# *AI BASED DIABETES PRIDICTION SYSTEM*

## *TEAM MEMBERS*

***AU922321104032 :M.Ragavi***

***AU922321104040 :X.Simifemina***

***AU922321104035 :S.Santhiya devi***

## INTRODUCTION :

**Artificial Intelligence (AI)-based diabetes prediction is a**

**transformative approach to healthcare that leverages the power of machine learning and data analysis to foresee the risk of diabetes in individuals. Diabetes, a chronic and increasingly prevalent metabolic disorder, poses significant health challenges worldwide. Early detection and prevention are crucial in mitigating its impact, reducing healthcare costs, and improving the quality of life for affected individuals.**

**AI-based diabetes prediction systems utilize advanced algorithms and vast datasets encompassing a range of health and lifestyle factors to assess the probability of an individual developing diabetes. These systems can be instrumental in providing healthcare practitioners and patients with valuable insights and recommendations, enabling timely intervention and lifestyle adjustments to prevent or manage the condition effectively.**

**This innovative approach to diabetes prediction offers numerous advantages. It can enhance the accuracy of risk assessment by analyzing complex interactions between various factors, including genetics, age, obesity, physical activity, dietary habits, and medical history. AI models are capable of uncovering intricate patterns and relationships in the data, often beyond the scope of conventional statistical methods.**

**Moreover, AI-based diabetes prediction systems are adaptable and can continuously evolve as new data becomes available, ensuring that predictions remain up to date and relevant. By harnessing this technology, healthcare providers can offer personalized guidance and treatment plans to individuals at risk, potentially reducing the prevalence of diabetes and its associated**

**complications.**

**The application of AI in diabetes prediction is an exemplar of the evolving landscape of personalized and data-driven healthcare. As the field of AI continues to advance, the potential for more accurate and effective predictions, coupled with proactive healthcare strategies, holds promise for a future with reduced diabetes-related health burdens and improved overall well-being. This introduction sets the stage for a deeper exploration of the methodologies and challenges involved in developing AI-based diabetes prediction systems.**

## Problem Statement:

**The problem is to develop a diabetes prediction system that can accurately identify individuals at risk of developing diabetes based on their health and lifestyle data. Diabetes is a widespread and growing health concern, and early prediction can help individuals take preventive measures and seek medical advice, ultimately reducing the burden of the disease.**

## Design Thinking Process:

### 1. Empathize:

**- Understand the target audience, including potential users and healthcare**

**professionals.**

**- Conduct interviews and surveys to gather insights into their needs and challenges related to diabetes prediction.**

**- Collect relevant data sources, such as electronic health records, lifestyle data, and medical history.**

### 2. Define:

**- Define the specific problem and the scope of the project, considering the data available and the end-users' requirements.**

**- Create user personas and use cases to understand who will use the prediction system and how.**

### 3. Ideate:

**- Brainstorm potential solutions, such as machine learning models, data preprocessing techniques, and user interfaces.**

**- Explore various data sources, algorithms, and features that can be leveraged for prediction.**

**- Identify key metrics for evaluating model performance, like accuracy, precision, recall, and F1-score.**

### 4. Prototype:

**- Develop a prototype of the diabetes prediction system, which includes the**

**user interface for data input and the model for making predictions.**

**- Create a user-friendly interface for inputting health and lifestyle data, and visualize the prediction results.**

**- Implement a machine learning model, such as logistic regression, decision trees, or deep learning, to predict diabetes risk.**

## 5. Test:

**- Evaluate the prototype with potential users, and gather feedback to refine the system.**

**- Conduct validation and testing of the machine learning model to assess its accuracy and generalization.**

**- Ensure that the system complies with data privacy and security regulations, such as GDPR or HIPAA.**

## 6. Implement:

**- Develop the full-fledged diabetes prediction system based on the prototype, incorporating user feedback and improvements.**

**- Integrate the system with existing healthcare infrastructure and databases, if necessary.**

**- Ensure scalability and reliability of the system.**

### 7. Launch:

**- Deploy the diabetes prediction system in a real-world environment, either in a clinical setting or as a mobile/web application.**

