University of Liberal Arts Bangladesh

Course Code: CSE 483

Course Title: Artificial Intelligence Lab

Complex Engineering Project

Submitted to:

Dr. Noman Hossain

Department of CSE

University of Liberal Arts Bangladesh

Submitted by:

Md. Mutasim Billah Abu Noman Akanda (192014038)

Sumaiya Islam Mouno (192014043)

Sabrina Sarwar (192014046)

Department of CSE

Spring'22

[25th May 2022]

Contents Page No.

Objective

Develop a traditional Machine Learning based Artificial Intelligence (AI) system to classify objects from real-world use cases.

Resources

We are using Spyder (python 3.9) to develop a supervised Machine Learning model. Spyder has all the benefits of a comprehensive development tool with the competencies of a scientific package.

Logic

We are using Support Vector Machine (SVM).

Algorithm

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane.

SVM can be of two types

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed linearly separable data, and classifier is used called Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed non-linear data, and the classifier used is called a Non-linear SVM classifier.

Introduction

Diabetes is a chronic (long-lasting) health condition that affects how your body turns food into energy.

Most of the food you eat is broken down into sugar (also called glucose) and released into your bloodstream.

Goal

We will classify diabetic patients by using SVM model from a dataset where there are 9 clinical features including 1 target feature and there are 2 classes in this problem (diabetic and not-diabetic).

Data set

['diabetic.csv']

Input

#Libraries

#Visualization

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap
```

#Read and Examination Dataset

#read data

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

data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctic
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

Fig. 1: 1st 5 rows of Diabetic dataset

#Split Data as M&B

```
p = data[data.Outcome == 1]
n = data[data.Outcome == 0]
```

#Basic Visualization

```
sns.countplot(x='Outcome',data=data)
plt.title("Count 0 & 1")
plt.show()
```

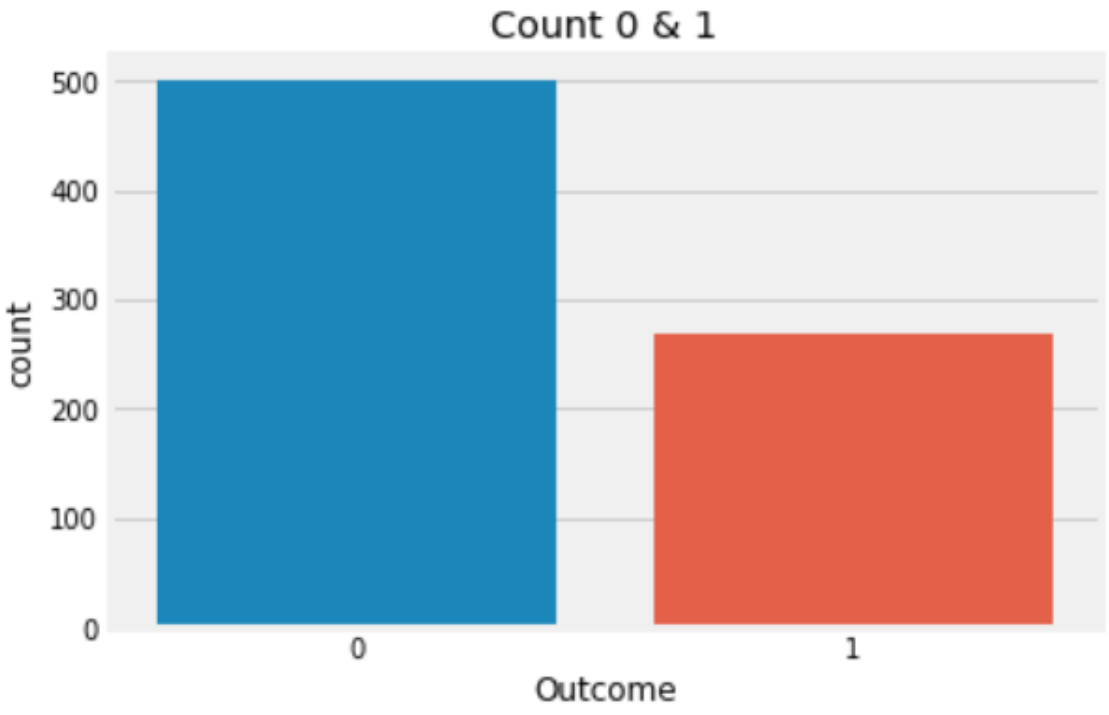

Fig. 2: Frequency Analysis of diabetic and not-diabetic patients

#Exploratory Data Analysis

```
print('Data Shape',data.shape)

print(data.info())

describe = data.describe()

describe.T
```

```
Data Shape (768, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
```

Fig. 3: Shape, information and description of the data

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Fig. 4: t-test of the data

