DIABETES PREDICTION PROJECT

* ABSTRACT:

Millions of people in the world are affected by diabetes. Early detection and taking the right measures can significantly increase the life span of an average human affected by diabetes. There are two types of diabetes: type1 and type 2, each involving different genetic as well as non-genetic factors related to the pancreases inability to produce insulin.   
  
The primary objective of this project is to train machine learning models using a dataset with variables such as glucose levels, blood pressure, skin thickness, insulin amount, BMI, Diabetes Pedigree Function, number of pregnancies, and age. We pre-processed the dataset by handling missing values, normalizing the data and replacing zero values in critical columns with RMS (root mean square) of the non-zero values.   
  
After preparing the data, we utilized two machine learning algorithms- (KNN) K-Nearest Neighbors and (DT) Decision Tree. The results showed 79.182% training accuracy and 70.472% testing accuracy for the KNN model and 100% training accuracy and 71.259% testing accuracy for the Decision Tree model.

* OBJECTIVE:

The Diabetes Prediction Project's fundamental objective is to use machine learning and predictive analysis to assess whether a person possesses diabetes. The program examines a number of health metrics, namely blood pressure, BMI, and glucose levels, with the aim to create a model that can accurately predict the risk of diabetes. This tool is designed to help medical professionals diagnose the disease early and initiate treatment for patients on time. Furthermore, it provides training on the primary causes of diabetes, which sheds new light on research and interventions in the realm of public health.

* INTRODUCTION:

Diabetes is a serious health problem that impacts millions of individuals worldwide. Early detection is essential for survival and to avoid major repercussions. This technique for predicting diabetes has been developed to assist in overcoming the difficulty of accurately identifying diabetes. In this study, a tool that determines if a patient is at risk of getting diabetes is trained using machine learning. The aim is to design an analytical model that can make accurate predictions through analyzing key health markers such as blood pressure, body mass index (BMI), and blood sugar levels. The objective of this tool is to help medical professionals diagnose patients at an early stage, which will enhance treatment results. Furthermore, the project will assist researchers to understand more about the elements that can lead to diabetes.

* METHODOLOGY:

The methodology for the Diabetes Prediction Project involves several key steps, from data collection to model deployment. Here’s an overview of the process:

1. Data Collection and Preprocessing:

The dataset is created by gathering information on different health markers from various resources such as hospital records. The data includes variables like glucose levels, blood pressure, BMI, age, and other relevant health metrics. The collected data is cleaned to handle missing values, remove duplicates, and correct any inconsistencies. This step also involves ensuring that all variables are in a comparable format.

2. Exploratory Data Analysis (EDA):

Exploratory Data Analysis (EDA) provides us with an understanding of the data we're working with by visualizing distributions, identifying patterns, and spotting anomalies. By taking a deeper look into the data, we can see how different features relate to diabetes risk. We also use statistical analyses to determine which features are most significant in predicting diabetes.

3. Feature Engineering:

In this step, we’re all about enhancing the features we’re using to make our model more effective. This could involve normalizing numerical features to bring them to a common scale or crafting new interaction terms that capture the relationships between different features.

4. Model Selection and Training:

In this model, we experiment with various machine learning algorithms, such as logistic regression and decision trees. The dataset is split into training and testing sets, to ensure the model can generalize well to new data.

During training, we tune hyperparameters using techniques like cross-validation to optimize the model’s performance. The model is trained using the training dataset, while the testing dataset is used to evaluate its accuracy and robustness.

5. Model Evaluation:

The trained models are evaluated using various metrics such as accuracy, precision, recall, F1 score, and area under the ROC curve (AUC-ROC). These metrics help in understanding how well the model is performing and where improvements can be made.

6. Model Interpretation and Validation:

The model is interpreted to understand which features are most influential in predicting diabetes risk. The model is validated using a separate validation dataset or through k-fold cross-validation to ensure it performs consistently across different data samples.

7. Deployment and Monitoring:

Once a satisfactory model is obtained, it is deployed in a real-world environment where healthcare providers can use it as a diagnostic aid. Continuous monitoring is set up to track the model’s performance over time, ensuring it remains accurate and reliable. Regular updates and retraining may be performed as more data becomes available or as medical standards change.

This structured methodology ensures a rigorous and comprehensive approach to building an effective diabetes prediction model, with the ultimate goal of supporting healthcare professionals in early diagnosis and treatment planning.

