AI Based Diabetes Prediction System

Problem Definition:

The problem is to bulid an AI-powered diabetes prediction system that uses machine learning alogorithms to analyze the medical data and predict the likelihood of an individuals developing diabetes.The system aims to provide early risk assessment and personalized preventive measures,allowing individuals to take proactive actions to manage their Health.

Preview:

1. Collect the medical dataset
2. After that pre-processing the data
3. Analyze the dataset
4. Provide the accurate result.

Data Collection:

The dataset must contain the medical feature such as **glucose level,blood pressure,BMI**,etc.,along with information about whether the individual has diabetes or not.The dataset are obtained from Kaggle.com.

Dataset:

B	C	D	E	F	G	H	I
Gulcose	**Blood pressure**	SkinThickness	Insulin	**BMI**	DiabetesPedigreeFunction	Age	Outcome
148	72	35	0	33.6	0.627	50	1
85	66	29	0	26.6	0.351	31	0
183	64	0	0	23.3	0.672	32	1
89	66	23	94	28.1	0.167	21	0
137	40	35	168	43.1	2.288	33	1
116	74	0	0	25.6	0.201	30	0
78	50	32	88	31	0.248	26	1
115	0	0	0	35.3	0.134	29	0
197	70	45	543	30.5	0.158	53	1
125	96	0	0	0	0.232	54	1
110	92	0	0	37.6	0.191	30	0
168	74	0	0	38	0.537	34	1
139	80	0	0	27.1	1.441	57	0
189	60	23	846	30.1	0.398	59	1
166	72	19	175	25.8	0.587	51	1
100	0	0	0	30	0.484	32	1
118	84	47	230	45.8	0.551	31	1
107	74	0	0	29.6	0.254	31	1

Parameters Description:

If a person have BMI level 30 or greater, Blood pressure 130/80 mm hg and glucose level before a meal:80 to 130 mg/dl.Two hours after the start of a meal:less than 180 mg/dl the person have diabetes.

Data pre-processing:

This phase of model handles inconsistent data in order to get more accurate and precies results.If the dataset contains missing value,we imputed missing values for few attributes like Gulcose levels,Blood pressure,BMI,age because these attributes cannot contain zero values.Then we scale the dataset to normalize all values.

Features Selection:

- Using this model we can easily predict the diabetes occur to the person in future or not and prevent from the diabetes.
- It also sent an alert message to the person and suggest some food pattern that are suggest by the Doctors(Diabetes Speacialist).

Model Selection:

In this phase we use various machine learning alogorithm for predict the diabetes.These algorithms like **Random forest Classifier,Logistic Regression and Gradient Boosting.**

Evaluation:

We evaluate the prediction results using various evaluation metrices like accuracy,precision,F1-score,recall.

1. **Accuracy:**

It is the ratio of number of correct prediction to the total number of input samples.It is given as

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

2. **Precision:**

It is the number of correct positive results divided by the number of positive results predicted by the classifier.It is expressed as

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

3. **Recall:**

It is the number of correct positive results divided by the number of all relevant samples.It is expressed as

$$\text{Recall} = \frac{TP}{(TP + FP)}$$

4. **F1-score:**

It is used to measure a test's accuracy.F1 score is the Harmonic Mean between precision and recall.The range for F1-score is[0,1].Mathematically,it given as

$$F1 = 2 * \frac{1}{(1/\text{precision})+(1/\text{recall})}$$

Iterative Improvement:

In this model we using the machine learning algorithm like logistic regression,Random Forest,etc.,so it provide the accuracy result.

AI-Based Diabetes Prediction System

Problem Descrption:

The problem is to bulid an - powered diabetes prediction system that uses machine learning algorithms to analyse the medical data and predict the likelihood an individuals developing diabetes.The system aims to provide early risk assessment and personaliszed preventive measures, allowing individuals to take proactive actions to manage their Health.

Process Involved:

Step 1: Start

Step 2: Upload and read the medical dataset.

Step 3: After uploading the dataset Preprocessing the data.If it is ready for processing, then it divides into two parts as training and testing .