```
corr_data = data.corr()
sns.clustermap(corr_data,annot= True,fmt = '.2f')
plt.title('Correlation Between Features')
plt.show()
```

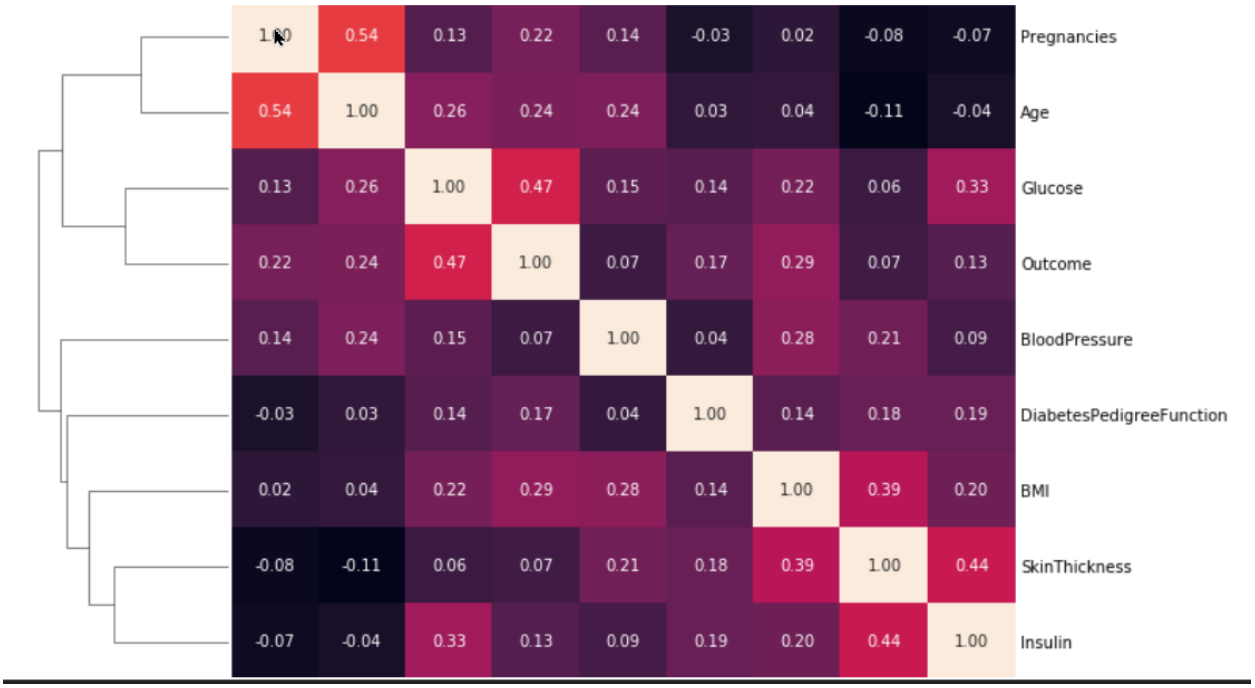

Fig. 5: Correlation among the features

#Feature Visualization with Box plot [Before Standardization]