**- Educate healthcare professionals and potential users about the system's capabilities and benefits.**

**- Monitor the system's performance and address any issues that arise during the initial deployment.**

### 8. Iterate:

**- Continuously gather user feedback and monitor the system's performance to make improvements.**

**- Enhance the machine learning model as more data becomes available and new research findings emerge.**

**- Stay up to date with the latest advancements in diabetes prediction and healthcare technology.**

## DESCRIBE ABOUT THE DATASET :

**The datasets consists of several medical predictor variables and one Target details variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.**

**Pregnancies:** **Number of times pregnant**

**Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test**

**BloodPressure: Diastolic blood pressure (mm Hg)**

**SkinThickness: Triceps skin fold thickness (mm)**

**Insulin: 2-Hour serum insulin (mu U/ml)**

**BMI: Body mass index (weight in kg/(height in m)^2)**

**DiabetesPedigreeFunction: Diabetes pedigree function**

**Age: Age (years)**

**Outcome: Class variable (0 or 1)**

**Number of Observation Units: 768**

**Variable Number： 9**

## Phases of Development for Diabetes Prediction:

### 1. Data Collection:

**- Gather relevant health records, lifestyle data, and medical history from individuals.**

**- Curate and preprocess the data to make it suitable for machine learning.**

### 2. Feature Engineering:

**- Select and extract informative features from the data that may be indicative of diabetes risk.**

**- Normalize and transform features as needed.**

**3. Model Development:**

**- Choose appropriate machine learning algorithms, train, and fine-tune the model using historical data.**

**- Optimize hyperparameters to improve model performance.**

**4. Validation and Testing:**

**- Split the dataset into training, validation, and testing sets.**

**- Assess the model's performance using metrics like accuracy, precision, recall, and F1-score.**

**- Validate the model's generalization on new, unseen data.**

**5. Deployment:**

**- Integrate the trained model into the diabetes prediction system.**

**- Develop a user-friendly interface for data input and results visualization.**

**6. Monitoring and Maintenance:**

**- Continuously monitor the system's performance and accuracy.**

**- Address issues, bugs, and security concerns as they arise.**

**- Keep the model up-to-date with new data and research.**

### 7. User Education and Support:

**- Provide training and support to healthcare professionals and end-users.**

**- Educate users on how to interpret and act on the system's predictions.**

### 8. Ongoing Improvement:

**- Collect user feedback and data to iteratively enhance the system.**

**- Incorporate new research findings and technological advancements for better prediction accuracy and user experience.**

**The development of a diabetes prediction system should follow these phases, incorporating the principles of design thinking to create a user-centered and effective solution.**

## 1. Dataset Description:

**- The dataset used for diabetes prediction typically consists of health-related and lifestyle data collected from individuals. It may include features such as age, gender, BMI (Body Mass Index), blood pressure, cholesterol levels, family history of diabetes, physical activity, and dietary habits.**

**- The target variable is typically a binary label indicating whether an individual has diabetes (1) or does not have diabetes (0).**

**Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.**

**Number of Instances: 768**
**Number of Attributes: 8 plus class**
**For Each Attribute: (all numeric-valued)**

- **Pregnancies: Number of times pregnant**
- **Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test**
- **BloodPressure: Diastolic blood pressure (mm Hg)**
- **SkinThickness: Triceps skin fold thickness (mm)**
- **Insulin: 2-Hour serum insulin (mu U/ml)**
- **BMI: Body mass index (weight in kg/(height in m)^2)**
- **DiabetesPedigreeFunction: Diabetes pedigree function**
- **Age: Age (years)**

**Outcome:** Class variable (0 or 1)

# 1. Import the important Libraries

```
In [1]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statistics import mean, stdev
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score,
mean_squared_error
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from lightgbm import LGBMClassifier
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=UserWarning)
```

