**NAME OF THE COURSE : ARTIFICIAL INTELLIGENCE**

**NAME OF THE PROJECT : AI BASED DIABETES PREDICTION SYSTEM**

**PHASE – 3**

*TEAM MEMBER : KRISHNAKANTH S*

**INTRODUCTION:**

 Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Why preprocessing?

Real-world data are generally:

* **Incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
* **Noisy**: containing errors or outliers
* **Inconsistent**: containing discrepancies in codes or names

Tasks in data preprocessing:

* **Data cleaning**: fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies.
* **Data integration**: using multiple databases, data cubes, or files.
* **Data transformation**: normalization and aggregation.
* **Data reduction**: reducing the volume but producing the same or similar analytical results.
* **Data discretization**: part of data reduction, replacing numerical attributes with nominal ones.

#### What is Exploratory data analysis?

In statistics, exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

**DATA CLEANING**

Data cleaning is the process of cleaning / standardising the data to make it ready for analysis. Most of times, there will be discrepancies in the captured data such as incorrect data formats, missing data, errors while capturing the data. This is an important step in any given data science project because the accuracy of the results depends heavily on the data we use.

**DATA INTEGERATION:**

Data integration is the process of combining data from multiple sources into a unified view. This can be done for a variety of reasons, such as:

* To improve data quality and consistency
* To gain insights from multiple data sources
* To avoid data duplication
* To improve data accessibility and governance

**DATA TRAFORMATION:**

Data transformation is the process of converting raw data into a format that is suitable for analysis. This may involve cleaning the data, converting data types, and merging data from multiple sources.

Normalization and aggregation are two common data transformation techniques.

Normalization is the process of transforming the values of a dataset into a common scale. This is important to ensure that all of the features in the dataset are treated equally by machine learning algorithms.

**PROGRAMS**

**DATA CLEANING:**

*#importing libraries*

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

*#read dataset*

df=pd.read\_csv('../input/diabetes

-data-set/diabetes.csv')

### EDA

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.columns

Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],

dtype='object')

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

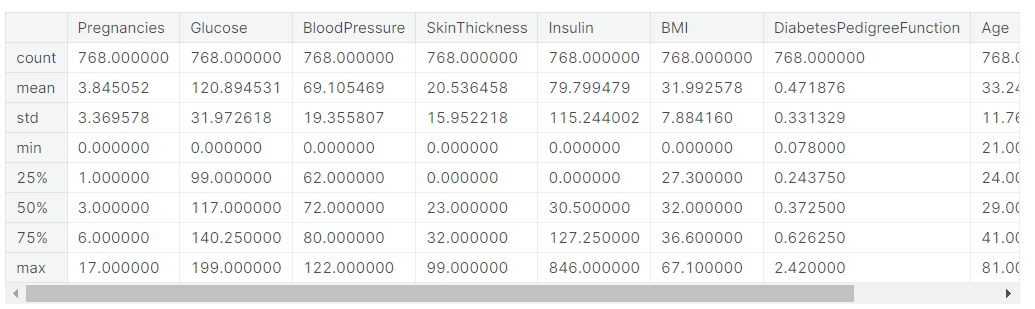
df.shape

(768, 9)

In [7]:

linkcode

df.describe()



*#checking null values*

df.isnull().sum()

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

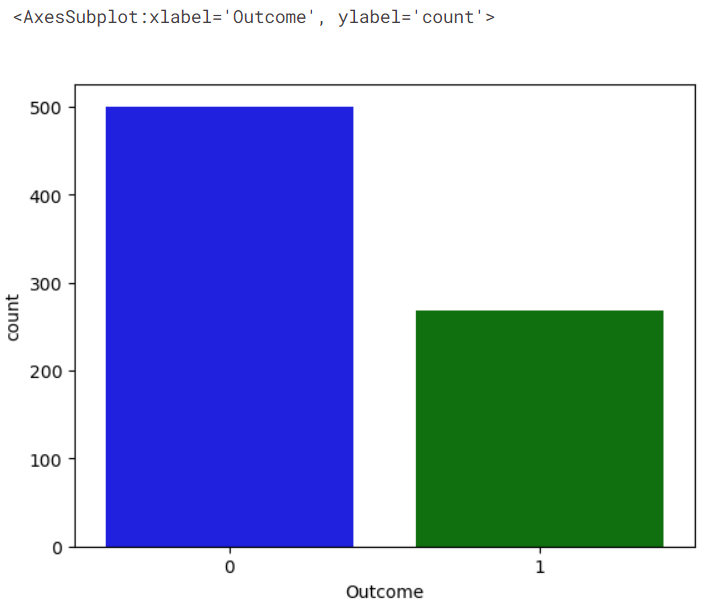
Age 0

Outcome 0

dtype: int64

*#countplot*

sns.countplot(x='Outcome',data=df,palette=['b','g'])



