REPORT ON

DIABETES PREDICTION USING DECISION TREE

D.Chandra Sekhar (2021BCSE07AED149)
M.Rahul Raj (2021BCSE07AED125)
G.Rajesh Naidu (2021BCSE07AED146)

A mini project report submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

Branch: COMPUTER SCIENCE AND ENGINEERIGN

Specialisation: AIML

of Alliance University

APRIL 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ALLIANCE COLLEGE OF ENGINEERING AND DESIGN

ALLIANCE UNIVERSITY, BENGALURU

ALLIANCE COLLEGE OF ENGINEERING AND DESIGN

(ALLIANCE UNIVERSITY, BENGALURU)

MEDIACL INSURENCE PRICE PREDICTION USING RANDOM FOREST

Bona fide record of work done by

D.Chandra Sekhar (2021BCSE07AED149)
M.Rahul Raj (2021BCSE07AED125
G.Rajesh Naidu (2021BCSE07AED146)

A mini project report submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

Branch: COMPUTER SCIENCE AND ENGINEERING

Specialization: AIML

Of Alliance University

April 2024

…………..……………….

Dr. Chetan J Shelke
Faculty guide
Department of Computer Science and Engineering
Alliance College of Engineering and Design

ABSTRACT

Diabetes, a chronic metabolic disorder, poses significant public health challenges worldwide. Early detection and prediction of diabetes risk are crucial for effective prevention and management strategies. Machine learning (ML) techniques have shown promising results in predicting diabetes onset based on various clinical and demographic factors. This paper presents a comprehensive review of recent advancements in ML-based diabetes prediction models. We discuss different types of ML algorithms employed, including decision trees, support vector machines, neural networks, and ensemble methods, along with their strengths and limitations. Furthermore, we analyze the features used in these models, such as demographic information, clinical measurements, lifestyle factors, and genetic markers. Additionally, we evaluate the performance metrics utilized to assess the predictive accuracy of these models, such as sensitivity, specificity, accuracy, and area under the receiver operating characteristic curve. Finally, we highlight the challenges and future directions in diabetes prediction research, emphasizing the need for robust, interpretable, and personalized prediction models to aid in early intervention and improved healthcare outcomes for individuals at risk of developing diabetes.

INTRODUCTION

Diabetes mellitus, characterized by elevated blood sugar levels, presents a significant health burden globally, affecting millions of individuals and posing substantial challenges to healthcare systems. Early detection and prediction of diabetes risk are vital for implementing timely interventions and personalized management strategies to mitigate complications and improve outcomes. In recent years, machine learning (ML) techniques have emerged as valuable tools in predicting diabetes onset, leveraging various clinical, demographic, lifestyle, and genetic factors to generate accurate risk assessments. This paper provides an overview of recent developments in ML-based diabetes prediction models, examining the types of algorithms utilized, the features incorporated, and the performance metrics employed to evaluate predictive accuracy. Additionally, we discuss the implications of these models for healthcare practice and highlight the challenges and opportunities in advancing diabetes prediction research. By exploring the current landscape of ML-driven diabetes prediction, this paper aims to contribute to the ongoing efforts to enhance early detection and preventive care for individuals at risk of developing diabetes.

METHODOLOGY

Data Collection:

This involves gathering relevant data points from patients, which often includes

Medical history

Blood test results (glucose levels)

Physical characteristics (age, weight, height)

Lifestyle habits (diet, exercise)

Data Preprocessing:

The raw data might need cleaning and preparation for analysis. This could involve,

Handling missing values

Identifying and correcting errors

Transforming data (e.g., scaling numerical values)

Feature Selection:

Not all collected data points may be equally important for prediction. Techniques are used to identify the most impactful features that contribute to the model's accuracy.

Model Training:

The chosen machine learning algorithm is trained on a portion of the data. The model learns to recognize patterns that differentiate diabetic and non-diabetic individuals based on the features.

Model Evaluation:

The trained model's performance is assessed using another portion of the data (testing set). Metrics like accuracy, sensitivity, and specificity are used to gauge how well the model predicts diabetes.

Model Optimization:

Based on the evaluation, the model might be fine-tuned by adjusting parameters or trying different algorithms. The goal is to achieve the most accurate and reliable predictions.