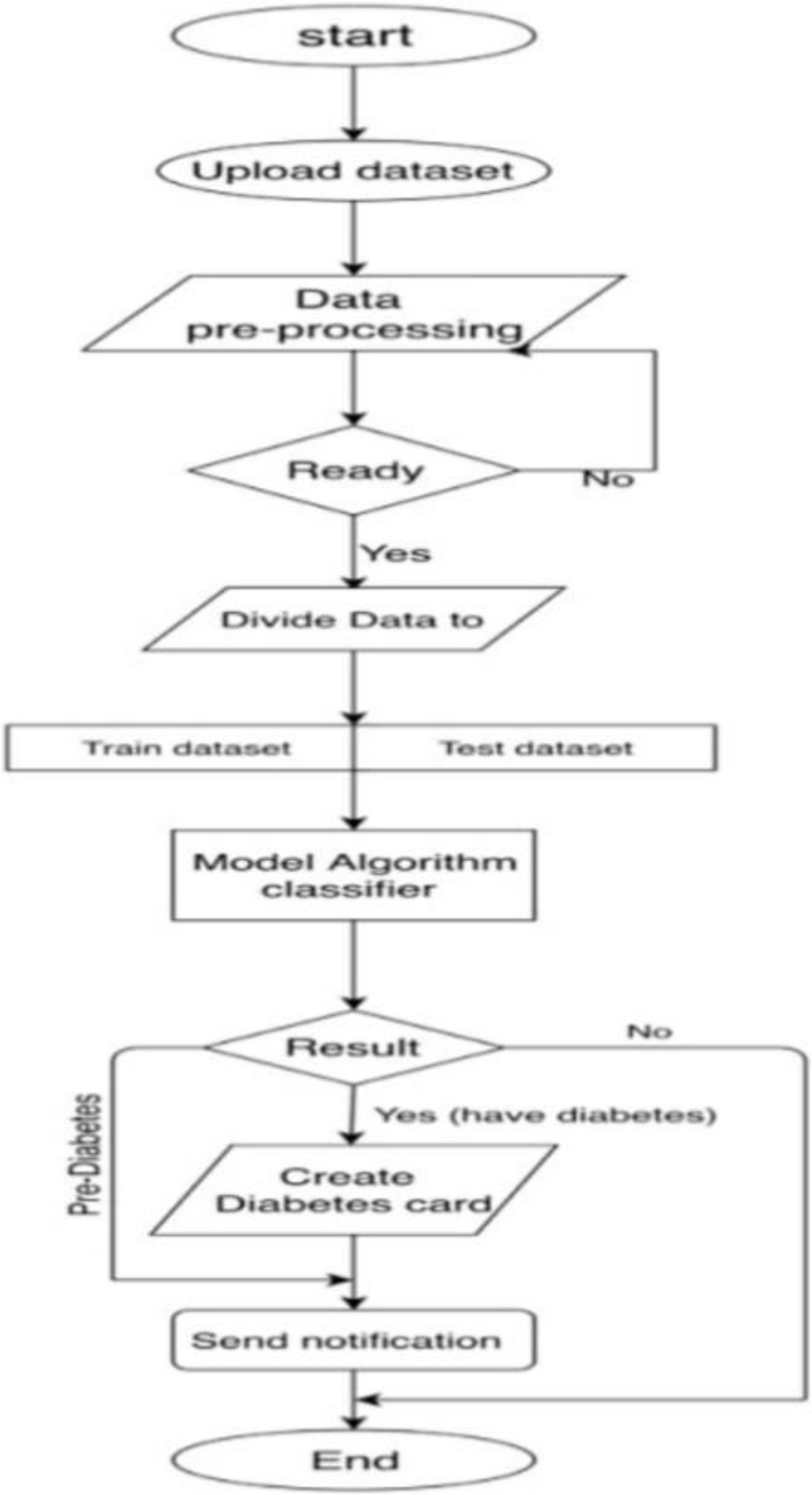
Step 4: If it is not ready, then again preprocess the data.

Step 5: For predicting the result apply the Macihne learning algorithms such as logistic regression, Random forest, Gradient Descent Algorithms.

Step 6: After predicting the result, the user will be notified.

Step 7: If the person have diabetes positive, then it creates the diabetes card and send the notification to the user. Otherwise it stops the process.