```
data_melted = pd.melt(data,id_vars='Outcome',
                      var_name='Features',
                      value_name='Value')
plt.figure()
sns.boxplot(x='Features',y='Value',hue='Outcome',data=data_melted)
plt.xticks(rotation=75)
plt.show()
```

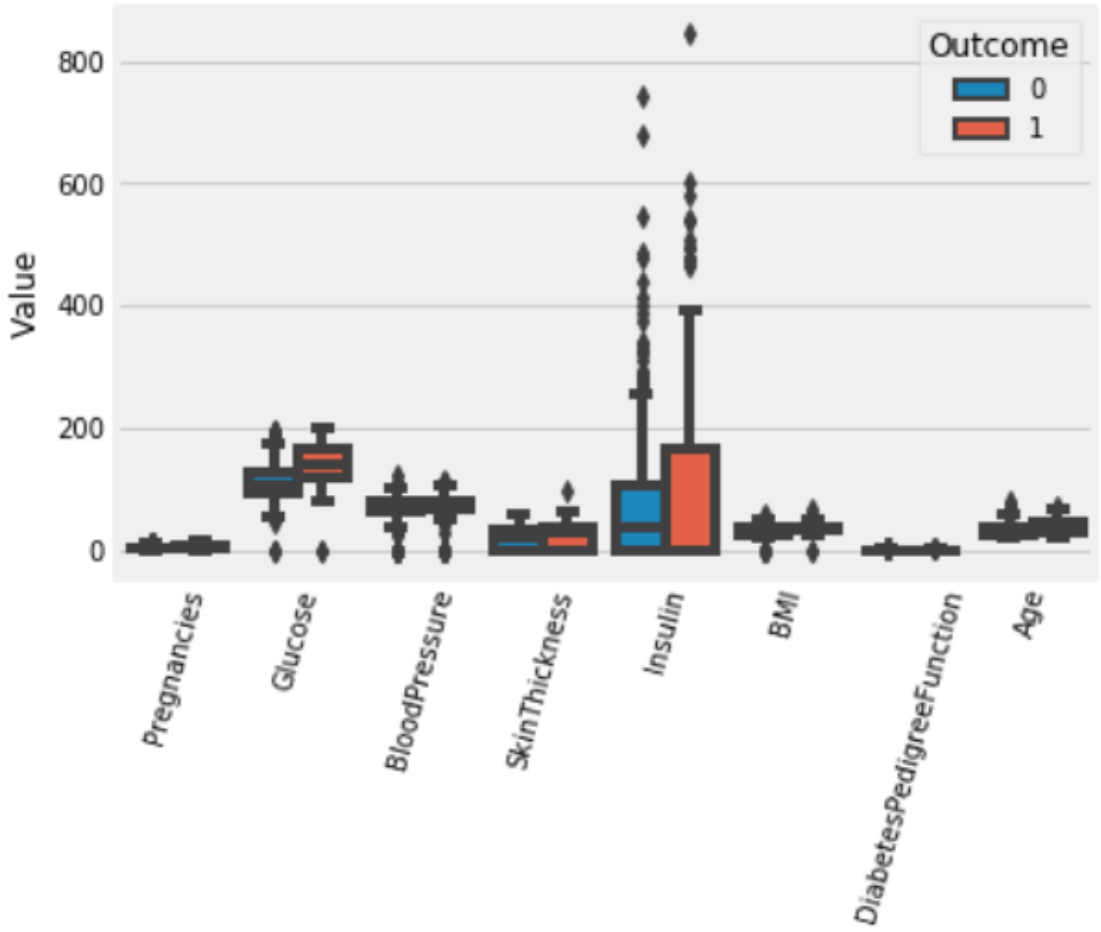

Fig. 6: Box plot of the data

#Analysis of Diabetic Cases

#General Analysis

```
data1 = data[data["Outcome"]==1]
columns = data.columns[:8]
plt.subplots(figsize=(18,18))
length =len(columns)
for i,j in itertools.zip_longest(columns,range(length)):
    plt.subplot((length/2),3,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    plt.ylabel("Count")
    data1[i].hist(bins=20,edgecolor='black')
```

```
  plt.title(i)
plt.show()
```

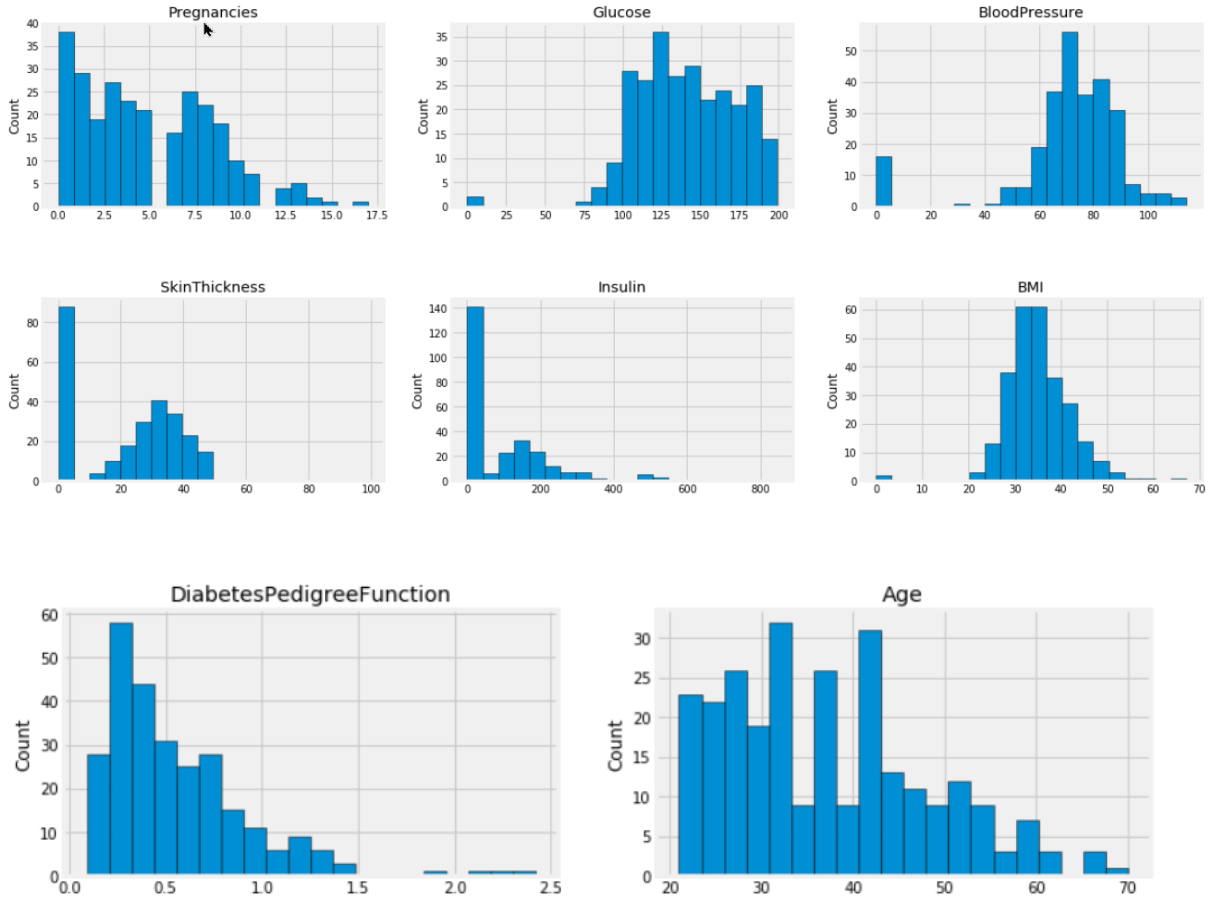

Fig. 7: Histogram of the features of diabetic patients

#Analysis of Non-Diabetic Cases