## 2. Loading the Dataset

```
In [2]:
diabetes = pd.read_csv("/kaggle/input/diabetes-data-set/diabetes.csv")
```

## 3. Inspecting the Dataset

```
In [3]:
diabetes.head()
Out[3]:
```

| | 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 |

```
In [4]:
diabetes.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [5]:

```
diabetes.describe()
```

Out[5]:

| 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 |

## 2. Data Preprocessing Steps:

Data preprocessing involves cleaning and preparing the dataset for use in a machine learning model. Some common steps for data preprocessing in diabetes prediction include:

### a. Handling Missing Values:

- Check for missing values in the dataset and decide on a strategy to deal with them. Common approaches include imputation (e.g., filling missing values with the mean or median) or removing rows or columns with a significant number of missing values.

### b. Data Scaling and Normalization:

- Features like age, BMI, and blood pressure may have different scales. Scaling and normalizing the features to a standard range (e.g., between 0 and 1) can help prevent certain features from dominating the model training process.

### c. Handling Categorical Data:

- If the dataset contains categorical features (e.g., gender), they need to be converted into numerical representations, such as one-hot encoding or

**label encoding.**

### d. Outlier Detection and Treatment:

**- Identify and handle outliers in the data, as extreme values can affect the performance of the machine learning model. You can use statistical techniques like Z-score or IQR (Interquartile Range) to detect and deal with outliers.**

### e. Feature Engineering:

**- Create new features or transformations that might be more informative. For instance, you could calculate a person's age group (e.g., young, middle-aged, senior) from their birthdate.**

### f. Splitting the Dataset:

**- Divide the dataset into training, validation, and testing sets to assess model performance. The most common split ratio is 70-15-15 (or 80-10-10), where 70% (or 80%) is used for training, and the rest for validation and testing.**

**By inspection, dataset has no missing values. Let's check out for Duplicates and Outliers.**