Step 8: Stop


start
Upload dataset
Data pre-processing
Ready
No
Yes
Divide Data to
Train dataset
Test dataset
Model Algorithm classifier
Result
No
Yes (have diabetes)
Pre-Diabetes
Create Diabetes card
Send notification
End

AI Based Diabetes Prediction System

Dataset Preprocessing:

In the merged dataset, we discovered a few exceptional zero values. For example, skin thickness and Body Mass Index (BMI) cannot be zero. The zero value has been replaced by its corresponding mean value. The training and test dataset has been separated using the holdout validation technique, where 80% is the training data and 20% is the test data.

Need for Data Preprocessing:

For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore to execute random forest algorithm null values have to be managed from the original raw data set.Another aspect is that the data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithm are executed in one data set, and best out of them is chosen.

Data Preprocessing Steps:

There are four major steps in Data Preprocessing they are:

- Data quality assessment
- Data Cleaning
- Data Transformation
- Data Reduction

1. Data quality assessment:

There are a number of data anomalies and inherent problems to look out for in almost any data set, for example:

- Mismatched data types
- Mixed data values
- Data outliers
- Missing data

2. **Data Cleaning:**

Data cleaning is the process of adding missing data and correcting, repairing, or removing incorrect or irrelevant data from a data set. Dating cleaning is the most important step of preprocessing because it will ensure that your data is ready to go for your downstream needs.

3. **Data Transformation:**

With data cleaning, we've already begun to modify our data, but data transformation will begin the process of turning the data into the proper formats.

4. **Data Reduction:**

The more data you're working with, the harder it will be to analyze, even after cleaning and transforming it. Depending on your task at hand, you may actually have more data than you need. Data reduction not only makes the analysis easier and more accurate, but cuts down on data storage.

Importing Libraries:

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.set()

from mlxtend.plotting import plot_decision_regions
import missingno as msno
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Reading the dataset:

Code:

```
diabetes_df = pd.read_csv('diabetes.csv')
diabetes_df.head()
```

Output:

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

Data Cleaning :

Check the dataset have null values or not:

Code:

```
diabetes_df.isnull().sum()
```

Output:

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

We found no missing values in the dataset, yet independent features like skin thickness, insulin, blood pressure, ;and glucose each have some 0 values, which is practically impossible. A particular column's mean or median scores must be used to replace unwanted 0 values.

Code:

Replacing the value 0 with the mean of that column:

```
data['Glucose'] = data['Glucose'].replace(0, data['Glucose'].median())
data['BloodPressure'] = data['BloodPressure'].replace(0, data['BloodP
ressure'].median())
data['BMI'] = data['BMI'].replace(0, data['BMI'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0, data['SkinT
hickness'] mean())
data['Insulin'] = data['Insulin'].replace(0, data['Insulin'].mean())
data.head()
```

	Pregn - ancies	Glucose	Blood Pressure	Skin Thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome

0	6	148	72	35.000	79.799	33.6	0.627	50	1
1	1	85	66	29.000	79.799	26.6	0.351	31	0
2	8	183	64	20.536	79.799	23.3	0.672	32	1
3	1	89	66	23.000	94.000	28.1	0.167	21	0
4	0	137	40	35.000	168.00	43.1	2.288	33	1