```
data1 = data[data["Outcome"]==0]
columns = data.columns[:8]
plt.subplots(figsize=(18,18))
length =len(columns)
for i,j in itertools.zip_longest(columns,range(length)):
    plt.subplot((length/2),3,j+1)
    plt.subplots_adjust(wspace=0.2,hspace=0.5)
    plt.ylabel("Count")
```

```
  data1[i].hist(bins=20,edgecolor='black')
    plt.title(i)
plt.show()
```

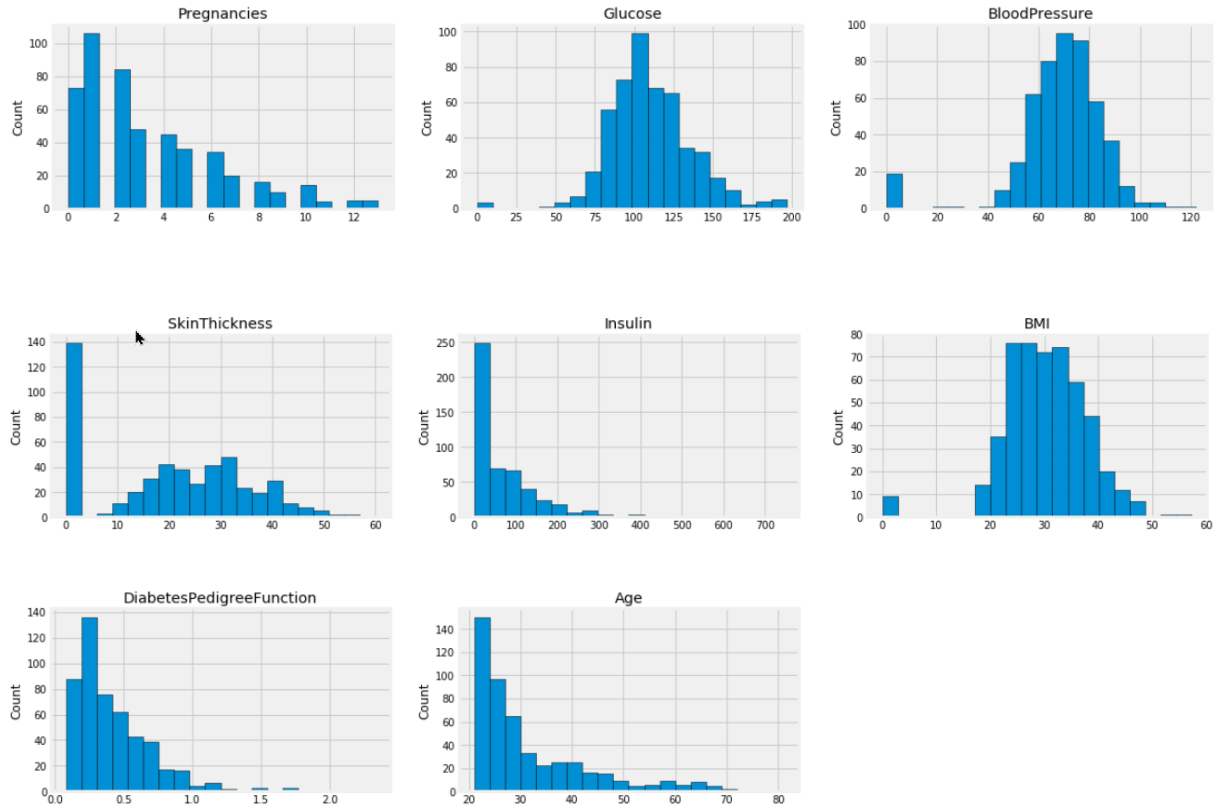

Fig. 7: Histogram of the features of not-diabetic patients

#Visualization of Features