```
In [6]:
diabetes.drop_duplicates()
```

```
Out[6]:
```

| | 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 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 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 |

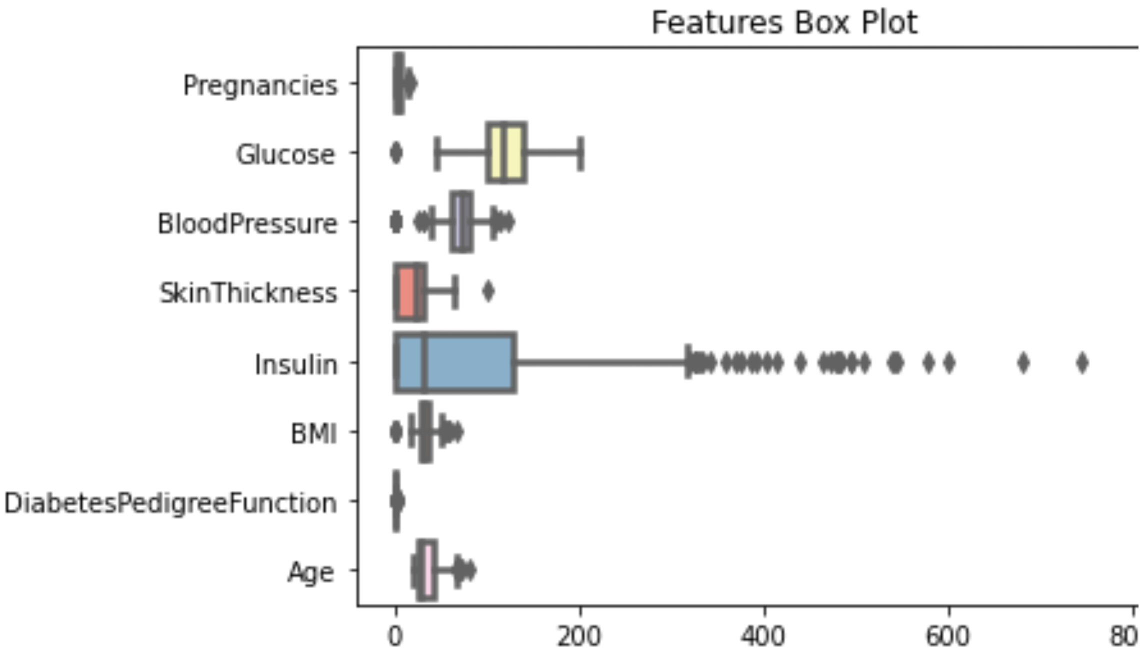
768 rows × 9 columns

### *4.1 Outliers*

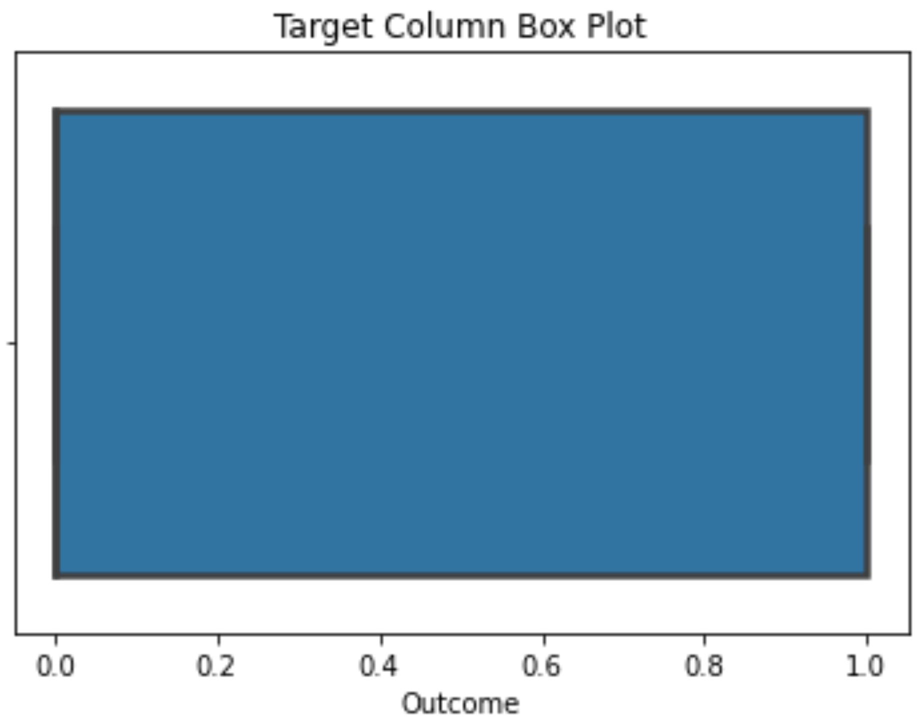
Checking for outliers using the box plot.

```
In [7]:
linkcode
#first store the features in a seperate dataframe.
features = diabetes.drop("Outcome",axis = 1).copy()
#Now plot a boxplot to identify the outliers in our features.
sns.boxplot(data = features, orient = 'h', palette = 'Set3', linewidth = 2.5 )
plt.title("Features Box Plot")
```