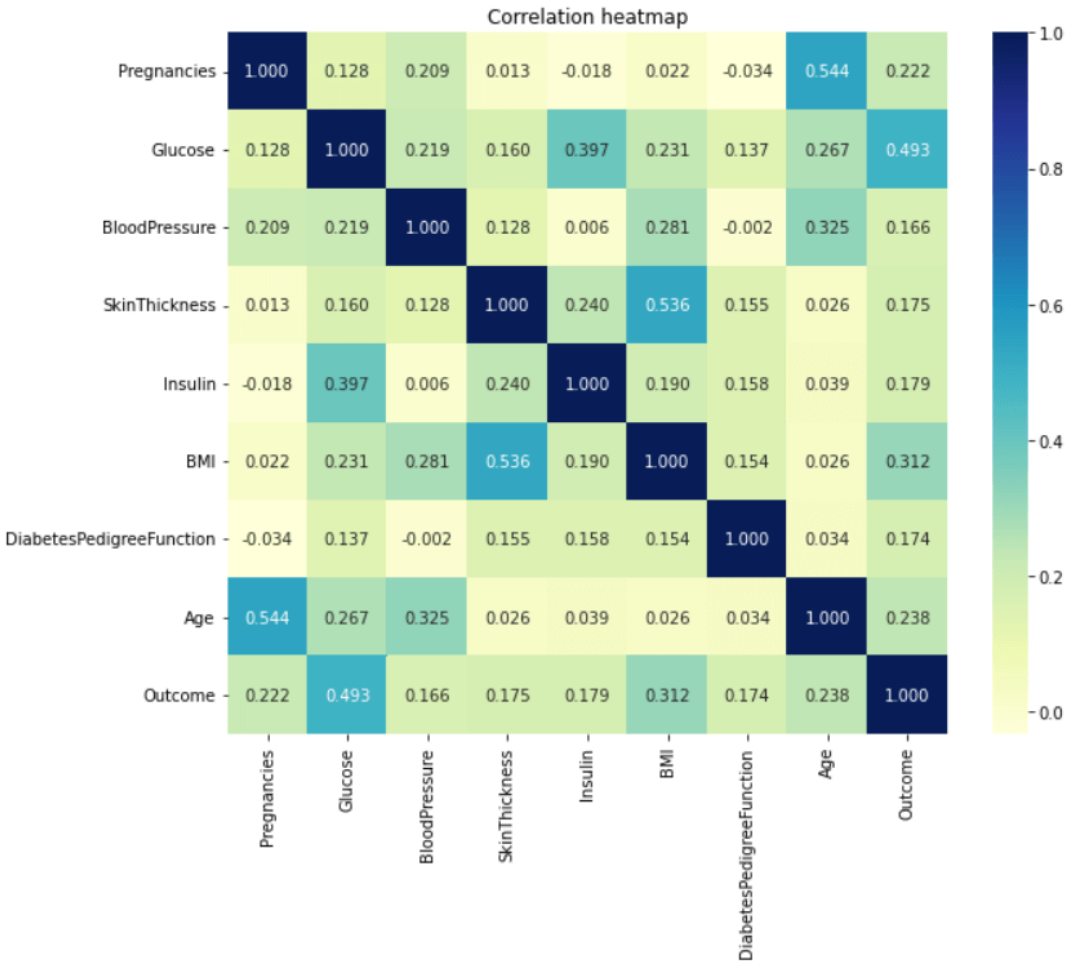
Exploratory Data Analysis :

Correlation

Correlation is the relationship between two or more variables. Finding the important features and cleaning the dataset before we begin modelling also helps make the model efficient.

Code

```
plt.figure(figsize = (10, 8))
sns.heatmap(data.corr(), annot = True, fmt = ".3f", cmap = "Yl
GnBu")
plt.title("Correlation heatmap")
```

