In [1]:

```
import pandas as pd
import numpy  as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
data=pd.read_csv('health care diabetes.csv')
```

In [3]:

```
data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

```
768 rows × 9 columns
```

In [4]:

```
data.shape
```

```
(768, 9)
```

In [5]:

```
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
```

No Null Values in Data

In [6]:

```
data['Pregnancies'].isnull().sum()
data['Glucose'].isnull().sum()
data['BloodPressure'].isnull().sum()
data['SkinThickness'].isnull().sum()
data['Insulin'].isnull().sum()
data['BMI'].isnull().sum()
data['DiabetesPedigreeFunction'].isnull().sum()
data['Age'].isnull().sum()
```

```
0
```

In [7]:

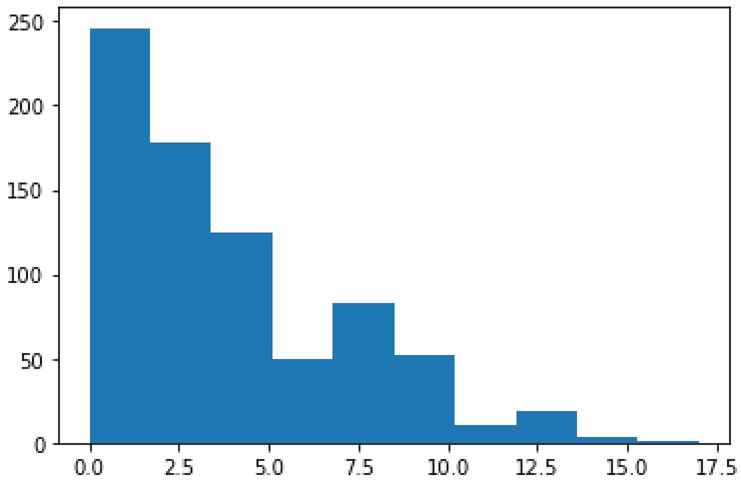
```
data.isnull().any()
```

```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

In [8]:

```
data['Pregnancies'].value_counts()
plt.hist(data['Pregnancies'])
```

```
(array([246., 178., 125.,  50.,  83.,  52.,  11.,  19.,   3.,   1.]),
 array([ 0. ,  1.7,  3.4,  5.1,  6.8,  8.5, 10.2, 11.9, 13.6, 15.3, 17. ]),
 <BarContainer object of 10 artists>)
```



In [9]:

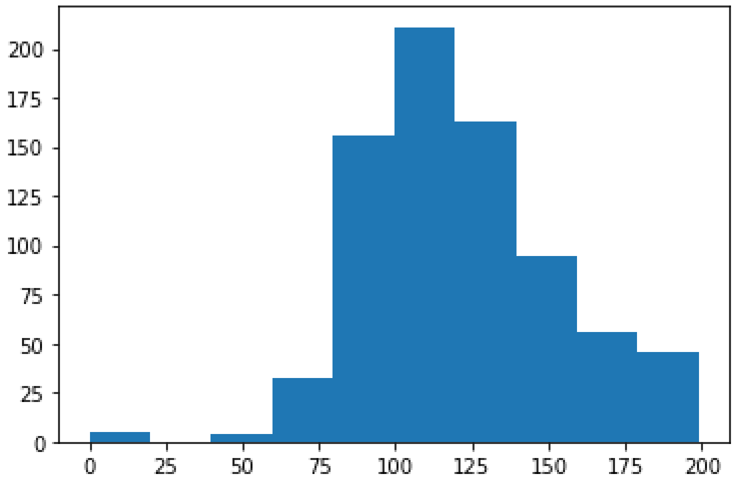
```
data['Pregnancies'].value_counts()
```

```
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
Name: Pregnancies, dtype: int64
```

