino e teste
    X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, rand

    return X_train, X_test, y_train, y_test, scaler

def treinamento_modelo(X_train, y_train):
    # Treinar um modelo de classificação
    model = RandomForestClassifier(random_state=42)
    model.fit(X_train, y_train)
    return model

def avaliacao_modelo(model, X_test, y_test):
    # Realizar predições com o conjunto de teste
    y_pred = model.predict(X_test)

    # Calcular a acurácia
    acuracia = accuracy_score(y_test, y_pred)

    print(f"Acurácia do modelo: {acuracia * 100:.2f}%")
    return acuracia

def predicao_interativa(model, scaler):
    # Coletar inputs do usuário
    print("Por favor, insira os seguintes dados:")
    Pregnancies = float(input("Número de gravidezes: "))
    Glucose = float(input("Nível de glicose: "))
    BloodPressure = float(input("Pressão arterial: "))
    SkinThickness = float(input("Espessura da pele: "))
    Insulin = float(input("Nível de insulina: "))
    BMI = float(input("Índice de massa corporal (BMI): "))
    DiabetesPedigreeFunction = float(input("Função de pedigree diabético: "))
    Age = float(input("Idade: "))

    # Montar a entrada para o modelo
    user_data = np.array([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin,
    user_data_scaled = scaler.transform(user_data)  # Normalizar os dados

    # Fazer a predição
    prediction = model.predict(user_data_scaled)
```

```
    probability = model.predict_proba(user_data_scaled)

    # Retornar os resultados
    if prediction[0] == 1:
        print(f"\n**Resultado**: Alta probabilidade de diabetes (Chance: {probability[0]
    else:
        print(f"\n**Resultado**: Baixa probabilidade de diabetes (Chance: {probability[0

def run():
    # Chamar a função para carregar o dataset
    df = importa_dataset_diabetes()

    # Chamar a função de pré-processamento
    X_train, X_test, y_train, y_test, scaler = pre_processamento_modelo(df)

    # Chamar a função de treinamento do modelo
    model = treinamento_modelo(X_train, y_train)

    # Avaliar o modelo e imprimir a acurácia
    avaliacao_modelo(model, X_test, y_test)

    # Chamar a função interativa para predição
    predicao_interativa(model, scaler)

run()
```

5 Explicação do Código

O primeiro código é responsável por importar o conjunto de dados de diabetes do Kaggle. Ele utiliza a biblioteca 'kagglehub' para baixar o arquivo CSV do dataset. O dataset contém várias colunas, como:

- **Pregnancies**: Número de gravidezes.
- **Glucose**: Nível de glicose no sangue.
- **BloodPressure**: Pressão arterial.
- **SkinThickness**: Espessura da pele.
- **Insulin**: Nível de insulina.
- **BMI**: Índice de massa corporal.
- **DiabetesPedigreeFunction**: Função de pedigree diabético.
- **Age**: Idade do paciente.

- **Outcome**: Rótulo indicando se a pessoa tem diabetes (1) ou não (0).

O segundo código realiza o pré-processamento dos dados, normalizando as variáveis e separando os dados em conjuntos de treino e teste. Em seguida, ele treina um modelo de Random Forest para prever a variável de saída (**Outcome**) com base nas outras variáveis. A acurácia do modelo é então calculada e exibida. Por fim, o código oferece uma funcionalidade interativa onde o usuário pode inserir suas próprias informações e obter uma previsão do risco de diabetes.

6 Link para o GitHub

O código completo está disponível no GitHub através do seguinte link: https://github.com/SkynnerBlack/F