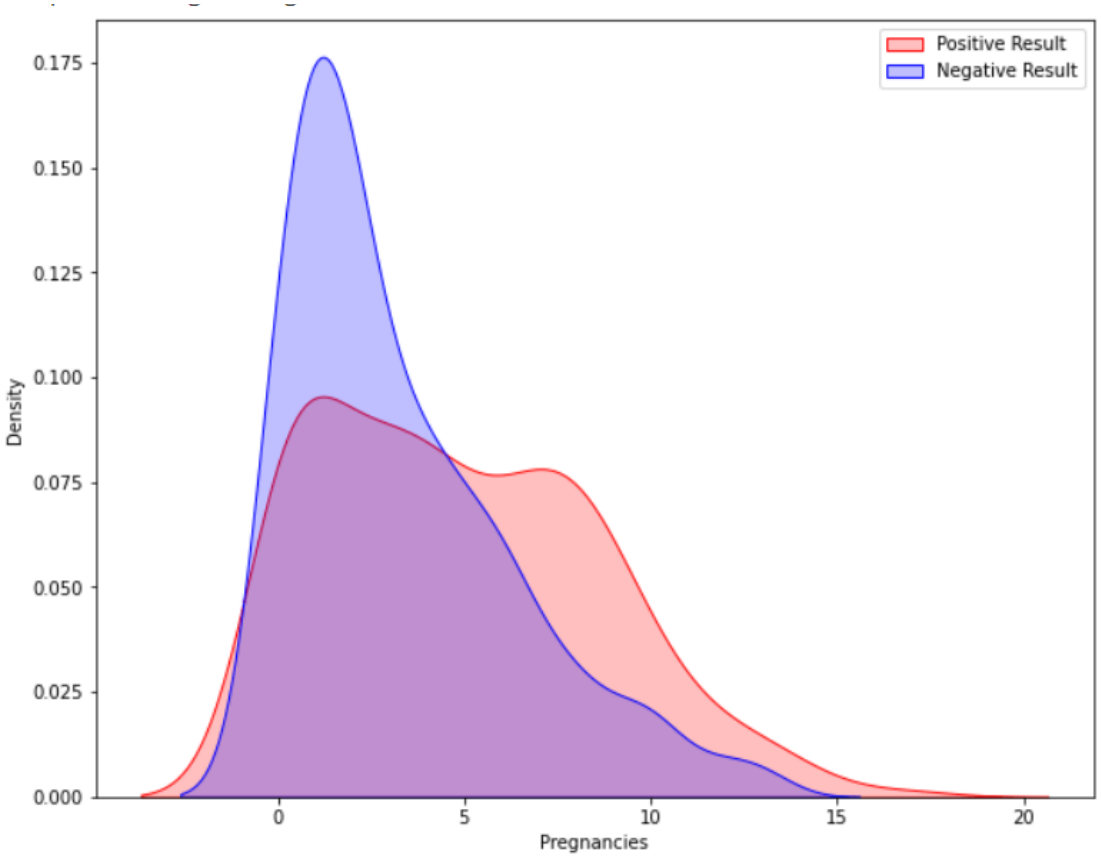


Observation show that characteristic like pregnancy,glucose are more closely associated with the outcomes.Demonstrated a detailed illustration.

Pregnancy:

Code:

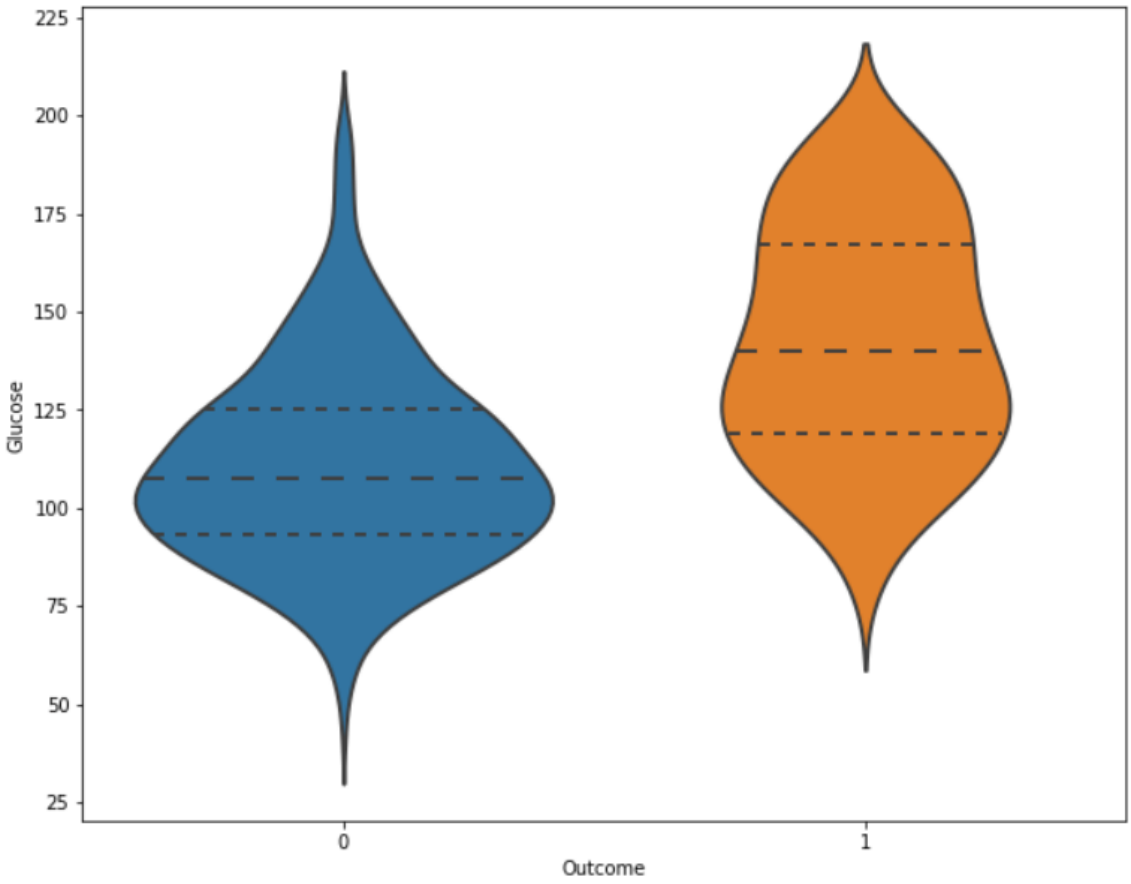
```
plt.figure(figsize = (10, 8))
kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 1], color = "Red" shade = True)
kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 0], ax = kde, color = "Blue", shade= True)
kde.set_xlabel("Pregnancies")
kde.set_ylabel("Density")
kde.legend(["Positive Result", "Negative Result"])
```



