|  |
| --- |
| EXPOSYS Data Labs |
| Predictions & models for Diabetes |
| Data Analysis |

|  |
| --- |
| ADITI SHUKLA  9/28/2024 |

Prediction for Diabetes Using Data-Science Models

Aditi Shukla,

[aditishukla564@gmail.com](mailto:aditishukla564@gmail.com)

**Table of Contents**

1. **Introduction**
2. **Objectives**
3. **Data Collection**
4. **Data Preprocessing**
5. **Exploratory Data Analysis (EDA)**
6. **Model Development**
7. **Model Evaluation**
8. **User interface**
9. **Results and Discussion**
10. **Deployment**
11. **Conclusion**

## 1. Introduction

Diabetes is a chronic disease characterized by high blood sugar levels. Early detection and intervention can significantly improve health outcomes. This project focuses on analyzing diabetes data to build a predictive model using the Decision Tree algorithm.

**2. Objectives**

* To analyze diabetes data for key insights.
* Develop Models For Prediction.
* Check Accuracy of Models, and use the best accuracy Model For Predictions.
* To develop a predictive model using Decision Tree algorithms.

## 3. Data Collection

## The **Diabetes prediction dataset** is a collection of medical and demographic data from patients, along with their diabetes status (positive or negative). The data includes features such as age, gender, body mass index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level. This dataset can be used to build machine learning models to predict diabetes in patients based on their medical history and demographic information

### .Dataset Features

* Age
* Gender
* Body mass index (BMI)
* Hypertension
* Heart disease
* Smoking History
* HbA1c level
* Blood glucose level
* Diabetes (0: No Diabetes, 1: Diabetes)

**4. Data Preprocessing**

* **Handling Missing Values:** Missing data was imputed with the median for continuous variables.
* **Feature Scaling:** Standardization was applied to normalize the feature set.

**5. Exploratory Data Analysis (EDA)**

* **Correlation Analysis:** A correlation matrix was created to identify relationships between variables.
* **Visualizations:** Histograms were generated to visualize distributions and relationships.

## 6. Model Development

### Naïve Bayes Model

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.
* **Use GussianNB**

### Decision Tree Model

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.
* **Hyperparameter Tuning:** Parameters such as max\_depth , random\_states, and min\_samples\_split were optimized using Grid Search.

### **Random Forest Model**

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.
* **Use random state 42**

### K Neighbors Model

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.
* **Use n\_neighbors = 1**

### **Logistic Regression Model**

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.

### Support Vector Model

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.
* **Use kernel and gamma**

### **Neural layers Model**

* **Library Used:** Scikit-learn.
* **Training and Testing Split:** The dataset was split into 80% training and 20% testing.
* **Use 4 layers, 1 Flatten, 3 Dense, actvation used ‘relu’ and ‘sigmoid’, optimizer is ‘adam’ ,loss is ‘binary-**cross-entropy’, epochs are 50 and batch-size is 10

## ****7.**** Model Evaluation

### Naïve Bayes Model:

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2, random\_state = 42)

model1 = GaussianNB()

model1.fit(x\_train, y\_train)

y\_pred1 = model1.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred1)

conf\_matrix = confusion\_matrix(y\_test, y\_pred1)

class\_report = classification\_report(y\_test, y\_pred1)

### Decision Tree Model:

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

model2 = DecisionTreeClassifier(criterion = "gini", random\_state = 42, max\_depth = 5, min\_samples\_leaf = 9)

model2.fit(x\_train, y\_train)

y\_pred2 = model2.predict(x\_test)

target = list(data['diabetes'].unique())

feature\_name = list(x.columns)

from sklearn.tree import export\_text

r = export\_text(model2, feature\_names = feature\_name)

r

accuracy = accuracy\_score(y\_test, y\_pred2)

conf\_matrix = confusion\_matrix(y\_test, y\_pred2)

class\_report = classification\_report(y\_test, y\_pred2)

import joblib

joblib.dump(model2, 'diabetes\_decision\_tree\_model\_last.pkl')

### Random Forest Model:

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

model3 = RandomForestClassifier(random\_state = 42)

model3.fit(x\_train, y\_train)

y\_pred3 = model3.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred3)

conf\_matrix = confusion\_matrix(y\_test, y\_pred3)

class\_report = classification\_report(y\_test, y\_pred3)

