AI DIABETES PREDICTION SYSTEM

NAME:GOWTHAM.S

REG NO:720421104015

INTRODUCTION:

AI-based diabetes prediction systems use machine learning algorithms to analyze medical data and predict the likelihood of a person developing diabetes. These systems are designed to help healthcare professionals identify patients who are at high risk of developing diabetes and take preventive measures to reduce the risk.

The machine learning algorithms used in these systems are trained on large datasets of medical records, which include information such as age, gender, body mass index (BMI), blood pressure, and glucose levels. The algorithms learn to identify patterns in the data that are associated with the development of diabetes and use these patterns to make predictions about future cases.

There are several types of machine learning algorithms that can be used for diabetes prediction, including decision trees, logistic regression, support vector machines (SVMs), and artificial neural networks (ANNs). Each algorithm has its own strengths and weaknesses, and the choice of algorithm depends on the specific requirements of the application.

For example, decision trees are simple to understand and interpret but may not be as accurate as other algorithms. Logistic regression is a popular algorithm for binary classification problems like diabetes prediction. SVMs are effective for high-dimensional data but may be computationally expensive. ANNs are powerful algorithms that can learn complex patterns in the data but require large amounts of training data.

DATASET LINK:

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

DATASET:

Pregnanci	Glucose	BloodPres	SkinThickr	Insulin	BMI	DiabetesF	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1
10	125	70	26	115	31.1	0.205	41	1
7	147	76	0	0	39.4	0.257	43	1
1	97	66	15	140	23.2	0.487	22	0
13	145	82	19	110	22.2	0.245	57	0

Here's a list of tools and software commonly used in the process:

1. Programming Language:

- Python is the most popular language for machine learning due to

its extensive libraries and frameworks. You can use libraries like NumPy,

pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE):

- Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.

3. Machine Learning Libraries:

- You'll need various machine learning libraries, including:

- scikit-learn for building and evaluating machine learning models.

- TensorFlow or PyTorch for deep learning, if needed.

- XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools:

- Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

5. Data Preprocessing Tools:

- Libraries like pandas help with data cleaning, manipulation, and preprocessing.

6. Data Collection and Storage:

- Depending on your data source, you might need web scraping tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite, PostgreSQL) for data storage.

7. Version Control:

- Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

8. Notebooks and Documentation:

- Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating README files and documentation.

PROGRAM:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model  import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
data = pd.read_csv('/kaggle/input/diabetes-data-set/diabetes.csv')
data.head()
data.describe()
data.isnull().sum()
data['BMI'] = data['BMI'].replace(0,data['BMI'].mean())
data['BloodPressure'] = data['BloodPressure'].replace(0,data['BloodPressure'].mean())
data['Glucose'] = data['Glucose'].replace(0,data['Glucose'].mean())
data['Insulin'] = data['Insulin'].replace(0,data['Insulin'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0,data['SkinThickness'].mean())
import matplotlib.pyplot as plt
import seaborn as sns
fig, ax = plt.subplots(figsize=(15,10))
sns.boxplot(data=data, width= 0.5,ax=ax,  fliersize=3)
X = data.drop(columns = ['Outcome'])
y = data['Outcome']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
```

```
X_train.shape, X_test.shape

import pickle

##standard Scaling- Standardization

def scaler_standard(X_train, X_test):

    #scaling the data

    scaler = StandardScaler()

    X_train_scaled = scaler.fit_transform(X_train)

    X_test_scaled = scaler.transform(X_test)


    #saving the model

    file = open('standardScalar.pkl','wb')

    pickle.dump(scaler,file)

    file.close()

return X_train_scaled, X_test_scaled

X_train_scaled, X_test_scaled = scaler_standard(X_train, X_test)

X_train_scaled

log_reg = LogisticRegression()

log_reg.fit(X_train_scaled,y_train)
```

MODEL CHOSEN FOR THE PREDICTION:

KNN algorithm is used for predicting the ai diabetes for the given data set.

KNN Algorithm:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score


dataset=pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")

print(len(dataset))

print(dataset.head())
```