Glucose

```
plt.figure(figsize  = (10, 8))
sns.violinplot(data = data, x = "Outcome", y = "Glucose",  split
 = True, inner = "quart", linewidth = 2)
```

Output:


225
200
175
150
125
100
75
50
25
Glucose
0
1
Outcome

AI Based Diabetes Prediction System

For AI based Diabetes Prediction system we use the Logistic Regression machine learning algorithm for implement the code.

Algorithm (Logistic Regression):

```
# === libraries ===
import numpy as np
# === sigmoid === #
description: the S shape function which gives us one or zero.
def sigmoid(x):
        return 1 / (1 + np.exp(-x))
# === Logistic Regression ===
class LogisticRegression():
        def __init__(self, lr=0.001, n_iters=1000):
                self.lr = lr
                self.n_iters = n_iters
                self.weights = None
                self.bias = None
        # X ===> Training inputs samples
        # y ===> Target values
        def fit(self, X, y):
                n_samples, n_features = X.shape
                self.weights = np.zeros(n_features)
```

```
        self.bias = 0
        # this is gradient descent
    for _ in range(self.n_iters):
        linear_pred = np.dot(X, self.weights) + self.bias
        predictions = sigmoid(linear_pred)
        # formula: dw = (1 / n_samples) * X.T * (predictions - y)

        # formula: db = (1 / n_samples) * sum(predictions - y)
        dw = (1 / n_samples) * np.dot(X.T, (predictions - y))
        db = (1 / n_samples) * np.sum(predictions - y)
        self.weights = self.weights - self.lr * dw
        self.bias = self.bias - self.lr * db

    # labeling the data
    def predict(self, X):
        linear_pred = np.dot(X, self.weights) + self.bias
        y_pred = sigmoid(linear_pred)
        class_pred = [0 if y <= 0.5 else 1 for y in y_pred]
        return class_pred
```

Training the Model:

Code:

```
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from logisticRegression import LogisticRegression
import pandas as pd
```

Code:

```
learn_d = pd.read_csv("diabetes.csv")
X = learn_d.drop('Outcome', axis=1)
# Assuming 'Outcome' is the diabetes label
y = learn_d['Outcome']
print(X)
```

Output :