```
Out[7]:
Text(0.5, 1.0, 'Features Box Plot')
```



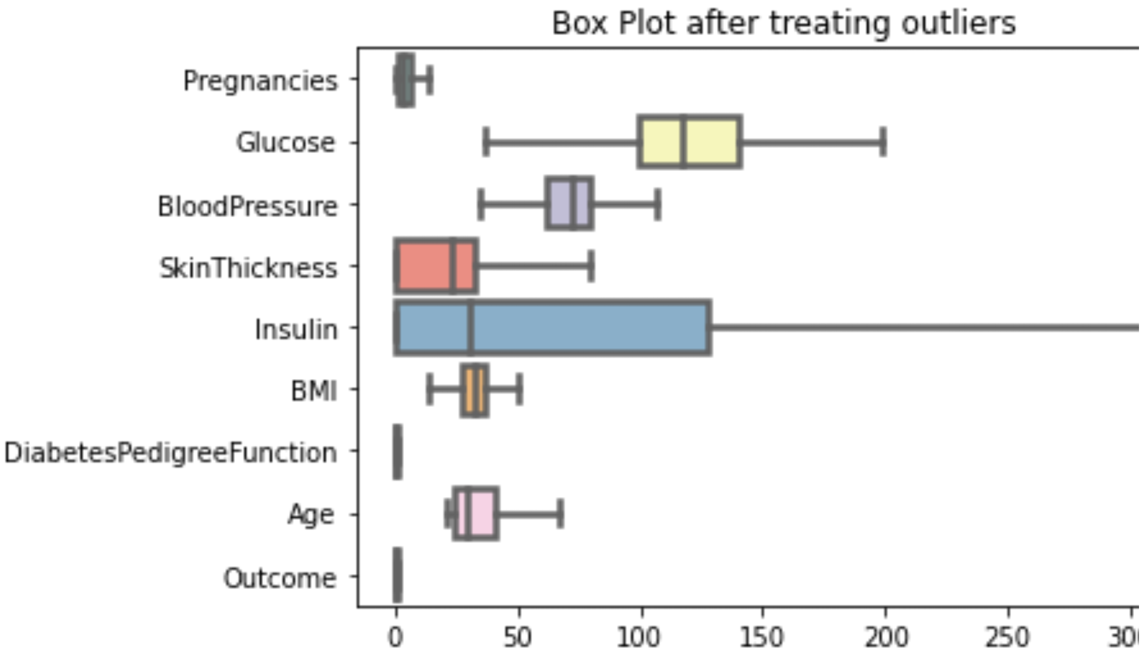
```
sns.boxplot(x = diabetes["Outcome"], orient = 'h', linewidth = 2.5 )
plt.title("Target Column Box Plot")
plt.show()
```



Our target column does not contain outliers. Now let's treat our feature columns for outliers.

```
In [9]:
from scipy import stats
def removeoutliers(df=None, columns=None):
    for column in columns:
        Q1 = df[column].quantile(0.25)
        Q3 = df[column].quantile(0.75)
        IQR = Q3 - Q1
        floor, ceil = Q1 - 1.5 * IQR, Q3 + 1.5 * IQR
        df[column] = df[column].clip(floor, ceil)
        print(f"The columnn: {column}, has been treated for outliers.\n")
    return df
diabetes = removeoutliers(diabetes,[col for col in features.columns])
The columnn: Pregnancies, has been treated for outliers.

The columnn: Glucose, has been treated for outliers.

The columnn: BloodPressure, has been treated for outliers.

The columnn: SkinThickness, has been treated for outliers.

The columnn: Insulin, has been treated for outliers.

The columnn: BMI, has been treated for outliers.

The columnn: DiabetesPedigreeFunction, has been treated for outliers.

The columnn: Age, has been treated for outliers.
```

```
In [10]:
linkcode
sns.boxplot(data = diabetes, orient = 'h', palette = 'Set3', linewidth = 2.5 )
plt.title("Box Plot after treating outliers")
Out[10]:
Text(0.5, 1.0, 'Box Plot after treating outliers')
```



## 3. Feature Selection Techniques:

**Feature selection is the process of choosing the most relevant and informative features for building the prediction model. It helps reduce model complexity and overfitting. Common feature selection techniques for diabetes prediction include:**

### a. Univariate Feature Selection:

**- Univariate feature selection methods like chi-squared tests or ANOVA can help identify features with the strongest statistical relationship with the target variable.**

### b. Recursive Feature Elimination (RFE):

**- RFE is an iterative technique that starts with all features and recursively removes the least important features, based on a chosen model's feature importance score.**

### c. Feature Importance from Tree-Based Models:

**- Tree-based models (e.g., Random Forest or XGBoost) can provide feature importance scores. Features with higher importance can be selected for the final model.**

### d. L1 Regularization (Lasso):

**- L1 regularization can be applied in linear models to automatically select important features by driving some feature coefficients to zero.**

### e. Correlation Analysis:

**- Calculate the correlation between features and the target variable. Features with higher correlation can be considered more important.**

### f. Principal Component Analysis (PCA):

**- PCA is a dimensionality reduction technique that can be used to create new features that are linear combinations of the original features, potentially reducing the number of features while preserving information.**

## CHOICE OF MACHINE LEARNING ALGORITHM,MODEL TRAINING AND EVALUATION :

### MACHINE LEARNING ALGORITHM :LOGISTIC REGRESSION

# Importing the Required Libraries