```
plt.scatter(p.Pregnancies,p.Glucose,color = "brown",label="Diabet
Positive",alpha=0.4)
plt.scatter(n.Pregnancies,n.Glucose,color = "Orange",label="Diabet
Negative",alpha=0.2)
plt.xlabel("Pregnancies")
plt.ylabel("Glucose")
plt.legend()
plt.show()
```

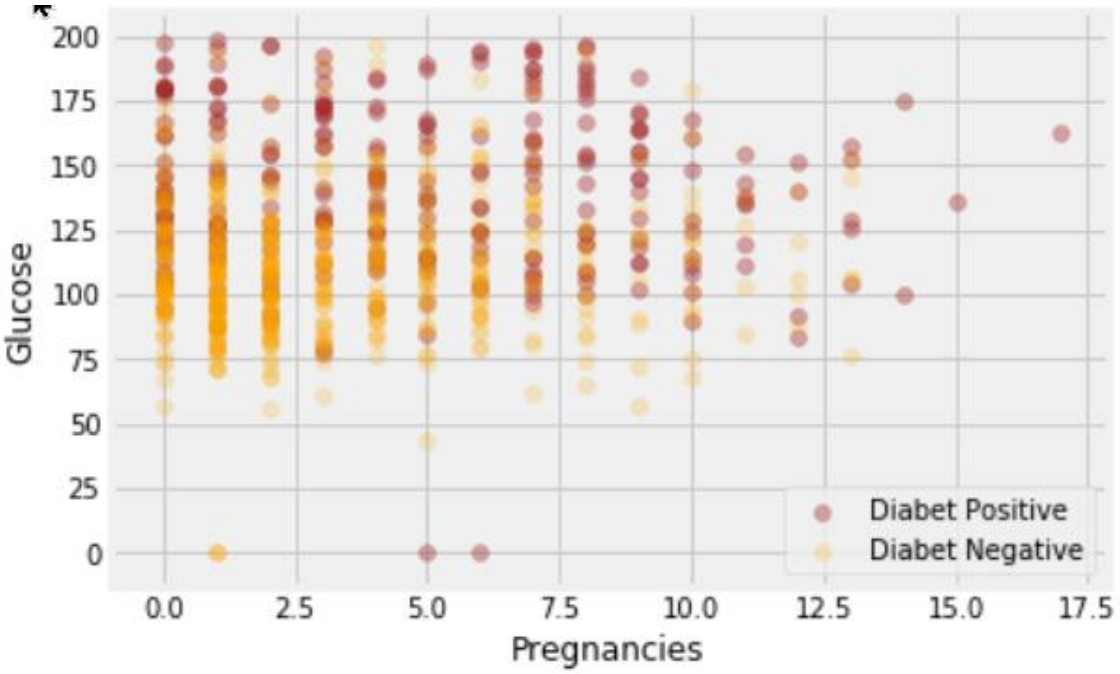

Fig. 8: Scatter plot between Pregnancies and Glucose features

#Visualization, Scatter Plot

```
plt.scatter(p.Age,p.Pregnancies,color = "lime",label="Diabet
Positive",alpha=0.4)

plt.scatter(n.Age,n.Pregnancies,color = "black",label="Diabet
Negative",alpha=0.2)

plt.xlabel("Age")

plt.ylabel("Pregnancies")

plt.legend()