*#pairplot*

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

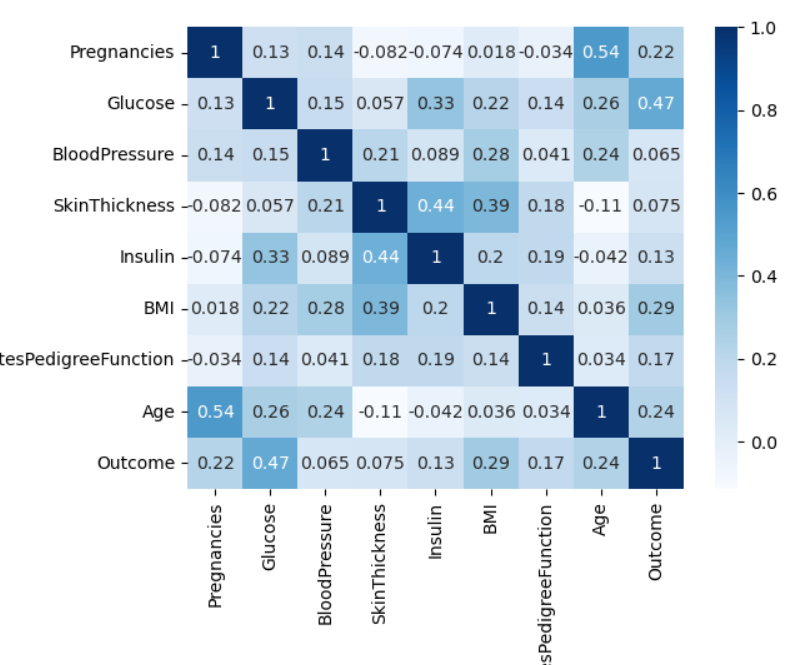
plt.show()

****

*#correlation heatmap*

sns.heatmap(df.corr(),annot=True,cmap='Blues')

plt.show()



*#replacing zero value with NaN*

df\_new=df

df\_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]] = df\_new[["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]].replace(0, np.NaN)

In [13]:

*#count of Nan*

df\_new.isnull().sum()

Out[13]:

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

*#replacing NaN with mean values*

df\_new["Glucose"].fillna(df\_new["Glucose"].mean(), inplace = True)

df\_new["BloodPressure"].fillna(df\_new["BloodPressure"].mean(), inplace = True)

df\_new["SkinThickness"].fillna(df\_new["SkinThickness"].mean(), inplace = True)

df\_new["Insulin"].fillna(df\_new["Insulin"].mean(), inplace = True)

df\_new["BMI"].fillna(df\_new["BMI"].mean(), inplace = True)

In [15]:

*#checking null values*

df\_new.isnull().sum()

Out[15]:

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

y=df\_new['Outcome']

X=df\_new.drop('Outcome',axis=1)

In [17]:

linkcode

*#spliting X and y*

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,y,test\_size=0.20,random\_state=0,stratify=df\_new['Outcome'])

from sklearn.linear\_model import LogisticRegression

model=LogisticRegression()

model.fit(X\_train,Y\_train)

/opt/conda/lib/python3.7/site-packages/sklearn/linear\_model/\_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max\_iter) or scale the data as shown in:

https://scikit-learn.org/stable/modules/preprocessing.html

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear\_model.html#logistic-regression

extra\_warning\_msg=\_LOGISTIC\_SOLVER\_CONVERGENCE\_MSG,

LogisticRegression()

In [19]:

*#get prediction*

y\_predict=model.predict(X\_test)

y\_predict

Out[19]:

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

0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,

1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0,

1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,

0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,

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

In [20]:

linkcode

*#confusion matrix*

from sklearn.metrics import confusion\_matrix

cm=confusion\_matrix(Y\_test,y\_predict)

cm

Out[20]:

array([[89, 11],

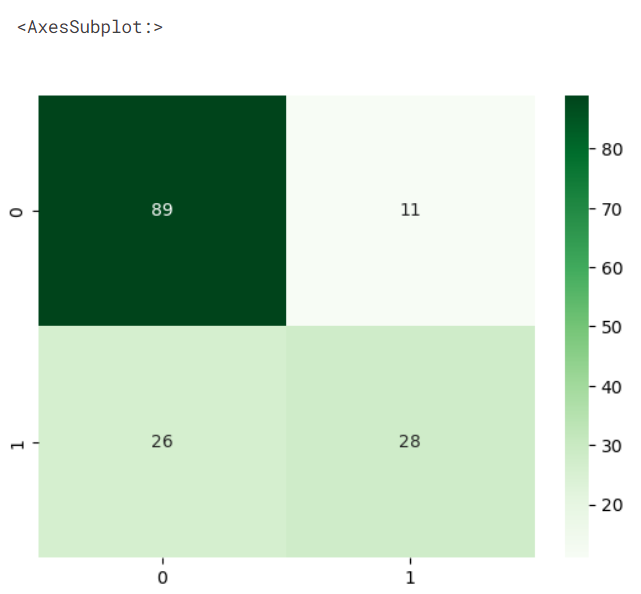
[26, 28]])

In [21]:

linkcode

*#heatmap of confusion matrix*

sns.heatmap(pd.DataFrame(cm),annot=True, cmap="Greens")

****

*#accuracy score*

from sklearn.metrics import accuracy\_score

accuracy=accuracy\_score(Y\_test,y\_predict)

print("Accuracy : ",round(accuracy,2)\*100,'%')

**ACCURACY:**

**76.0%**

**Conclusion:**

**In this phase we have used some techniques based on data preprocessing and loading the above code is one of the technique used and got the accuracy of about 76.0% with around run time of 59.5s.**

**COLLEGE NAME:**

**PANIMALAR INSTITUTE OF TECHNOLOGY**

**NM ID:**

**au211521106088**

**REGISTER NO:**

**211521106088**