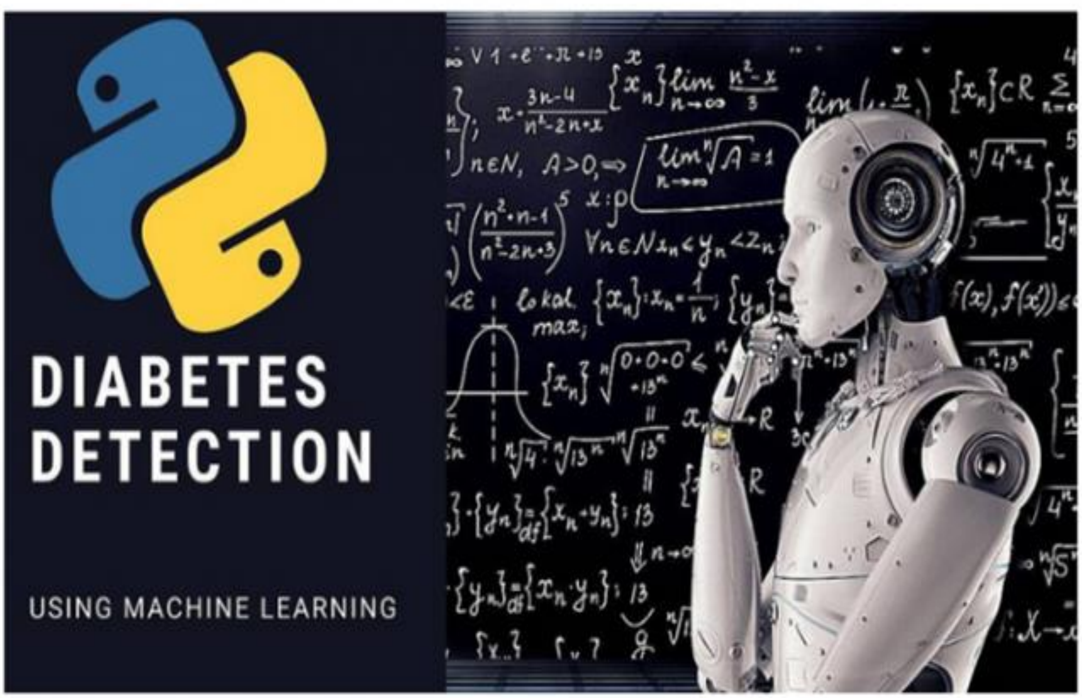
AI BASED DIABETES PREDICTION SYSTEM

TEAM MEMBERS

AU922321104032 :M.RAGAVI
AU922321104040 :X.SIMI FEMINA
AU922321104035 :S.SANTHIYA DEVI

PHASE 4 :SELECTING,LOADING AND EVALUATING THE LOGISTIC ALGORITHM USING MACHINE LEARNING



INTRODUCTION :

Creating a diabetes prediction model using logistic regression algorithm involves several steps.Below we will provide a comprehensive outline of the process,including code examples using python and popular libraries like scikit learn,pandas etc..

LOGISTIC REGRESSION ALGORITHM :

Logistic regression is a statistical model used for binary classification tasks, where the goal is to predict the probability that an input belongs to one of two possible classes. It's called "logistic" because it uses the logistic function to model the relationship between the input features and the binary outcome.

Here's a step-by-step explanation of logistic regression with an example:

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 :

Step 1:

Data Collection :

You collect a dataset that includes past student records, including GPA, test scores, and admission status.

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

Data Preprocessing :

You preprocess and clean the data, ensuring it's in a suitable format for modeling. This may include dealing with missing values, normalizing features, and splitting the data into training and testing sets.

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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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
```

Training the Model :

You use the training data to find the values of the coefficients that minimize a cost function, which measures the difference between the predicted probabilities and the actual outcomes. This is typically done using optimization algorithms such as gradient descent.

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

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 :

You evaluate the model's performance using various metrics, such as accuracy, precision, recall, and F1 score, on a separate testing dataset. This step helps you assess how well the model generalizes to new data.

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]:

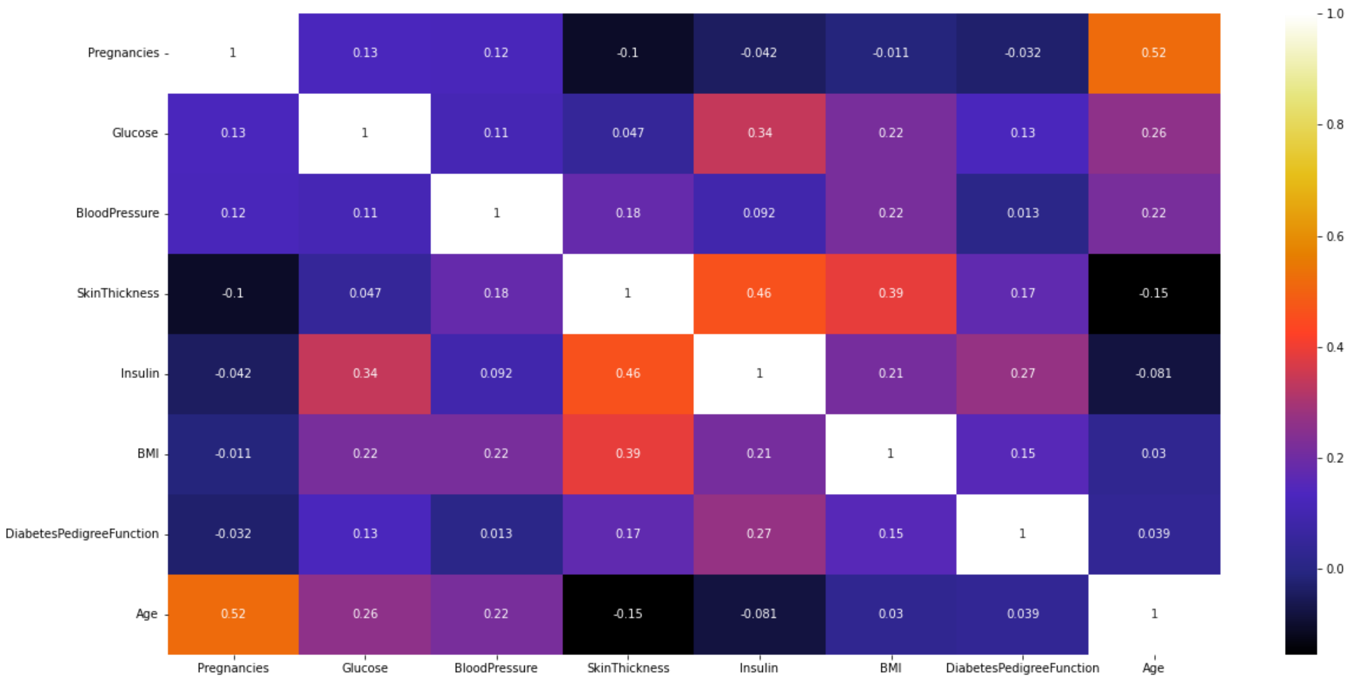
```
print(classification_report(y_test,y_pre))
```

```
              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]:

```
import seaborn as sns
plt.figure(figsize=(20,10))
cor = X_train.corr()
```

```
sns.heatmap(cor, annot=True, cmap=plt.cm.CMRmap)
plt.show();
```



CONCLUSION :

In conclusion,This code provides a basic example of using logistic regression for diabetes prediction. Logistic regression is a widely used and interpretable method for binary classification tasks, and it's especially useful when you want to understand the relationship between your input features and the binary outcome.