```
In [1]:
```

```
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk
import missingno as msn
import seaborn as sns
```

## Reading the Dataset

**Gather a dataset that includes features (independent variables) and target labels. For diabetes prediction, the target labels would be binary (0 for non-diabetic, 1 for diabetic).**

```
In [2]:
```

```
df=pd.read_csv(r"../input/diabetes-data-set/diabetes.csv")
```

## Data Preprocessing

**This step involves cleaning and preparing your data. This may include handling missing values, normalizing or standardizing features,**

**and splitting the data into training and testing sets for model evaluation.**

```
In [3]:
```

```
df.head()
Out[3]:
```

| | 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 |

```
In [4]:
```

```
df.tail()
Out[4]:
```

| | 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 |

In [5]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [6]:

```
df.corr()
```

Out[6]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 | -0.073535 | 0.017683 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 | 0.331357 | 0.221071 | 0.137337 | 0.263514 | 0.466581 |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 | 0.088933 | 0.281805 | 0.041265 | 0.239528 | 0.065068 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 | 0.436783 | 0.392573 | 0.183928 | -0.113970 | 0.074752 |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 | 1.000000 | 0.197859 | 0.185071 | -0.042163 | 0.130548 |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 | 0.197859 | 1.000000 | 0.140647 | 0.036242 | 0.292695 |
| DiabetesPedigreeFunction | -0.033523 | 0.137337 | 0.041265 | 0.183928 | 0.185071 | 0.140647 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 | -0.042163 | 0.036242 | 0.033561 | 1.000000 | 0.238356 |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 | 0.130548 | 0.292695 | 0.173844 | 0.238356 | 1.000000 |

In [7]:

```
df.columns
```

Out[7]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [8]:

```
df.dtypes
```

Out[8]:

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

## Defining X and y as independent and target variable

In [9]:

```
X=df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']]
y=df['Outcome']
```

## Splitting the dataset into training and testing

**Use the training dataset to fit a logistic regression model. The model will learn the relationship between the input features and the likelihood of a person having diabetes.**

In [10]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=
1)
```

## Logistic Regression

In [11]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(X_train,y_train)
```

Out[11]:

```
LogisticRegression()
```

In [12]:

```
y_pre=model.predict(X_test)
```

## Model Evaluation

**Assess the model's performance using various metrics like accuracy, precision, recall, F1-score, and ROC-AUC on the testing dataset. You can also use techniques like cross-validation to get a more robust assessment of your model's performance.**

In [13]:

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
accuracy_score(y_test,y_pre)
```

