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 The zero value has been replaced by its corresponding mean value.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

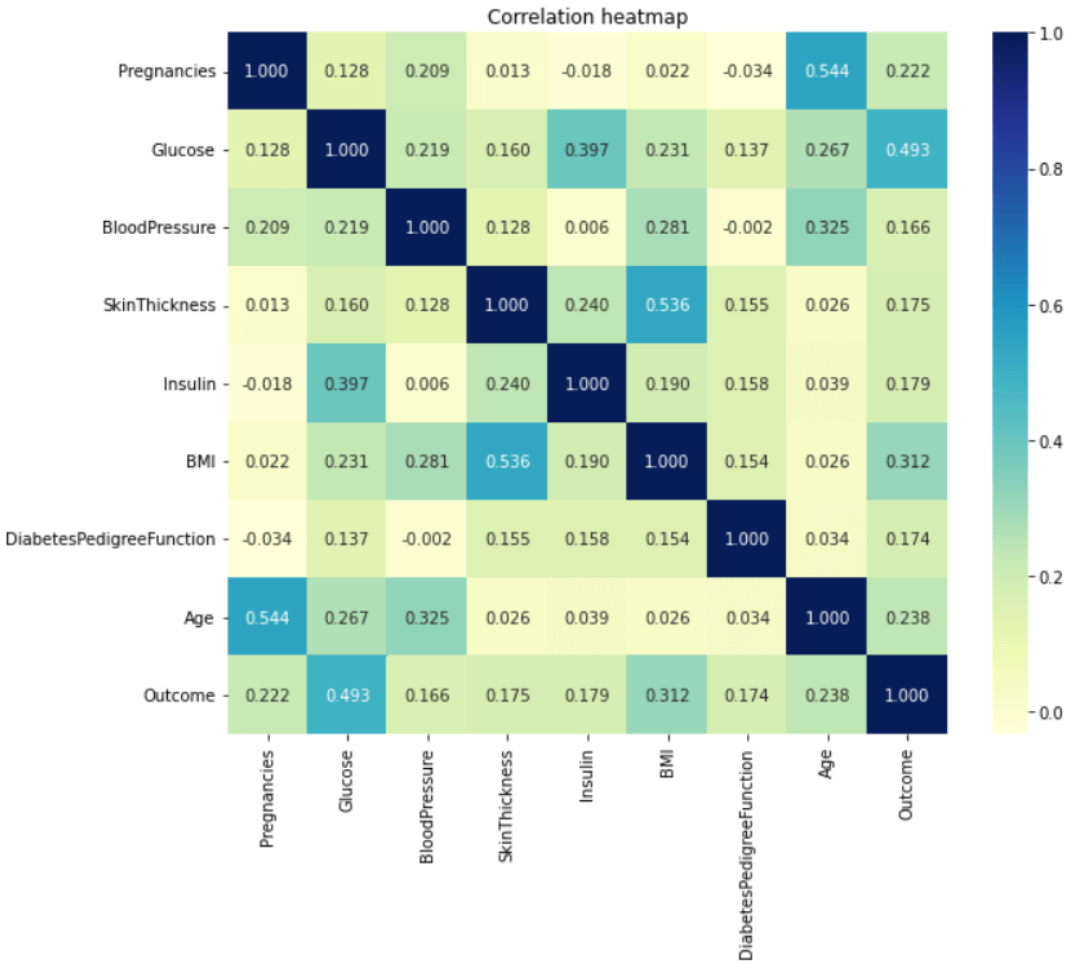
Exploratory Data Analysis :

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")
```

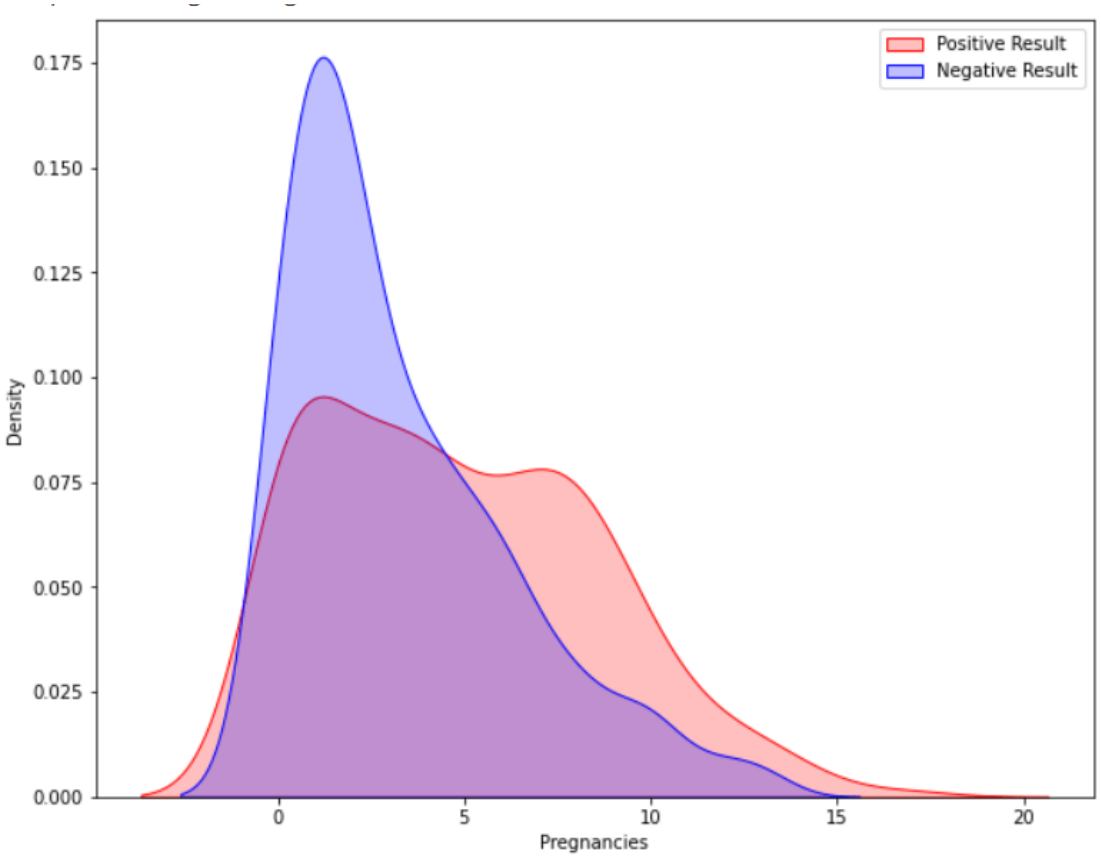


Observation show that characteristic like pregnancy,glucose are more closely associated with the outcomes.Demonstrated a detailed illustration.