In [10]:

```
plt.hist(data['Glucose'])
```

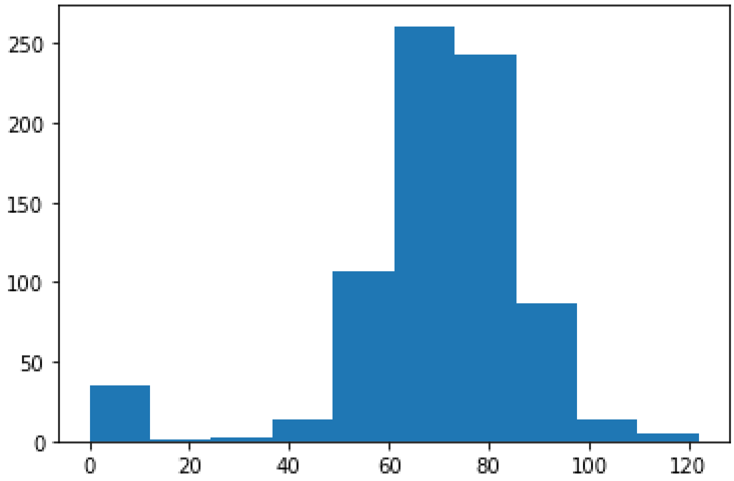
```
(array([  5.,   0.,   4.,  32., 156., 211., 163.,  95.,  56.,  46.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <BarContainer object of 10 artists>)
```



In [11]:

```
data['BloodPressure'].value_counts()
plt.hist(data['BloodPressure'])
```

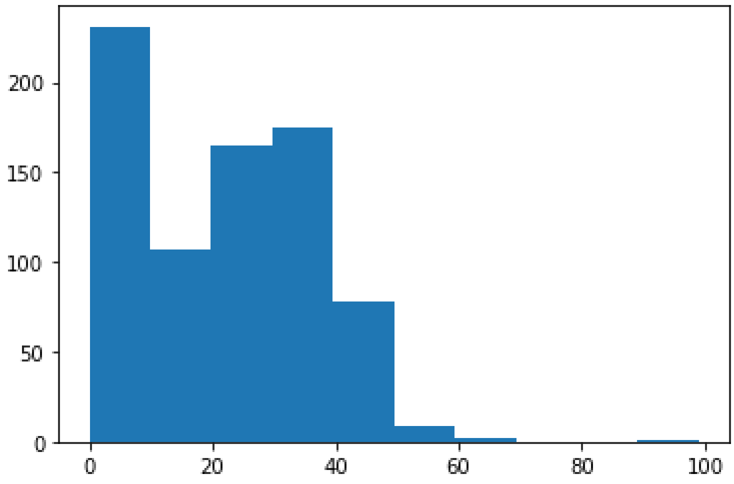
```
(array([ 35.,   1.,   2.,  13., 107., 261., 243.,  87.,  14.,   5.]),
 array([  0. ,  12.2,  24.4,  36.6,  48.8,  61. ,  73.2,  85.4,  97.6,
        109.8, 122. ]),
 <BarContainer object of 10 artists>)
```



In [12]:

```
data['SkinThickness'].value_counts()
plt.hist(data['SkinThickness'])
```

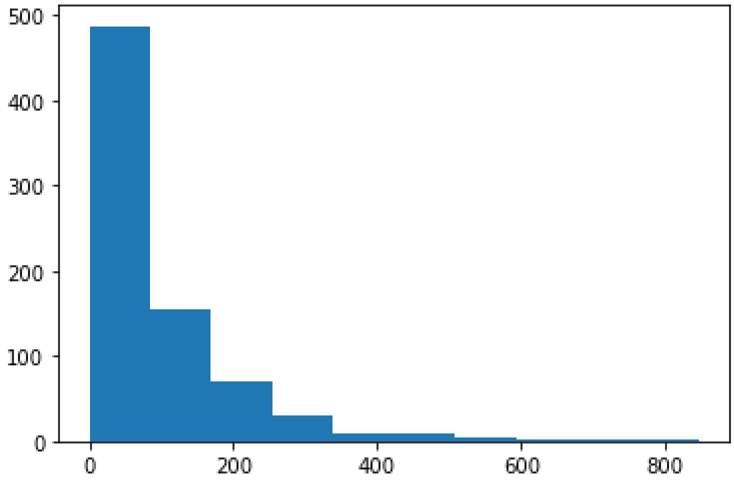
```
(array([231., 107., 165., 175.,  78.,   9.,   2.,   0.,   0.,   1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <BarContainer object of 10 artists>)
```