	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	

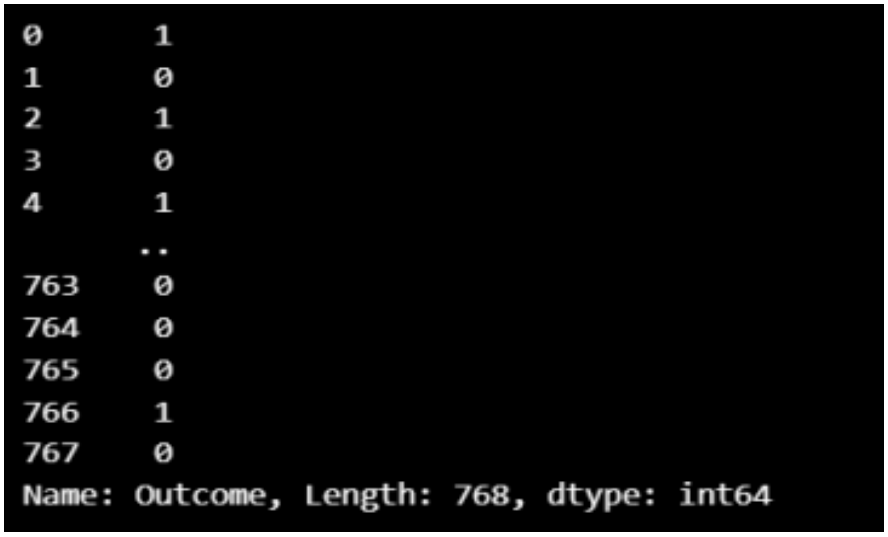
	DiabetesPedigreeFunction	Age
0	0.627	50
1	0.351	31
2	0.672	32
3	0.167	21
4	2.288	33
..	...	...
763	0.171	63
764	0.340	27
765	0.245	30
766	0.349	47
767	0.315	23

[768 rows x 8 columns]

Code:

```
print(y)
```

Output:



Code:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
print(X_test)
```

Output :

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
668            6       98             58             33      190  34.0
324            2      112             75             32        0  35.7
624            2      108             64              0        0  30.8
690            8      107             80              0        0  24.6
473            7      136             90              0        0  29.9
..           ...      ...            ...            ...      ...   ...
355            9      165             88              0        0  30.4
534            1       77             56             30       56  33.3
344            8       95             72              0        0  36.8
296            2      146             70             38      360  28.0
462            8       74             70             40       49  35.3

     DiabetesPedigreeFunction  Age
668                     0.430   43
324                     0.148   21
624                     0.158   21
690                     0.856   34
473                     0.210   50
..                        ...  ...
355                     0.302   49
534                     1.251   24
344                     0.485   57
296                     0.337   29
462                     0.705   39

[154 rows x 8 columns]
```

Code:

```
LR = LogisticRegression(lr = 0.0001, n_iters= 100000)
LR.fit(X_train, y_train)

Y_pred = LR.predict(X_test)
print(Y_prediction)
```

Output :

```
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
<logisticRegression.LogisticRegression object at 0x000001DA26374690>
```

Code:

```
def accuracy(y_pred, y_test):
    return np.sum(y_pred == y_test)/len(y_test)


# result = accuracy(Y_prediction, Y_test)
# print(result)


# === main ===
def new_func():
    option = int(input("select the option:"))
    return option
while(True):
    print("Select an option: \n 1) Evaluation\n 2) Give input\n 3) Exit Program")
    option = new_func()


    if(option == 1):
        acc = accuracy(y_test, Y_pred)
        print("Accuracy is:",acc)
    elif(option == 2):
```

```
        pregnancies = float(input("Please enter number of
        pregnancy you had: "))

        glucose = float(input("Please enter your glucose rate ==>
        mg/dl: "))

        bloodPressure = float(input("Please enter your blood
        pressure ==> mm/Hg: "))

        skinThickness = float(input("Please enter thickness of your
        skin ==> (0,99): "))

        insulin = float(input("Please enter insulin level of your
        blood ==> mm: "))

        bmi = float(input("Please enter you BMI: "))

        diabetesPedigreeFunction = float(input("Please enter
        Diabetes pedigree function: ")) age = float(input("Please
        enter your age: "))

        x_input = [[pregnancies, glucose, bloodPressure,
        skinThickness, insulin, bmi, diabetesPedigreeFunction,
        age]]

        prob = LR.predict(x_input)

        print("Outcome: ", prob[0])


elif(option == 3):

        print("exit")

        break
```

Output :

```
Select an option:
 1) Evaluation
 2) Give input
 3) Exit Program
1
Accuracy is: 0.7272727272727273
Select an option:
 1) Evaluation
 2) Give input
 3) Exit Program
2
Please enter number of pregnancy you had: 2
Please enter your glucose rate ==> mg/dl: 197
Please enter your blood pressure ==> mm/Hg: 70
Please enter thickness of your skin ==> (0,99): 45
Please enter insulin level of your blood ==> mm: 543
Please enter you BMI: 30.5
Please enter Diabetes pedigree function: 1.658
Please enter your age: 54
Outcome:  1
```