Pregnancy:

Code:

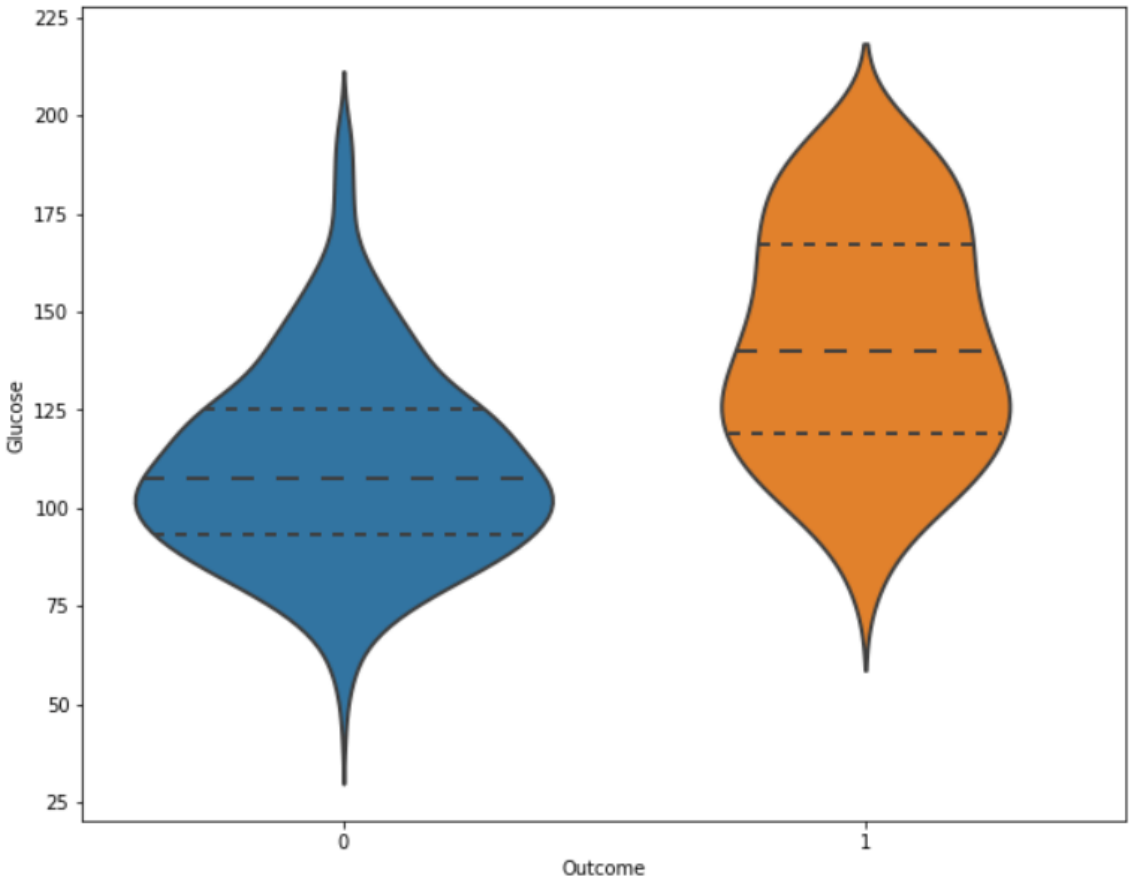
```
plt.figure(figsize = (10, 8))
kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 1], color = "Red" shade = True)
kde = sns.kdeplot(data["Pregnancies"][data["Outcome"] == 0], ax = kde, color = "Blue", shade= True)
kde.set_xlabel("Pregnancies")
kde.set_ylabel("Density")
kde.legend(["Positive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