Out[13]:

```
0.7835497835497836
```

In [14]:

```
confusion_matrix(y_test,y_pre)
```

Out[14]:

```
array([[132,  14],
       [ 36,  49]])
```

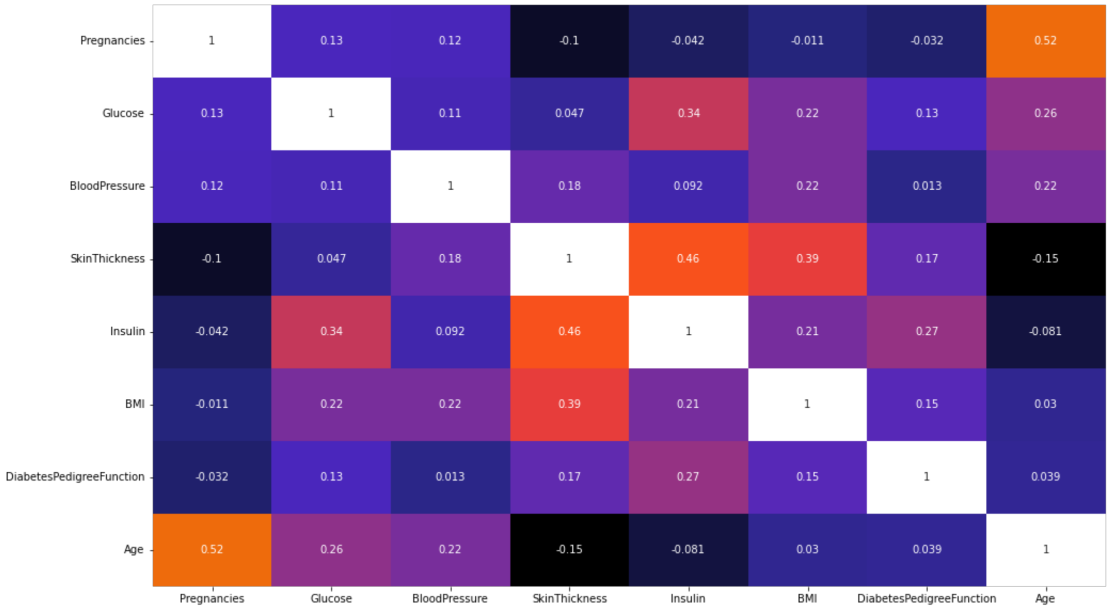
In [15]:

```
print(classification_report(y_test,y_pre))
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       146
           1       0.78      0.58      0.66        85

    accuracy                           0.78       231
   macro avg       0.78      0.74      0.75       231
weighted avg       0.78      0.78      0.78       231
```

In [16]:

linkcode

```
import seaborn as sns
plt.figure(figsize=(20,10))
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap)
plt.show();
```



| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age |
|---|---|---|---|---|---|---|---|---|
| Pregnancies | 1 | 0.13 | 0.12 | -0.1 | -0.042 | -0.011 | -0.032 | 0.52 |
| Glucose | 0.13 | 1 | 0.11 | 0.047 | 0.34 | 0.22 | 0.13 | 0.26 |
| BloodPressure | 0.12 | 0.11 | 1 | 0.18 | 0.092 | 0.22 | 0.013 | 0.22 |
| SkinThickness | -0.1 | 0.047 | 0.18 | 1 | 0.46 | 0.39 | 0.17 | -0.15 |
| Insulin | -0.042 | 0.34 | 0.092 | 0.46 | 1 | 0.21 | 0.27 | -0.081 |
| BMI | -0.011 | 0.22 | 0.22 | 0.39 | 0.21 | 1 | 0.15 | 0.03 |
| DiabetesPedigreeFunction | -0.032 | 0.13 | 0.013 | 0.17 | 0.27 | 0.15 | 1 | 0.039 |
| Age | 0.52 | 0.26 | 0.22 | -0.15 | -0.081 | 0.03 | 0.039 | 1 |

## Feature Selection:

**Identify which features contribute most to the prediction. You can use statistical tests, feature importance scores, or domain knowledge to select the most relevant features.**

## Model Optimization:

**You can fine-tune the logistic regression model by adjusting hyperparameters or exploring regularization techniques (e.g., L1 or L2 regularization) to prevent overfitting.**

## Deployment:

**Once you are satisfied with the model's performance, you can deploy it for real-world diabetes prediction. This might involve creating a user-friendly interface for healthcare professionals or patients to input their data and get predictions.**

### Monitoring:

Continuous monitoring of your model's performance in a production environment is essential to ensure its accuracy and reliability over time.

## CONCLUSION :

A conclusion for an AI-based diabetes prediction system should summarize the key points, findings, and implications of the project. Here's a sample conclusion In conclusion, the development of an AI-based diabetes prediction system holds great promise in healthcare. The project aimed to harness the power of artificial intelligence to accurately forecast the likelihood of an individual developing diabetes based on their health data and risk factors.