* CODE:

#DIABETES CODE:

#IMPORT LIBRARIES

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

#Import Dataset

dataset = pd.read\_csv("D:\Disha\Project\_Dataset\diabetes.csv")

dataset

dataset.info()

dataset.isnull().sum()

dataset.describe()

#Correlation plot of independent variables

plt.figure(figsize=(10,8))

sns.heatmap(dataset.corr(), annot=True, fmt=".3f", cmap="YlGnBu")

plt.title("Correlation Heatmap")

#Exploring pregnancy and target variables

plt.figure(figsize=(10,8))

#Plotting pregnancy and target variables on graph

plt.figure(figsize=(10, 8))

sns.kdeplot(dataset["Pregnancies"][dataset["Outcome"] == 1], color="Blue", shade=True, label="Positive")

sns.kdeplot(dataset["Pregnancies"][dataset["Outcome"] == 0], color="Red", shade=True, label="Negative")

plt.xlabel("Pregnancies")

plt.ylabel("Density")

plt.legend(["Positive", "Negative"])

plt.title("Distribution of Pregnancies by Outcome")

plt.show()

#Exploring Glucose and target variables

plt.figure(figsize=(10,8))

sns.violinplot(data=dataset,x="Outcome",y="Glucose",split=True, linewidth=2, inner="quart")

#Glucose

plt.figure(figsize=(10, 8))

sns.kdeplot(dataset["Glucose"][dataset["Outcome"] == 1], color="Blue", shade=True, label="Positive")

sns.kdeplot(dataset["Glucose"][dataset["Outcome"] == 0], color="Red", shade=True, label="Negative")

plt.xlabel("Glucose")

plt.ylabel("Density")

plt.legend(["Positive", "Negative"])

plt.title("Distribution of Glucose by Outcome")

plt.show()

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

dataset = pd.read\_csv("D:\\Disha\\Project\_Dataset\\diabetes.csv")

def calculate\_rms(dataset):

non\_zero\_elements = dataset[dataset != 0]

rms\_value = np.sqrt(np.mean(non\_zero\_elements \*\* 2))

return rms\_value

# Replace zeros with the RMS of the respective column

columns\_to\_replace = ["Glucose", "BloodPressure", "BMI", "Insulin", "SkinThickness", "DiabetesPedigreeFunction"]

for column in columns\_to\_replace:

column\_data = dataset[column].to\_numpy()

rms\_value = calculate\_rms(column\_data)

column\_data[column\_data == 0] = rms\_value

dataset[column] = column\_data

# Display the dataset

print(dataset)

x=dataset.drop(["Outcome"],axis=1)

y=dataset["Outcome"]

x

x=dataset.drop(["Outcome"],axis=1)

y=dataset["Outcome"]

y

#Splitting data into training and testing dataset

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.33,random\_state=42)

x\_train

#KNN

from sklearn.neighbors import KNeighborsClassifier

training\_accuracy=[]

test\_accuracy=[]

for n\_neighbors in range(1,11):

knn=KNeighborsClassifier(n\_neighbors=n\_neighbors)

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

#checking accuracy

training\_accuracy.append(knn.score(x\_train,y\_train))

test\_accuracy.append(knn.score(x\_test,y\_test))

plt.plot(range(1,11),training\_accuracy,label="training\_accuracy")

plt.plot(range(1,11),test\_accuracy,label="test\_accuracy")

plt.xlabel("Accuracy")

plt.xlabel("n\_neighbors")

plt.legend()

knn=KNeighborsClassifier(n\_neighbors=9)

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

print(knn.score(x\_train,y\_train),":Training accuracy")

print(knn.score(x\_test,y\_test),":Test accuracy")

from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier(random\_state=0)

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

print(dt.score(x\_train,y\_train),":Training accuracy")

print(dt.score(x\_test,y\_test),":Test accuracy")

* CONCLUSION:

The Decision Tree model is slightly better at predicting diabetes when we test it on new data, with a test accuracy of 71.26%. However, it’s worth noting that it scored a perfect 100% on the training data, which might suggest it’s memorizing the data too well and could be overfitting. In essence, the model becomes too specialized to the training data and doesn’t generalize well to new data.

The KNN model, while not as perfect on the training data, shows a pretty close test accuracy of 70.47%. This might indicate it’s doing a better job of generalizing to new data.

So, while the Decision Tree is a bit more accurate in testing, the KNN model might be more robust and less likely to overfit. Fine-tuning and further testing could help improve these models even more.

-Disha Vidyasagar Pavoor