IMPLEMENTATION AND OUTPUT

Importing the Dependencies

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

Data Collection and Analysis

PIMA Diabetes Dataset

```
df = pd.read_csv('/content/diabetes (1).csv')
```

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.tail()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
df.shape
```

(768, 9)

```
df.dtypes
```

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

```
[ ] df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
[ ] df['Outcome'].value_counts()
```

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

0 --> Non-Diabetic

1 --> Diabetic

```
[ ] df.groupby('Outcome').mean()
```

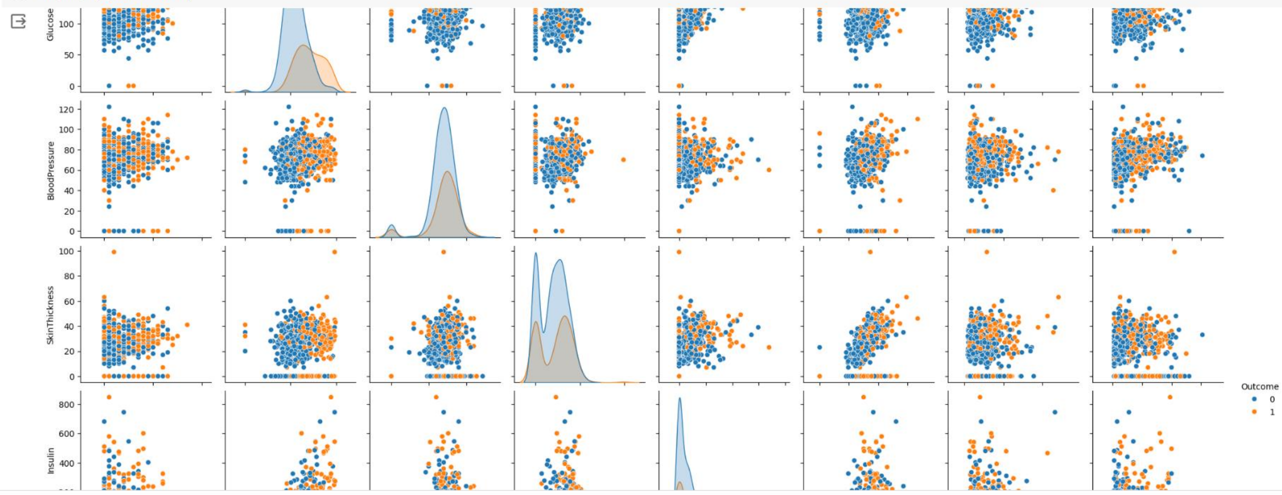
Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

```
[ ] X = df.drop(columns = 'Outcome', axis=1)
    Y = df['Outcome']
```

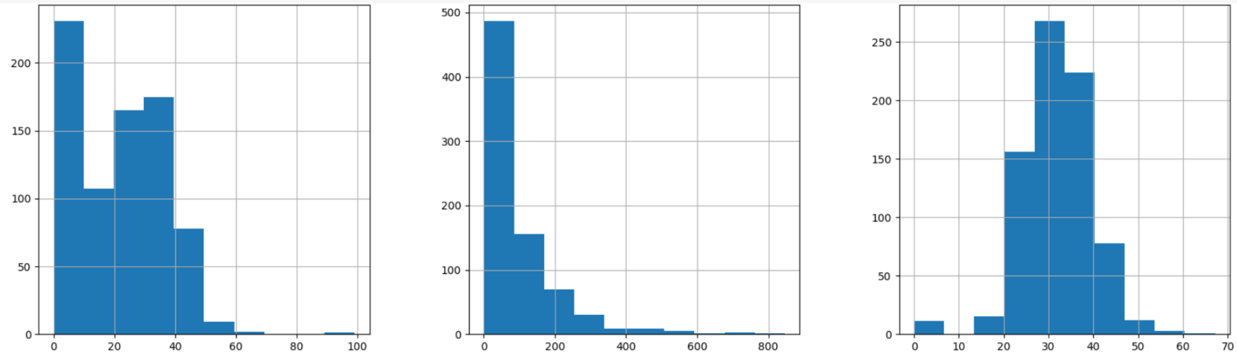
```
[ ]  print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

```
sns.pairplot(df,hue='Outcome')
```