OUTPUT:

```
768

   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


   DiabetesPedigreeFunction  Age  Outcome

0                     0.627   50        1

1                     0.351   31        0

2                     0.672   32        1

3                     0.167   21        0

4                     2.288   33        1
```

```
nonzero=['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
```

```
for col in nonzero:
    dataset[col]=dataset[col].replace(0,np.NaN)
    mean=int(dataset[col].mean(skipna=True))
    dataset[col]=dataset[col].replace(np.NaN,mean)
print(dataset['Glucose'])
```

OUTPUT:

```
0      148.0
1       85.0
2      183.0
3       89.0
4      137.0
       ...
763    101.0
764    122.0
765    121.0
766    126.0
767     93.0
Name: Glucose, Length: 768, dtype: float64
```

```
x=dataset.iloc[:,0:8]

y=dataset.iloc[:,8]

x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.3)
```

```
sc=StandardScaler()

x_train=sc.fit_transform(x_train)

x_test=sc.transform(x_test)

classifier=KNeighborsClassifier(n_neighbors=15,p=2,metric='euclidean')

model=classifier.fit(x_train,y_train)

yp=classifier.predict(x_test)

yp
```

OUTPUT:

```
array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
```

0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0])

```
CM=confusion_matrix(y_test,yp)
print(CM)
```

OUTPUT:

```
[[133  13]
 [ 34  51]]
```

```
print("F-Score: ",(f1_score(y_test,yp)))
```

OUTPUT:

F-Score: 0.6845637583892616

ADVANTAGES:

Predicting diabetes using machine learning algorithms has several advantages. Here are some of them:

Early detection: Machine learning algorithms can identify patients who are at high risk of developing diabetes before the onset of symptoms. This allows healthcare professionals to take preventive measures to reduce the risk of developing diabetes.

Improved accuracy: Machine learning algorithms can analyze large datasets of medical records and identify patterns that are associated with the development of diabetes. This can lead to more accurate predictions and better patient outcomes.

Personalized treatment: Machine learning algorithms can be used to develop personalized treatment plans for patients with diabetes. By analyzing a patient's medical history, the algorithms can identify the most effective treatments for that patient.

Reduced healthcare costs: Early detection and prevention of diabetes can lead to reduced healthcare costs by avoiding expensive treatments and hospitalizations.

Improved patient outcomes: By identifying patients who are at high risk of developing diabetes and providing personalized treatment plans, machine learning algorithms can improve patient outcomes and quality of life.

DISADVANTAGES:

There are some potential disadvantages of using machine learning algorithms for diabetes prediction. Here are some of them:

Data bias: Machine learning algorithms are only as good as the data they are trained on. If the training data is biased or incomplete, the algorithm may produce inaccurate predictions.

Privacy concerns: Machine learning algorithms require access to large amounts of medical data, which can raise privacy concerns. Patients may be hesitant to share their medical information with healthcare providers or researchers.

Lack of transparency: Some machine learning algorithms are difficult to interpret, which can make it challenging for healthcare professionals to understand how the algorithm arrived at a particular prediction.

Cost: Developing and implementing machine learning algorithms can be expensive, which may limit their availability in some healthcare settings.

False positives and false negatives: Machine learning algorithms can produce false positives (predicting diabetes when it is not present) and false negatives (failing to predict diabetes when it is present), which can lead to incorrect diagnoses and treatment plans.

It's important to note that these disadvantages can be mitigated with careful design and implementation of machine learning algorithms. For example, bias in the training data can be addressed by using diverse datasets and ensuring that the data is representative of the population being studied. Similarly, privacy concerns can be addressed by implementing appropriate data security measures and obtaining informed consent from patients.

CONCLUSION:

Based on the analysis of the AI diabetes prediction system, it can be concluded that the system is capable of predicting diabetes disease effectively, efficiently, and instantly . The proposed model gives the best results for diabetic prediction.The performance of AI in disease prediction models for diabetes is expected to improve dramatically in the future due to the availability of organized data and abundant computational resources.