ML ASSIGNMENT-2

Team Members:
R.Aishwarya
M.Jaison Aro
S.snegitha

Dataset title: Diabetics

Dataset description: The dataset contains 768 rows and 9 columns. The following features have been provided to help us predict whether a person is diabetic or not:

- **Pregnancies:** Number of times pregnant
- **Glucose:** Plasma glucose concentration over 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 (a function which scores likelihood of diabetes based on family history)
- **Age:** Age (years)
- **Outcome:** Class variable (0 if non-diabetic, 1 if diabetic)

Data cleaning and preprocessing:

- We have to do data cleaning before implementation. We will find the missing values.
- If there is any missing values we will replace those values. But the data set does not have missing values. This is data cleaning and pre-processing.

Data exploration:

- Data exploration gives deep understanding about the data.
- We make a count plot from sea born library for the outcome variable to know the count of diabetic and non diabetic patients.

Data visualisation:

- Data visualisation helps us understand the data by placing it in visual context so that patterns,correlation and trends can be seen.
- For this data set we have made a pairplot from seaborn library.

Implementation:

- We'll be using Python and some of its popular data science related packages.
- First of all, we will import pandas to read our data from a CSV file and manipulate it for further use. We will also use numpy to convert out data into a format suitable to feed our classification model.
- We'll use seaborn and matplotlib for visualizations.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
```

```
data = pd.read_csv("diabetes2.csv")
data.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

```
data.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
```

```
data.describe()
```

	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

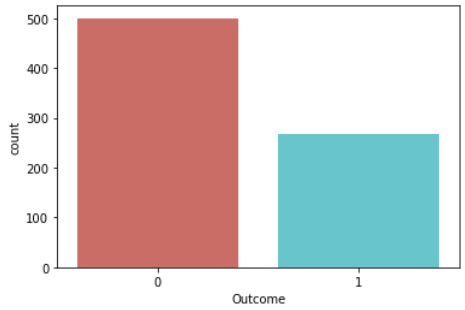
```
data.isnull().sum()
```

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

```
print(data.groupby('Outcome').size())
```

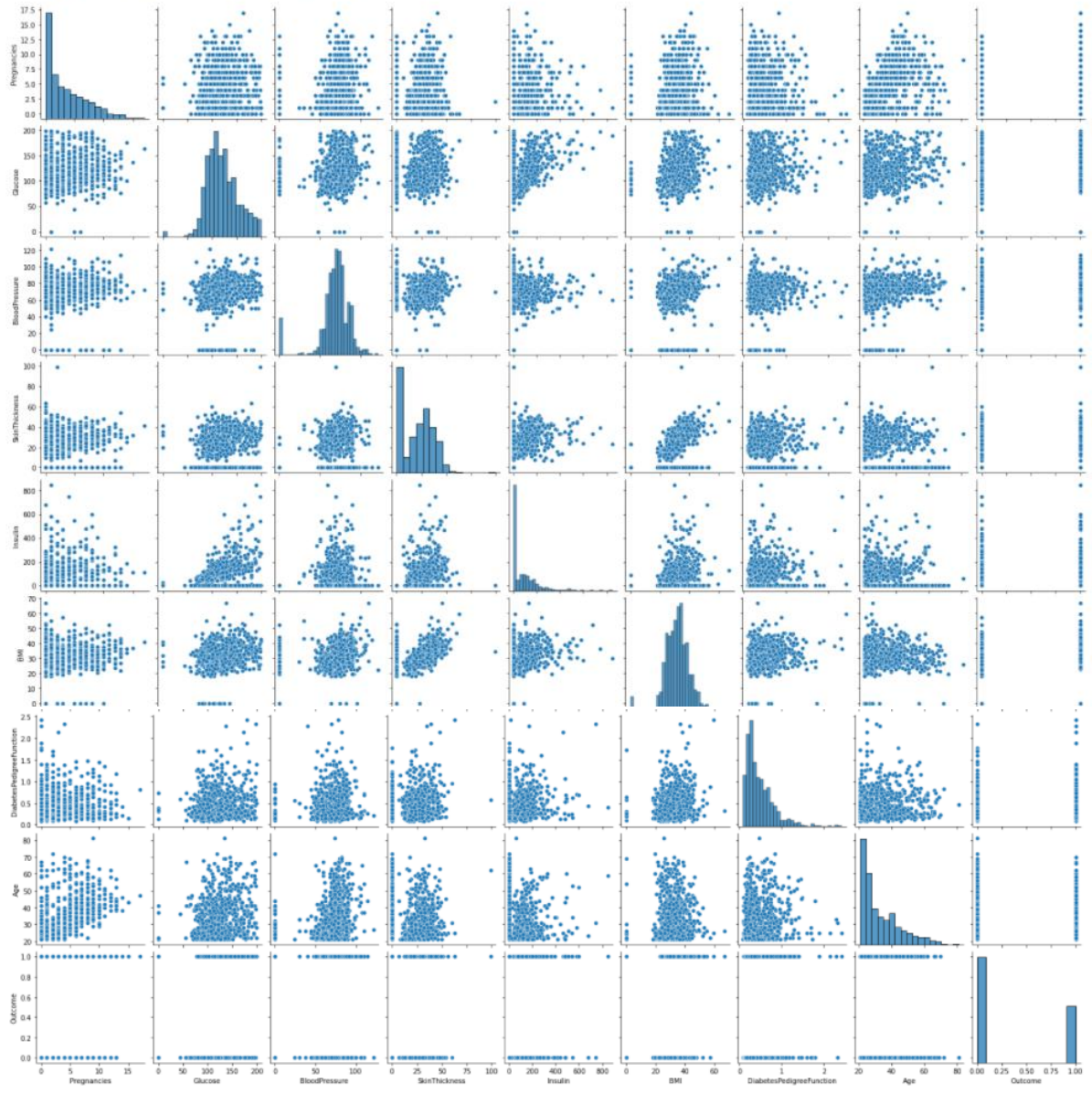
```
Outcome
0    500
1    268
dtype: int64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x='Outcome',data=data,palette="hls")
plt.show()
```



```
import seaborn as sns
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x20cab45b940>
```




```
data.interpolate(method ='linear', inplace=True, limit_direction ='both')
data
```

	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

```
data
```

	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

```
x = data.drop('Outcome',axis=1)
y = data['Outcome']
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

- We will then import Logistic Regression algorithm from sklearn.

```
## Logistic Regression without augumenting method

model1 = LogisticRegression()
model1.fit(x_train,y_train)
y_pred = model1.predict(x_test)
print(accuracy_score(y_pred,y_test))
```

```
0.7467532467532467
```

- Gradient descent is implemented by importing XGBClassifier and we fit the model with x_train and y_train data.
- We predict y value and display accuracy score.

```
## Gradient descent

from xgboost import XGBClassifier

model2 = XGBClassifier()
model2.fit(x_train,y_train)
y_pred = model2.predict(x_test)
print(accuracy_score(y_pred,y_test))
```

```
0.6883116883116883
```

- To implement LDA we import Linear Discriminant Analysis from sklearn library.

- We fit the model with number of components and predict y and analyse the accuracy.

```
## LDA

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=0.2,random_state=42)
model3 = LinearDiscriminantAnalysis(n_components=1)
model3.fit(x_train, y_train)
y_pred = model3.predict(x_test)
print(accuracy_score(y_test, y_pred))
```

```
0.7597402597402597
```

Result:

Thus, we performed gradient descent, LDA and logistic regression for pre-processing and classifying diabetic's patients.