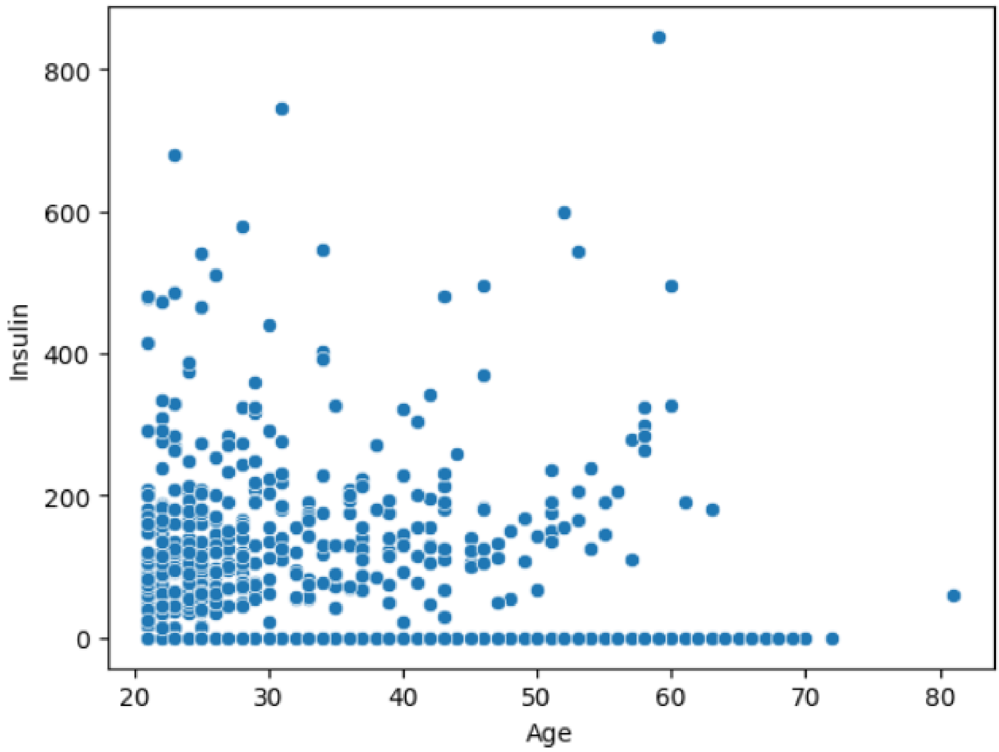


```
col=['Glucose' ,'BloodPressure' ,'SkinThickness', 'Insulin' ,'BMI']
for i in col:
  #df[i].replace(0,df[i].mean(),inplace=True)
  p=df.hist(figsize = (20,20))
```



```
sns.scatterplot(x='Age',y='Insulin',data=df)
```

<Axes: xlabel='Age', ylabel='Insulin'>



Data Standardization

```
[ ] scaler = StandardScaler()
```

```
[ ] scaler.fit(X)
```

```
▾ StandardScaler
StandardScaler()
```

```
[ ] standardized_data = scaler.transform(X)
```

```
print(standardized_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

```
[ ] X = standardized_data
    Y = df['Outcome']
```

Train Test Split

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2, stratify=Y, random_state=2)
```

```
[ ] print(X.shape, X_train.shape, X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

Training the Model

```
[ ]  from sklearn.tree import DecisionTreeClassifier

     dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5)
     dtc.fit(X_train, Y_train)


     dtc_acc= accuracy_score(Y_test,dtc.predict(X_test))

     print("Train Set Accuracy:"+str(accuracy_score(Y_train,dtc.predict(X_train))*100))
     print("Test Set Accuracy:"+str(accuracy_score(Y_test,dtc.predict(X_test))*100))
```

```
Train Set Accuracy:80.94462540716613
Test Set Accuracy:73.37662337662337
```

```
from sklearn.model_selection import train_test_split          #splitting the dataset

train,val_train,test,val_test = train_test_split(X,Y,test_size=.50,random_state=3)
```

Making a Predictive System

```
from sklearn.tree import DecisionTreeClassifier

# Assuming you have a trained model stored in 'models'
# Create an instance of the model
input_data = (5,116,74,0,0,25.6,0.201,30)

# changing the input_data to numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the array as we are predicting for one instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardize the input data
std_data = scaler.transform(input_data_reshaped)
models = DecisionTreeClassifier()

# Fit the model before making predictions
models.fit(X_train, Y_train)

# Now you can use the model to make predictions
prediction = models.predict(std_data)
print(prediction)

if prediction[0] == 0:
   print('The person is not diabetic')
else:
   print('The person is diabetic')
```

```
[0]
The person is not diabetic
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

CONCLUSION

In conclusion, decision trees offer a powerful and interpretable framework for predicting diabetes risk based on various demographic, clinical, lifestyle, and genetic factors. Through recursive partitioning of the feature space, decision trees provide insight into the complex decision-making process underlying diabetes onset. Our review highlights the importance of early detection and prevention in managing diabetes, and the role of machine learning techniques, particularly decision trees, in facilitating personalized healthcare interventions. While decision trees offer simplicity and interpretability, they may face challenges such as overfitting and limited predictive performance with complex datasets. However, ensemble methods like Random Forests and Gradient Boosting Machines can mitigate these issues by combining multiple decision trees. By leveraging decision trees and other machine learning approaches, we can improve early detection, intervention, and management of diabetes, ultimately leading to better healthcare outcomes and quality of life for individuals at risk of this chronic metabolic disorder.