In [13]:

```
data['Insulin'].value_counts()
plt.hist(data['Insulin'])
```

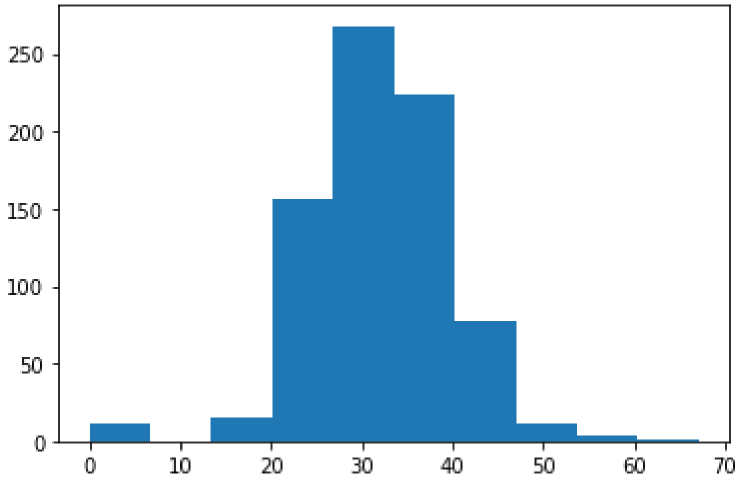
```
(array([487., 155.,  70.,  30.,   8.,   9.,   5.,   1.,   2.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <BarContainer object of 10 artists>)
```



In [14]:

```
data['BMI'].value_counts()
plt.hist(data['BMI'])
```

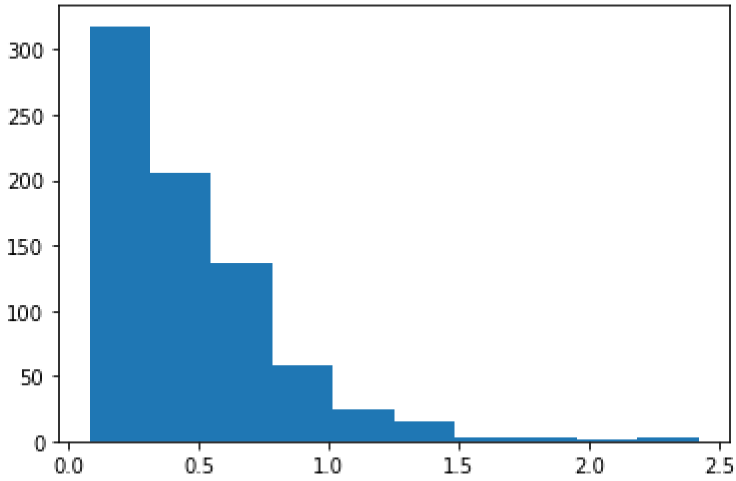
```
(array([ 11.,   0.,  15., 156., 268., 224.,  78.,  12.,   3.,   1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <BarContainer object of 10 artists>)
```



In [15]:

```
data['DiabetesPedigreeFunction'].value_counts()
plt.hist(data['DiabetesPedigreeFunction'])
```

```
(array([318., 206., 136.,  58.,  25.,  15.,   3.,   3.,   1.,   3.]),
 array([0.078 , 0.3122, 0.5464, 0.7806, 1.0148, 1.249 , 1.4832, 1.7174,
        1.9516, 2.1858, 2.42  ]),
 <BarContainer object of 10 artists>)
```



In [16]:

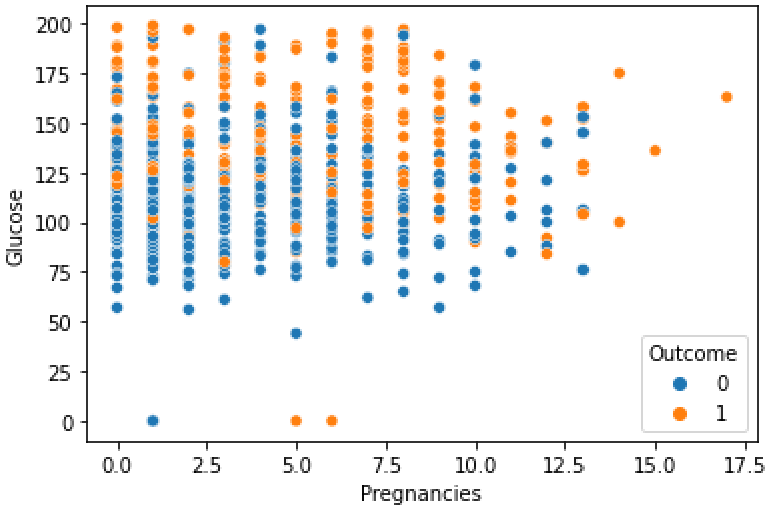
```
data.describe().transpose()
```

	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

In [17]:

```
sns.scatterplot(x='Pregnancies',y='Glucose',hue='Outcome',data=data)
```

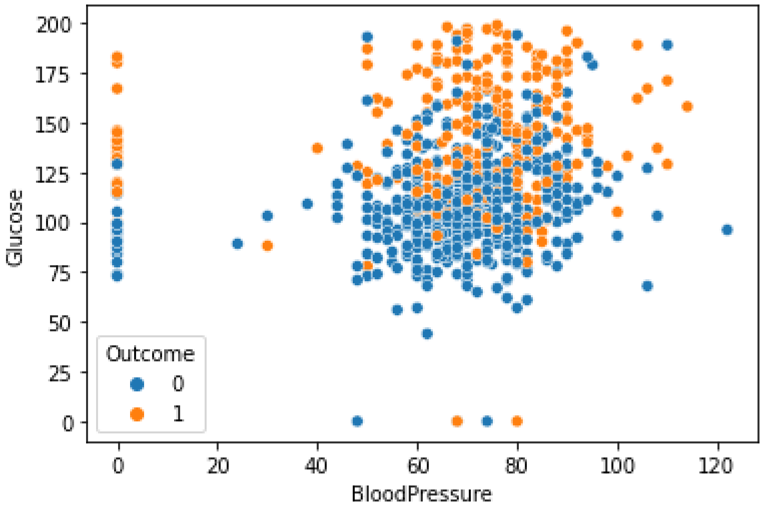
```
<AxesSubplot:xlabel='Pregnancies', ylabel='Glucose'>
```



In [18]:

```
sns.scatterplot(x='BloodPressure',y='Glucose',hue='Outcome',data=data)
```

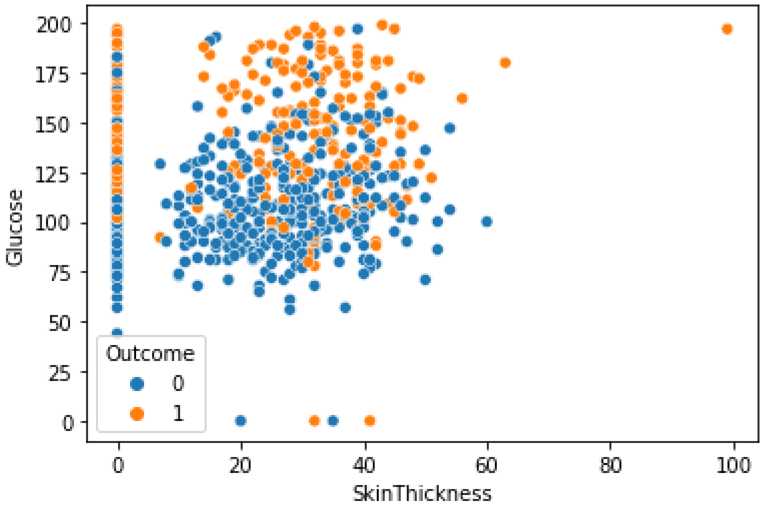
```
<AxesSubplot:xlabel='BloodPressure', ylabel='Glucose'>
```