plt.show()
```

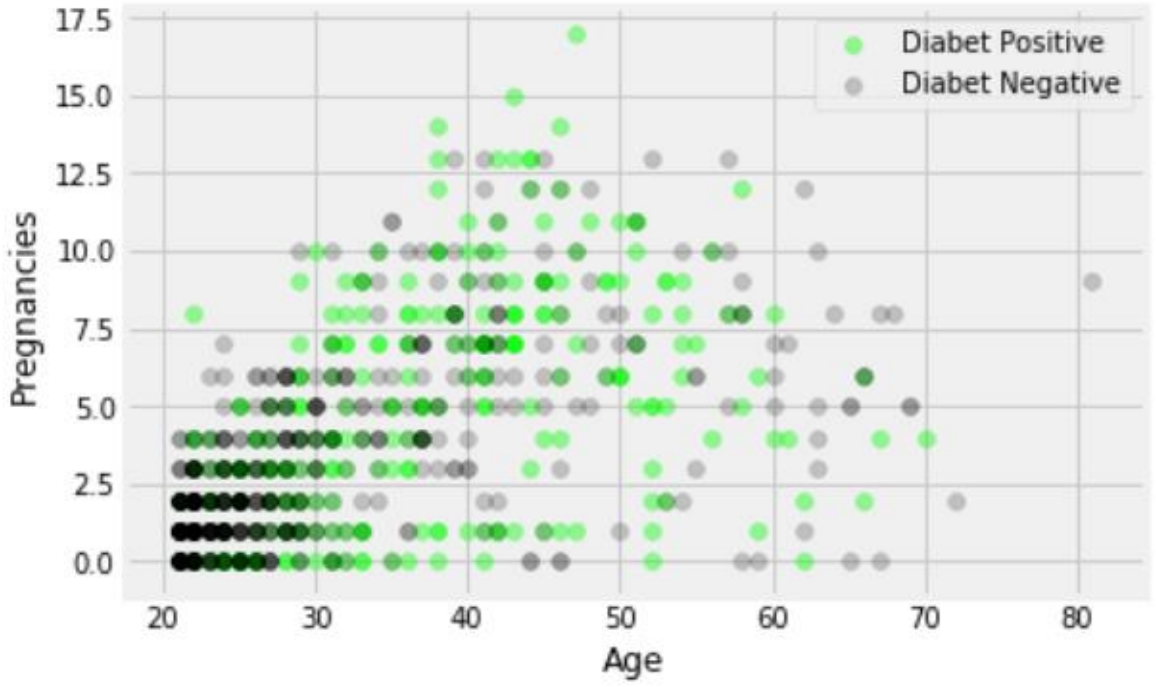

Fig. 9: Scatter plot between Age and Pregnancies features

#Visualization, Scatter Plot

```
plt.scatter(p.Glucose,p.Insulin,color = "lime",label="Diabet
Positive",alpha=0.4)

plt.scatter(n.Glucose,n.Insulin,color = "black",label="Diabet
Negative",alpha=0.1)

plt.xlabel("Glucose")

plt.ylabel("Insulin")

plt.legend()

plt.show()
```

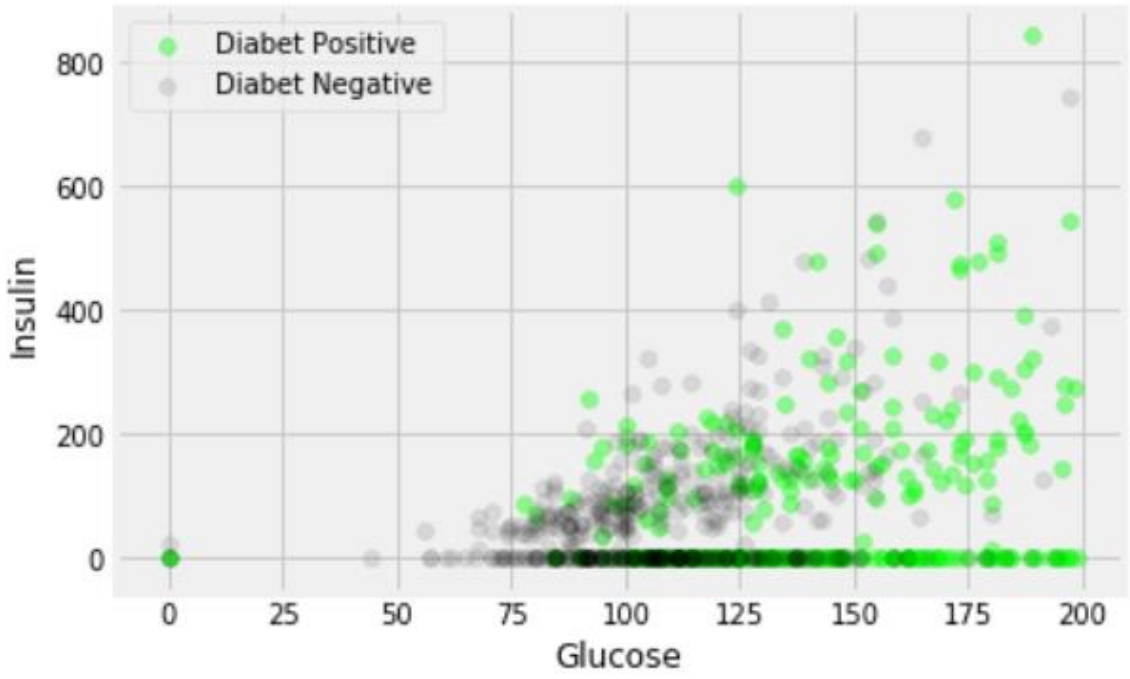

Fig. 10: Scatter plot between Glucose and Insulin features

#Outlier Detection

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

y = data.Outcome

columns = x.columns.tolist()

clf = LocalOutlierFactor()

y_pred = clf.fit_predict(x)

x_score = clf.negative_outlier_factor_

outlier_score = pd.DataFrame()

outlier_score['score'] = x_score

threshold_outliers = -1.5

filtre = outlier_score['score'] < threshold_outliers

outlier_index = outlier_score[filtre].index.tolist()

plt.figure()
```

```
plt.scatter(x.iloc[outlier_index,0],  x.iloc[outlier_index,1],color  =
'blue',s=50,label='outliers')

plt.scatter(x.iloc[:,0]

           ,x.iloc[:,1],color='k',s=3,label='data_point')

radius = (x_score.max() - x_score ) / (x_score.max() - x_score.min() )

outlier_score['radius '] = radius

plt.scatter(x.iloc[:,0],          x.iloc[:,1],          s=1000*radius,
edgecolors='r',facecolor='none',label='Outlier skores')

plt.legend()

plt.show();
```

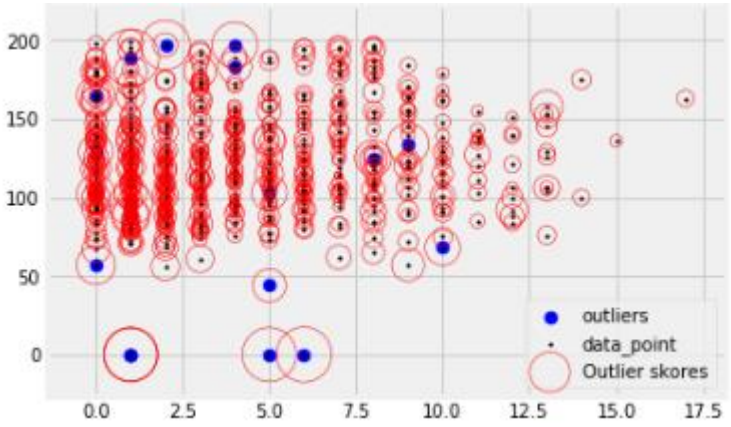

Fig. 11: Outlier Detection

#Drop Outliers

```
x = x.drop(outlier_index) #outliers remove

y = y.drop(outlier_index).values #outliers remove
```

#Train Test Split and Standardization Processing

```
test_size = 0.2

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

scaler = StandardScaler()
```

```
x_train = scaler.fit_transform(x_train)

x_test = scaler.transform(x_test)

x_train_df = pd.DataFrame(x_train,columns=columns)

x_train_df_describe = x_train_df.describe()

x_train_df['target'] = y_train
```

#Box Plot Visualization after the Standardization

```
data_melted = pd.melt(x_train_df,id_vars='target',
                      var_name='Features',
                      value_name='Value')
plt.figure()
sns.boxplot(x='Features',y='Value',hue='target',data=data_melted)
plt.xticks(rotation=75)
plt.show()
```

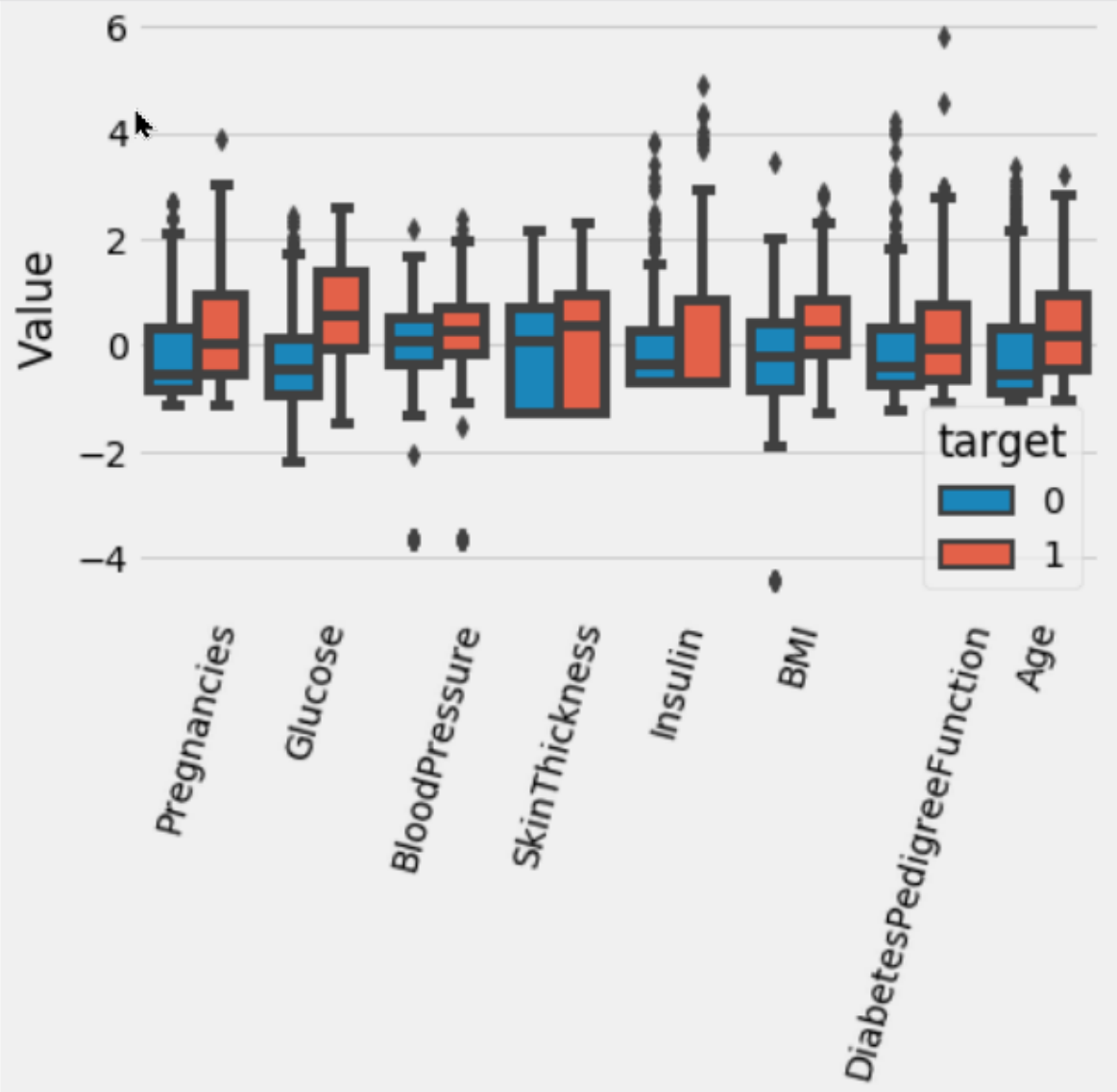

Fig. 11: Box plot after melting data

#Support Vector Machine (SVM)

#SVM with Sklearn

```
from sklearn.svm import SVC
SVM = SVC(random_state=42)
SVM.fit(x_train,y_train)  #learning
#SVM Test
print ("SVM Accuracy:", SVM.score(x_test,y_test))
```

```
SVMscore = SVM.score(x_test,y_test)
```

#Confusion Matrix

```
yprediciton3= SVM.predict(x_test)

ytrue = y_test

from sklearn.metrics import confusion_matrix

CM = confusion_matrix(ytrue,yprediciton3)
```

#CM visualization

```
import seaborn as sns

import matplotlib.pyplot as plt

f, ax = plt.subplots(figsize=(5,5))

sns.heatmap(CM,annot = True,
linewidths=0.5,linecolor="red",fmt=".0f",ax=ax)

plt.xlabel("Prediction(Ypred)")

plt.ylabel("Ytrue")

plt.show()
```

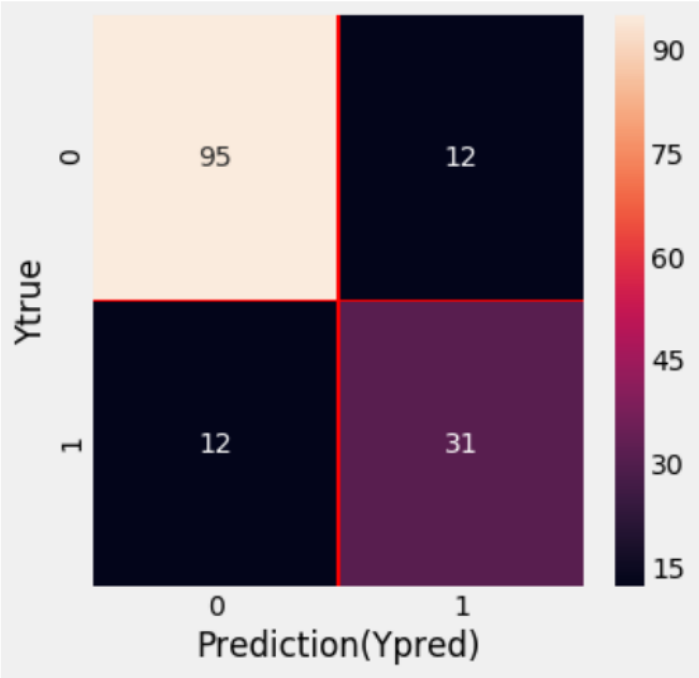

Fig. 12: Heatmap

Output

SVM accuracy: 0.84

That means our SVM model is classifying the diabetic patients 84% accurately.

Conclusion

Using the SVM model, we have successfully classified which patient has diabetic and which has not.

THE END