### KNeighors Models:

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

model4 = KNeighborsClassifier(n\_neighbors = 1)

model.4fit(x\_train, y\_train)

y\_pred4 = model4.predict(x\_test)

accuracy = accuracy\_score(y\_test, y\_pred4)

conf\_matrix = confusion\_matrix(y\_test, y\_pred4)

class\_report = classification\_report(y\_test, y\_pred4)

### Logistic Regression Model:

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

x\_train\_scaled = scaler.fit\_transform(x\_train)

x\_test\_scaled = scaler.transform(x\_test)

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

model5 = LogisticRegression()

model5.fit(x\_train\_scaled, y\_train)

y\_pred5 = model5.predict(x\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred5)

conf\_matrix = confusion\_matrix(y\_test, y\_pred5)

class\_report = classification\_report(y\_test, y\_pred5)

### SVM MODEL:

from sklearn import svm

from sklearn.model\_selection import GridSearchCV

model6 = svm.SVC(kernel = 'rbf', C = 1, gamma = 0.1)

model6.fit(x\_train\_scaled, y\_train)

y\_pred6 = model6.predict(x\_test\_scaled)

accuracy = accuracy\_score(y\_test, y\_pred6)

conf\_matrix = confusion\_matrix(y\_test, y\_pred6)

class\_report = classification\_report(y\_test, y\_pred6)

### Neural Layers Model:

import tensorflow as tf

model7 = tf.keras.Sequential()

layers = tf.keras.layers

model7.add(layers.Flatten())

model7.add(layers.Dense(16, input\_dim = x\_train.shape[1], activation = 'relu'))

model7.add(layers.Dense(8, activation = 'relu'))

model7.add(layers.Dense(1, activation = 'sigmoid'))

model7.compile(optimizer = 'adam', loss = 'binary\_crossentropy', metrics = ['accuracy'])

model7.fit(x\_train\_scaled, y\_train, epochs = 50, batch\_size = 10, validation\_data = (x\_test\_scaled, y\_test))

y\_pred7 = model7.predict(x\_test\_scaled)

accuracy = model7.evaluate(x\_test\_scaled, y\_test)

## 8. USER INTERFACE :

import streamlit as st

import joblib

import pandas as pd

# Load the trained model

model = joblib.load("C:/Users/hp/OneDrive/Desktop/jupyter notebook/diabetes/diabetes\_decision\_tree\_model\_last.pkl")

# Set up the Streamlit app title

st.title("Diabetes Prediction App")

# Create input fields for user data

gender = st.selectbox("Gender", options=["Female", "Male"])

age = st.number\_input("Age", min\_value=0, max\_value=120)

hypertension = st.selectbox("Hypertension", options=[0, 1])

heart\_disease = st.selectbox("Heart Disease", options=[0, 1])

smoking\_history = st.selectbox("Smoking History", options=["never", "current", "No Info"])

bmi = st.number\_input("BMI", format="%.2f")

hba1c\_level = st.number\_input("HbA1c Level", format="%.1f")

blood\_glucose\_level = st.number\_input("Blood Glucose Level", format="%.1f")

# Prepare the input data

input\_data = pd.DataFrame({

    'gender': [gender],

    'age': [age],

    'hypertension': [hypertension],

    'heart\_disease': [heart\_disease],

    'smoking\_history': [smoking\_history],

    'bmi': [bmi],

    'HbA1c\_level': [hba1c\_level],

    'blood\_glucose\_level': [blood\_glucose\_level]

})

# Convert categorical variables to dummy variables

input\_data = pd.get\_dummies(input\_data, drop\_first=True)

# Ensure input data matches the model's input

# Add missing columns if any

for col in model.feature\_names\_in\_:

    if col not in input\_data.columns:

        input\_data[col] = 0

# Make predictions

if st.button("Predict"):

    prediction = model.predict(input\_data)

    if prediction[0] == 1:

        st.success("The model predicts: Diabetes")

    else:

        st.success("The model predicts: No Diabetes")

#cd "C:\Users\hp\OneDrive\Desktop\jupyter notebook\diabetes"

#streamlit run interface.py

## 9. Results and Discussion

The Decision Tree model achieved an accuracy of approximately 97.215%. The confusion matrix indicated that the model successfully identified a high percentage of both diabetic and non-diabetic cases.

## 10.Deployment

## interface - Personal - Microsoft​ Edge 9_23_2024 10_22_07 PM.png

## 11. Conclusion

The Decision Tree model provided valuable insights and predictions for diabetes detection. While it demonstrated reasonable accuracy, the potential for overfitting highlights the need for further refinement and the consideration of alternative models.