In [19]:

```
sns.scatterplot(x='SkinThickness',y='Glucose',hue='Outcome',data=data)
```

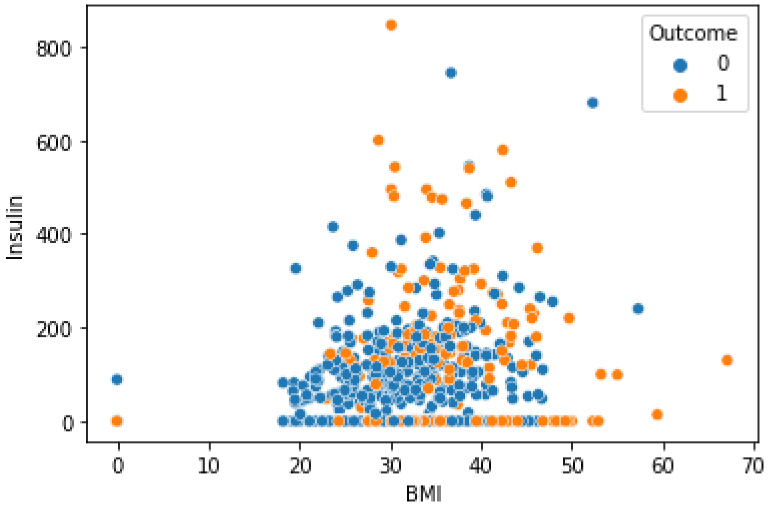
```
<AxesSubplot:xlabel='SkinThickness', ylabel='Glucose'>
```



In [20]:

```
sns.scatterplot(x='BMI',y='Insulin',hue='Outcome',data=data)
```

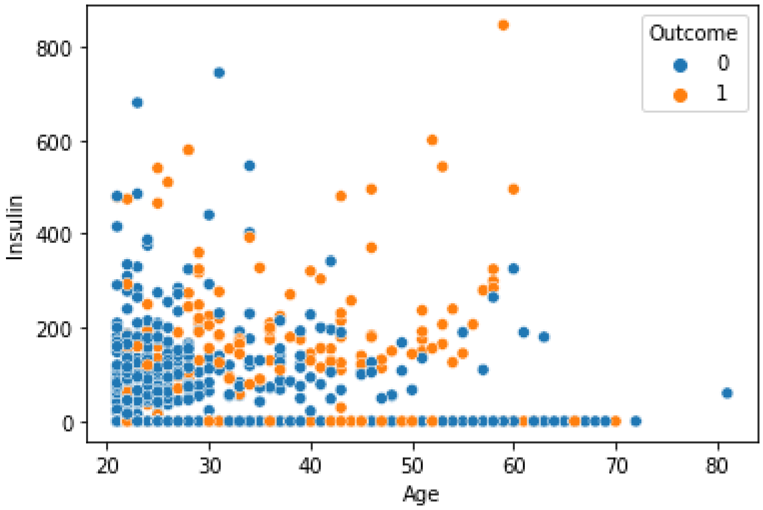
```
<AxesSubplot:xlabel='BMI', ylabel='Insulin'>
```



In [21]:

```
sns.scatterplot(x='Age',y='Insulin',hue='Outcome',data=data)
```

```
<AxesSubplot:xlabel='Age', ylabel='Insulin'>
```



In [22]:

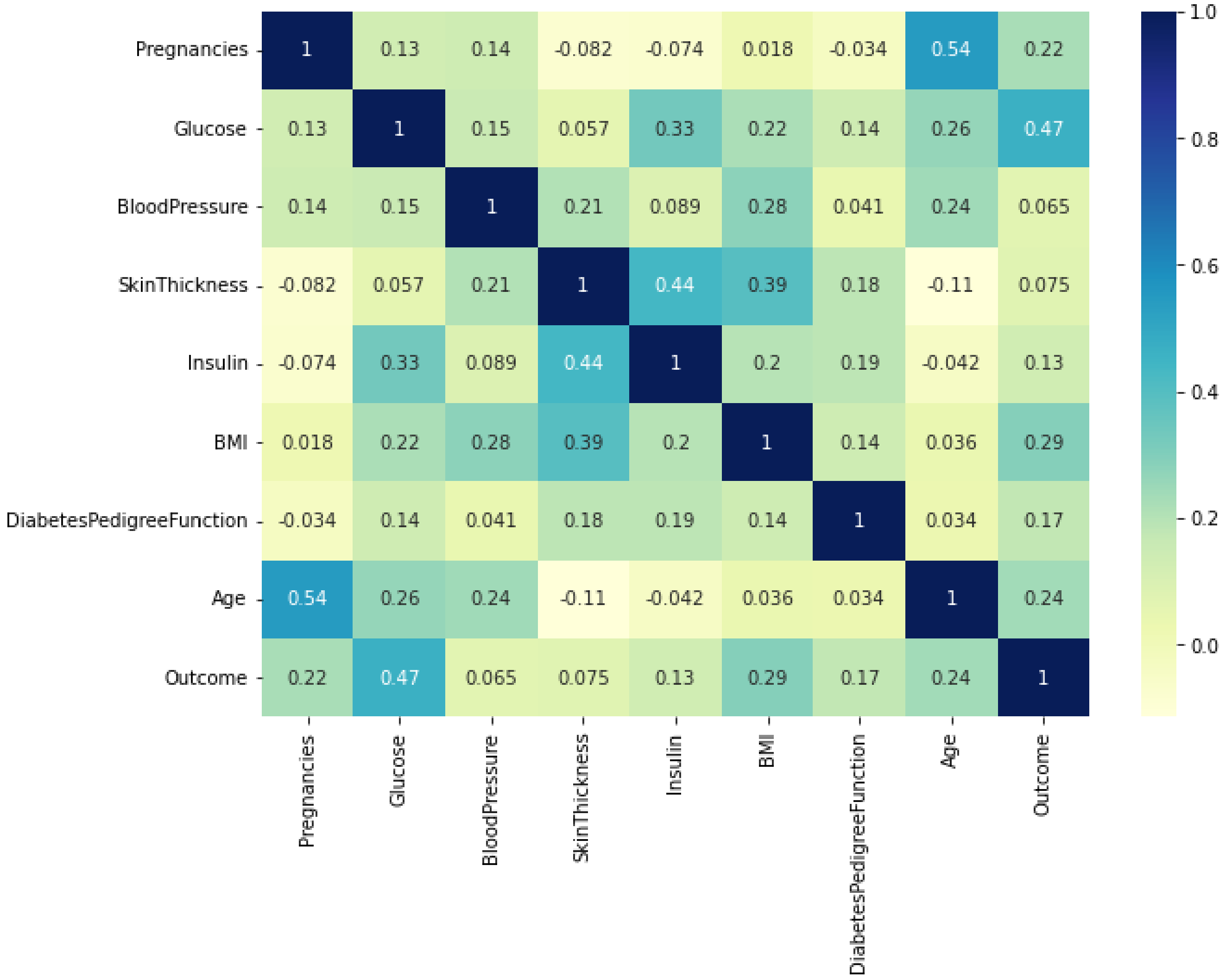
```
data.corr()
```

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

In [23]:

```
plt.figure(figsize=(10,7))
sns.heatmap(data.corr(),annot=True,cmap="YlGnBu")
```

```
<AxesSubplot:>
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.14	-0.082	-0.074	0.018	-0.034	0.54	0.22
Glucose	0.13	1	0.15	0.057	0.33	0.22	0.14	0.26	0.47
BloodPressure	0.14	0.15	1	0.21	0.089	0.28	0.041	0.24	0.065
SkinThickness	-0.082	0.057	0.21	1	0.44	0.39	0.18	-0.11	0.075
Insulin	-0.074	0.33	0.089	0.44	1	0.2	0.19	-0.042	0.13
BMI	0.018	0.22	0.28	0.39	0.2	1	0.14	0.036	0.29
DiabetesPedigreeFunction	-0.034	0.14	0.041	0.18	0.19	0.14	1	0.034	0.17
Age	0.54	0.26	0.24	-0.11	-0.042	0.036	0.034	1	0.24
Outcome	0.22	0.47	0.065	0.075	0.13	0.29	0.17	0.24	1

Model Building

In [24]:

```
X=data.iloc[:,[0,1,2,3,4,5,6,7]]
```

In [25]:

```
y=data.iloc[:,[8]]
```

In [26]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_stat
```

In [27]:

```
from sklearn.linear_model import LogisticRegression
```

In [28]:

```
model= LogisticRegression()
model.fit(X_train,y_train)
```

```
d:\Users\coold\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-v
array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
d:\Users\coold\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs f
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 (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 (https://scikit-learn.org/stable/modules/linear_model.ht
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

In [29]:

```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7801556420233463
0.7480314960629921
```

In [30]:

```
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import classification_report
```

In [31]:

```
yhat=model.predict(X_test)
```

In [32]:

```
print(accuracy_score(y_test,yhat))
```

```
0.7480314960629921
```

In [33]:

```
print(confusion_matrix(y_test,yhat))
```

```
[[136  32]
