AI BASED DIABETES PREDICTION SYSTEM

Creating a diabetes prediction system involves several data preprocessing steps. Here's an overview of the typical process:

****1.Data Collection****: Gather a comprehensive dataset that includes features such as age, gender, BMI, family history, blood pressure, and glucose levels, along with the target variable (diabetes status).

****2.Data Cleaning****: Remove or handle missing values, outliers, and duplicate records to ensure the dataset is of high quality.

****3.Feature Selection****: Identify which features are most relevant for predicting diabetes. You can use techniques like feature correlation analysis or feature importance.

****4.Feature Scaling****: Normalize or standardize numerical features to ensure they have a similar scale, which can improve the performance of machine learning algorithms.

****5.Categorical Encoding****: Convert categorical variables (e.g., gender) into numerical format (one-hot encoding or label encoding).

****6.Data Splitting****: Divide the dataset into training and testing sets to evaluate the model's performance.

****7.Data Balancing (if needed)****: Address class imbalance issues if the number of diabetes and non-diabetes cases is significantly skewed.

****8.Model-Specific Preprocessing****: Some machine learning algorithms may require specific preprocessing steps, such as scaling for SVM or embedding for neural networks.

****9.Data Pipeline****: Create a data pipeline to automate preprocessing, ensuring consistency between training and testing data.

****10.Save Preprocessing Configuration****: Save the preprocessing steps and parameters, so you can apply them consistently to new data when making predictions.

Once your data is preprocessed, you can train various machine learning models like decision trees, random forests, logistic regression, or neural networks to predict diabetes based on the prepared data.

PROGRAM:

Creating a program for diabetes prediction involves developing a machine learning model based on a dataset of relevant features and known diabetes outcomes. Here's a simplified Python program that uses the popular scikit-learn library to create a basic diabetes prediction model. Please note that this is a basic example, and real-world applications may require more comprehensive datasets and feature engineering.

# Import necessary libraries

import numpy as np

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

# Load your dataset. Make sure it has features and a target variable (diabetes in this case).

# You can obtain such a dataset from sources like UCI Machine Learning Repository.

# Load your dataset, assuming it's in a CSV file

data = pd.read\_csv('diabetes\_dataset.csv')

# Separate features and target variable

X = data.drop('diabetes', axis=1)

y = data['diabetes']

# Split the dataset into training and testing sets

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

# Create and train a machine learning model (Random Forest Classifier)

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = model.predict(X\_test)

# Calculate the accuracy of the model

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

print(f'Accuracy: {accuracy \* 100:.2f}%')

# You can use this model to predict diabetes for new data by passing feature values to model.predict()

# Example of predicting diabetes for a new sample (replace with your own data)

new\_sample = np.array([[125, 80, 35, 0, 0, 33.5, 0.545, 30]])

prediction = model.predict(new\_sample)

print(f'Predicted class: {"Diabetes" if prediction[0] else "No Diabetes"}')

Make sure to replace **'diabetes\_dataset.csv'** with the path to your dataset file and adapt the features and target variable according to your dataset. Also, it's crucial to choose the right machine learning algorithm, tune hyperparameters, and preprocess your data appropriately for better predictive performance in a real-world scenario.

窗体顶端

窗体底端

IMAGES:

