Implementation of Confusion Matrix

https://www.kaggle.com/datasets/mathchi/diabetes-data-set

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [36]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
```

In [38]:
```
data = pd.read_csv('C:/Users/Acer/Downloads/diabetes.csv')
```

In [40]:
```
data.head()
```

Out[40]:

	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 [42]:
```
data.shape
```

Out[42]:
```
(768, 9)
```

In [44]:
```
data.columns
```

Out[44]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

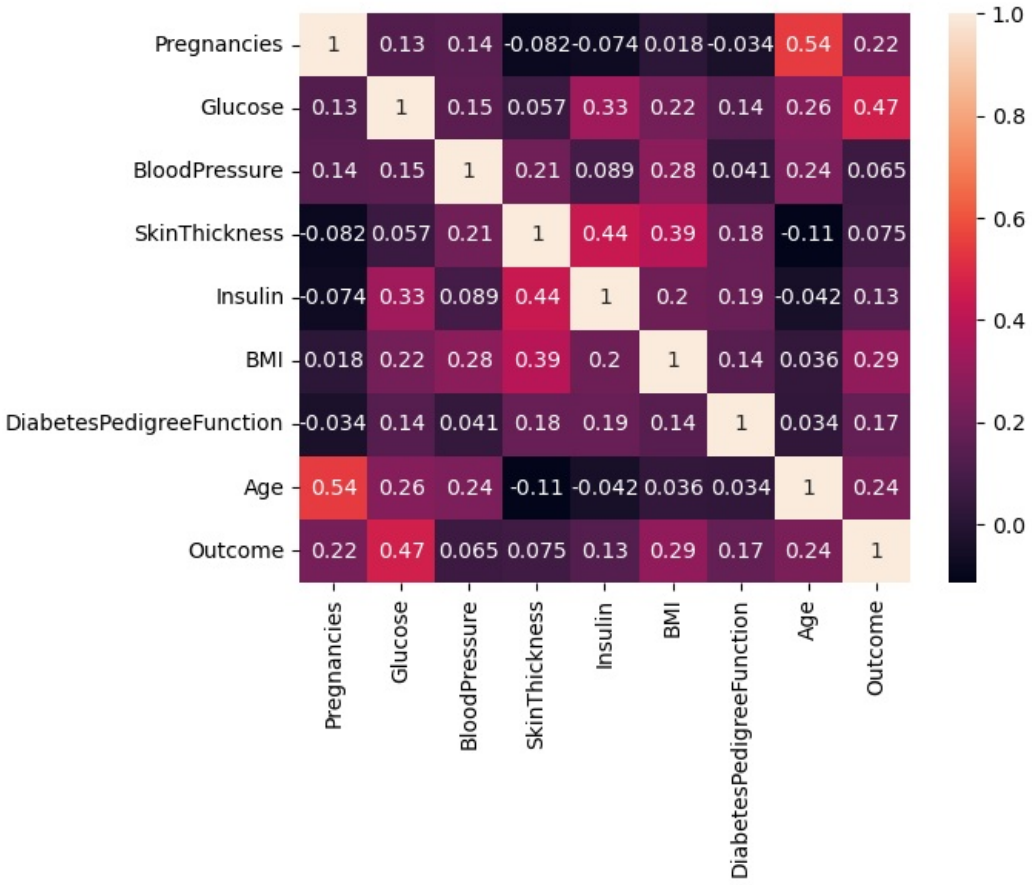
In [46]:
```
#Calculate person correlation between every pair of features(how one feature moves with another)
data.corr()
```

Out[46]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0

In [48]:
```
#Visualize the correlation nicely with colours :red for +ve and blue for -ve
sns.heatmap(data.corr(),annot=True)
```

Out[48]:
```
<Axes: >
```



In [50]:
```
#Separate feature and target
X=data.drop('Outcome', axis=1)  #Features
Y=data['Outcome']
```

In [52]:
```
#Split inti training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [54]:
```
#Feature scaling for better model performance
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
```

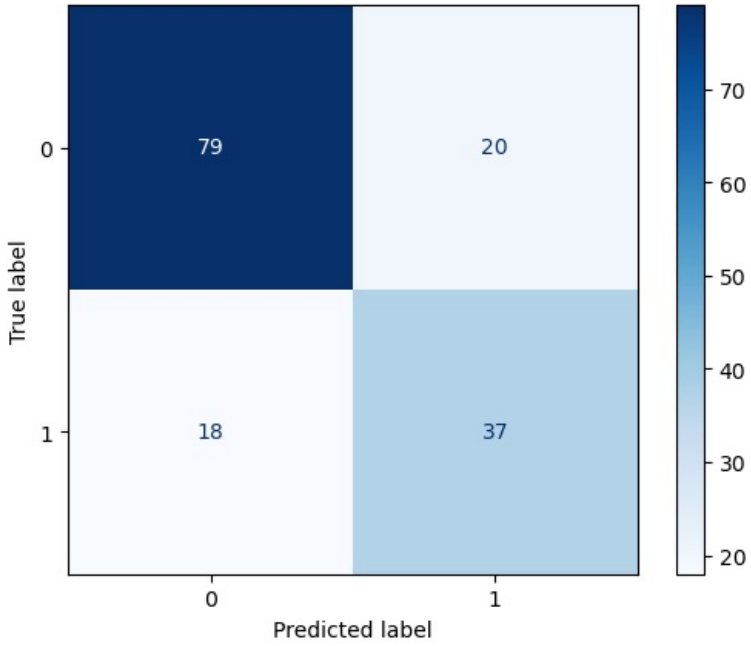
In [58]:
```
#Train a simple Logistics regression model
model=LogisticRegression()
model.fit(X_train_scaled,Y_train)
```

Out[58]:
```
LogisticRegression
LogisticRegression()
```

In [60]:
```
#Predict on the test set
Y_pred=model.predict(X_test_scaled)
```

In [62]:
```
#Generate confusion matrix
cm=confusion_matrix(Y_test, Y_pred)
```

In [64]:
```
#The COnfusion matrix shows True Positive,False Positive, True Negative and False Negative
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
disp.plot(cmap='Blues')
plt.show()
```



37 - True Positive 18 - False Positive 20 - False Negative 79 - True Negative

In [76]:
```
from sklearn import metrics
accuracy = metrics.accuracy_score(Y_test,Y_pred)
accuracy
```

Out[76]: 0.7532467532467533

In [80]:
```
precision = metrics.precision_score(Y_test,Y_pred)
precision
```

Out[80]: 0.6491228070175439

In [82]:
```
recall =metrics.recall_score(Y_test,Y_pred)
recall
```

Out[82]: 0.6727272727272727

In [86]:
```
f1_score = metrics.f1_score(Y_test,Y_pred)
f1_score
```

Out[86]: 0.6607142857142857