 [ 32  54]]
```

In [34]:

```
print(classification_report(y_test,yhat))
```

```
              precision    recall  f1-score   support

           0       0.81      0.81      0.81       168
           1       0.63      0.63      0.63        86

    accuracy                           0.75       254
   macro avg       0.72      0.72      0.72       254
weighted avg       0.75      0.75      0.75       254
```

In [35]:

```
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(p=2,n_neighbors=5,metric='minkowski')
```

In [36]:

```
model2.fit(X_train,y_train)
```

```
d:\Users\coold\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:198: DataConversionWarning: A
hen a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

```
KNeighborsClassifier()
```

In [37]:

```
print(model2.score(X_train,y_train))
print(model2.score(X_test,y_test))
yhat2=model2.predict(X_test)
print(confusion_matrix(y_test,yhat2))
print(accuracy_score(y_test,yhat2))
```

```
0.7937743190661478
0.7007874015748031
[[130  38]
 [ 38  48]]
0.7007874015748031
```

In [38]:

```
from sklearn.ensemble import RandomForestClassifier
model3=RandomForestClassifier(n_estimators=15,criterion='gini',max_features='auto')
```

In [39]:

```
model3.fit(X_train,y_train)
print(model3.score(X_train,y_train))
print(model3.score(X_test,y_test))
```

```
0.9922178988326849
0.7401574803149606
```

```
C:\Users\coold\AppData\Local\Temp\ipykernel_18212\2588249225.py:1: DataConversionWarning: A column-vector y w
expected. Please change the shape of y to (n_samples,), for example using ravel().
  model3.fit(X_train,y_train)
```

In [40]:

```
from sklearn.metrics import roc_curve,roc_auc_score
```

In [44]:

```
p1=model.predict_proba(X_train)
p2=model2.predict_proba(X_train)
p3=model3.predict_proba(X_train)
```

In [53]:

```
## kepping probs for positive outcomes only
p1=p1[:,1]
p2=p2[:,1]
p3=p3[:,1]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: