In [55]:
```
#Importing dataset
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

# Loading Dataset

In [56]:
```
data = pd.read_csv("diabetes.csv")
print("Successfully Imported Data!")
data.head()
```

```
Successfully Imported Data!
```

Out[56]:

| | 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 [57]:
```
print(data.shape)
```

```
(768, 9)
```

# Description

In [58]:
```
data.describe(include='all')
```

Out[58]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction |
|---|---|---|---|---|---|---|---|
| **count** | 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 |
| **std** | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 |
| **75%** | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 0.626250 |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 |

# Finding Null Values

In [59]:
```
print(data.isna().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
```

```
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [60]:
```
data.corr()
```

Out[60]:

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

In [61]:
```
data.groupby('Age').mean()
```

Out[61]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Ou |
|---|---|---|---|---|---|---|---|---|
| **Age** | | | | | | | | |
| **21** | 1.079365 | 108.317460 | 65.936508 | 19.349206 | 73.634921 | 27.817460 | 0.433825 | 0.0 |
| **22** | 1.555556 | 108.208333 | 63.722222 | 20.486111 | 74.486111 | 29.509722 | 0.430625 | 0.1 |
| **23** | 1.578947 | 111.578947 | 64.315789 | 22.368421 | 118.026316 | 31.502632 | 0.438579 | 0.1 |
| **24** | 1.891304 | 117.891304 | 64.956522 | 25.934783 | 88.021739 | 32.569565 | 0.393565 | 0.1 |
| **25** | 1.770833 | 110.083333 | 59.666667 | 23.958333 | 82.895833 | 31.943750 | 0.600500 | 0.2 |
| **26** | 1.969697 | 118.212121 | 64.181818 | 23.666667 | 90.878788 | 34.915152 | 0.413455 | 0.2 |
| **27** | 2.562500 | 115.281250 | 73.500000 | 18.375000 | 63.125000 | 31.950000 | 0.471750 | 0.2 |
| **28** | 3.028571 | 119.914286 | 68.314286 | 23.628571 | 94.600000 | 33.642857 | 0.459629 | 0.2 |
| **29** | 3.310345 | 127.379310 | 68.241379 | 21.000000 | 88.793103 | 33.541379 | 0.408897 | 0.4 |
| **30** | 3.619048 | 122.285714 | 64.857143 | 18.904762 | 82.666667 | 30.033333 | 0.367238 | 0.2 |
| **31** | 3.875000 | 126.958333 | 64.375000 | 20.000000 | 111.166667 | 34.016667 | 0.589583 | 0.5 |
| **32** | 4.437500 | 116.312500 | 70.062500 | 18.187500 | 35.812500 | 32.318750 | 0.613250 | 0.5 |
| **33** | 4.058824 | 122.882353 | 65.647059 | 21.705882 | 85.588235 | 32.335294 | 0.734176 | 0.5 |
| **34** | 5.857143 | 131.857143 | 74.000000 | 18.714286 | 148.071429 | 31.164286 | 0.649857 | 0.2 |
| **35** | 5.000000 | 121.400000 | 75.600000 | 22.600000 | 75.000000 | 33.780000 | 0.454000 | 0.5 |
| **36** | 5.187500 | 132.437500 | 69.125000 | 19.187500 | 65.812500 | 31.718750 | 0.472875 | 0.6 |
| **37** | 5.263158 | 130.157895 | 75.947368 | 18.315789 | 59.263158 | 32.078947 | 0.414632 | 0.3 |
| **38** | 6.875000 | 121.125000 | 71.125000 | 19.625000 | 33.500000 | 35.568750 | 0.413938 | 0.6 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **39** | 7.416667 | 126.750000 | 72.666667 | 26.083333 | 72.416667 | 31.983333 | 0.605917 | 0.2 |
| **40** | 6.230769 | 130.923077 | 69.230769 | 24.230769 | 72.307692 | 33.538462 | 0.376077 | 0.4 |
| **41** | 6.500000 | 129.090909 | 67.590909 | 17.409091 | 38.818182 | 35.259091 | 0.396273 | 0.5 |
| **42** | 6.888889 | 109.555556 | 73.388889 | 19.222222 | 61.277778 | 34.983333 | 0.388000 | 0.3 |
| **43** | 7.769231 | 133.000000 | 78.461538 | 27.846154 | 125.153846 | 36.892308 | 0.450846 | 0.8 |
| **44** | 7.250000 | 124.375000 | 61.750000 | 4.625000 | 32.250000 | 34.162500 | 0.668375 | 0.6 |
| **45** | 7.333333 | 131.200000 | 83.066667 | 20.600000 | 31.133333 | 34.960000 | 0.496467 | 0.5 |
| **46** | 6.384615 | 105.923077 | 76.000000 | 24.153846 | 112.307692 | 34.523077 | 0.426846 | 0.5 |
| **47** | 8.333333 | 137.000000 | 78.333333 | 14.500000 | 49.166667 | 34.566667 | 0.355333 | 0.6 |
| **48** | 8.800000 | 107.600000 | 78.400000 | 23.400000 | 52.000000 | 29.980000 | 0.456800 | 0.2 |
| **49** | 7.600000 | 153.000000 | 81.400000 | 21.600000 | 55.200000 | 32.020000 | 0.612000 | 0.6 |
| **50** | 6.750000 | 138.250000 | 78.250000 | 16.000000 | 26.375000 | 31.225000 | 0.470125 | 0.6 |
| **51** | 8.625000 | 147.625000 | 84.500000 | 21.875000 | 129.375000 | 33.975000 | 0.615250 | 0.6 |
| **52** | 4.625000 | 133.000000 | 81.500000 | 13.375000 | 94.500000 | 33.475000 | 0.505375 | 0.8 |
| **53** | 5.400000 | 158.000000 | 79.000000 | 21.200000 | 183.000000 | 30.500000 | 0.550600 | 0.8 |
| **54** | 7.000000 | 140.333333 | 89.333333 | 8.833333 | 61.000000 | 30.800000 | 0.465500 | 0.6 |
| **55** | 5.500000 | 140.750000 | 70.250000 | 16.250000 | 83.750000 | 27.025000 | 0.226500 | 0.2 |
| **56** | 8.000000 | 98.333333 | 76.333333 | 32.333333 | 69.000000 | 31.700000 | 0.936667 | 0.6 |
| **57** | 8.800000 | 137.800000 | 76.800000 | 9.600000 | 78.000000 | 29.700000 | 0.704000 | 0.2 |
| **58** | 7.142857 | 135.142857 | 78.285714 | 19.285714 | 167.857143 | 32.428571 | 0.554714 | 0.4 |
| **59** | 2.333333 | 173.333333 | 74.000000 | 16.666667 | 282.000000 | 26.966667 | 0.252667 | 0.6 |
| **60** | 6.000000 | 146.400000 | 80.000000 | 20.000000 | 164.200000 | 28.740000 | 0.436800 | 0.4 |
| **61** | 5.500000 | 144.000000 | 76.000000 | 16.500000 | 95.000000 | 30.000000 | 0.613000 | 0.5 |
| **62** | 3.750000 | 139.500000 | 71.500000 | 29.000000 | 0.000000 | 28.950000 | 0.565500 | 0.5 |
| **63** | 5.500000 | 133.250000 | 78.000000 | 23.500000 | 45.000000 | 30.775000 | 0.249250 | 0.0 |
| **64** | 8.000000 | 120.000000 | 78.000000 | 0.000000 | 0.000000 | 25.000000 | 0.409000 | 0.0 |
| **65** | 3.333333 | 137.000000 | 78.666667 | 12.333333 | 0.000000 | 31.600000 | 0.259000 | 0.0 |
| **66** | 5.000000 | 157.000000 | 86.000000 | 0.000000 | 0.000000 | 30.375000 | 0.408500 | 0.5 |
| **67** | 4.000000 | 132.333333 | 72.666667 | 0.000000 | 0.000000 | 28.766667 | 0.602000 | 0.3 |
| **68** | 8.000000 | 91.000000 | 82.000000 | 0.000000 | 0.000000 | 35.600000 | 0.587000 | 0.0 |
| **69** | 5.000000 | 134.000000 | 81.000000 | 0.000000 | 0.000000 | 13.400000 | 0.413000 | 0.0 |
| **70** | 4.000000 | 145.000000 | 82.000000 | 18.000000 | 0.000000 | 32.500000 | 0.235000 | 1.0 |
| **72** | 2.000000 | 119.000000 | 0.000000 | 0.000000 | 0.000000 | 19.600000 | 0.832000 | 0.0 |
| **81** | 9.000000 | 134.000000 | 74.000000 | 33.000000 | 60.000000 | 25.900000 | 0.460000 | 0.0 |

In [62]:
```
data['Outcome'].value_counts()
```

Out[62]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

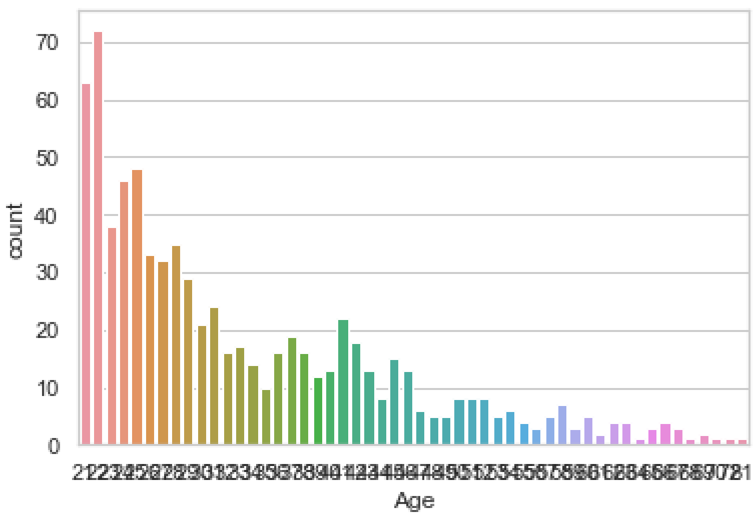
## 0 means no DIABETED

## 1 means patient with DIABETED

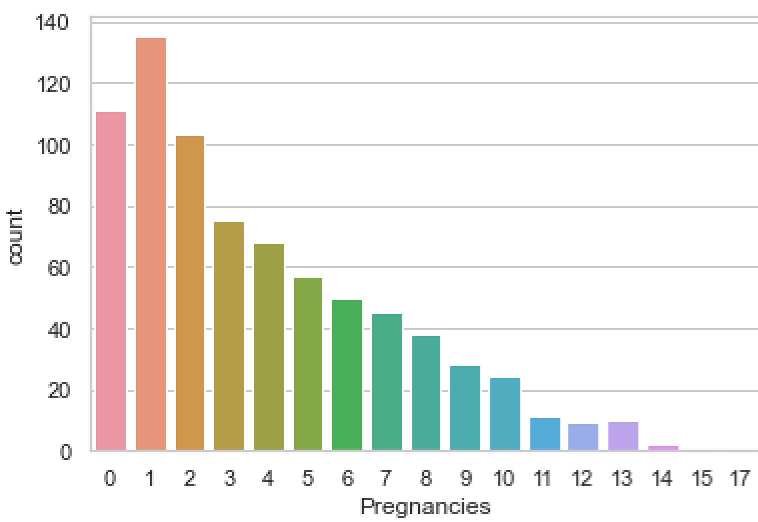
# Data Analysis:

## Countplot:

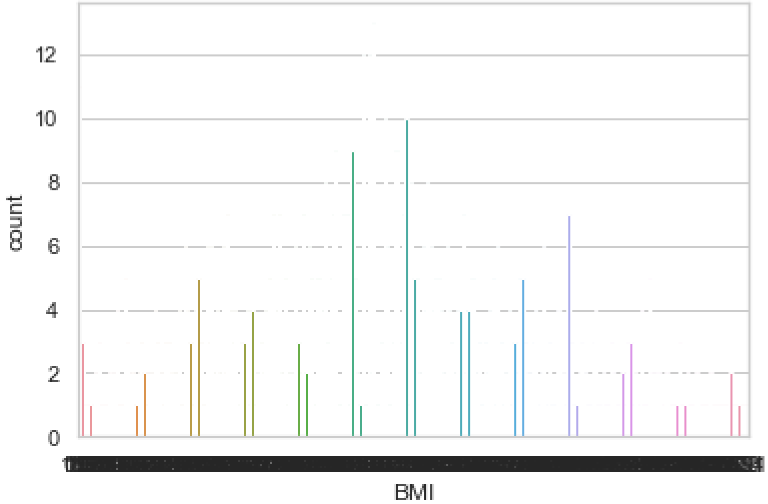
In [63]:
```
sns.countplot(data['Age'])
plt.show()
```



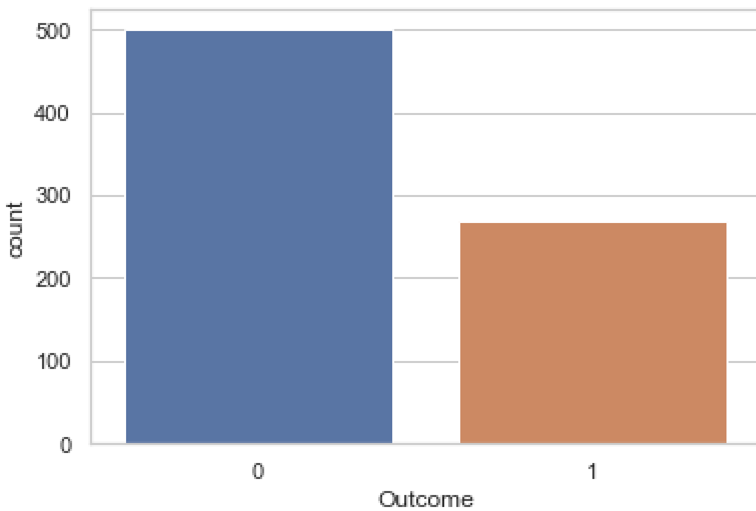

In [64]:
```
sns.countplot(data['Pregnancies'])
plt.show()
```




In [65]:
```
sns.countplot(data['BMI'])
plt.show()
```



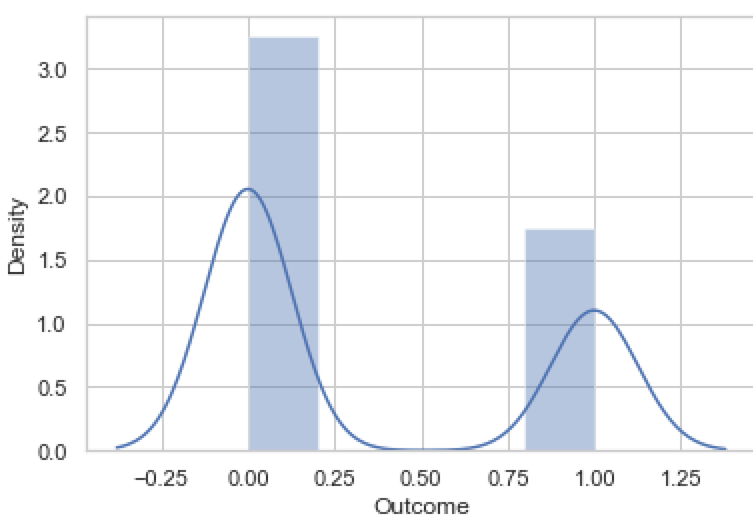

In [66]:
```
sns.countplot(data['Outcome'])
plt.show()
```




# Distplot:

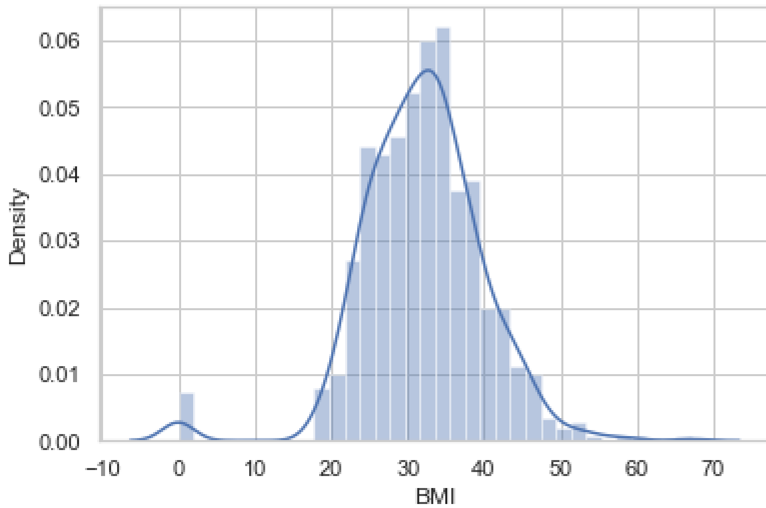
In [67]:
```
sns.distplot(data['Outcome'])
```

Out[67]:
```
<AxesSubplot:xlabel='Outcome', ylabel='Density'>
```



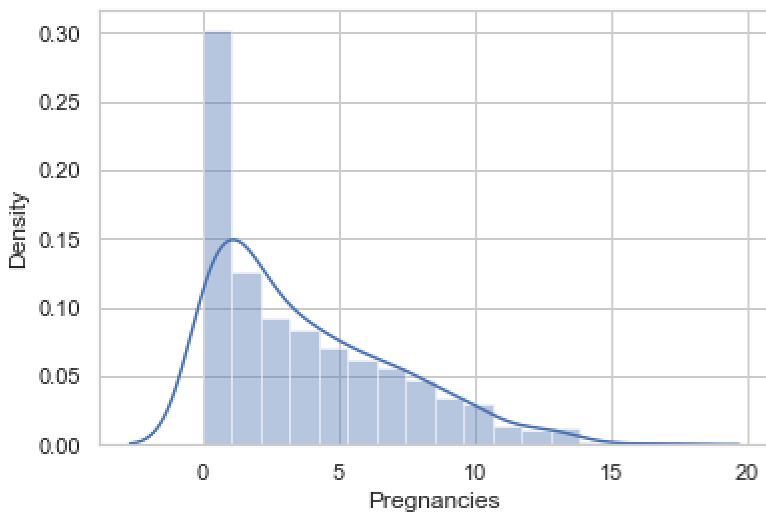

In [68]:
```
sns.distplot(data['BMI'])
```

Out[68]: <AxesSubplot:xlabel='BMI', ylabel='Density'>



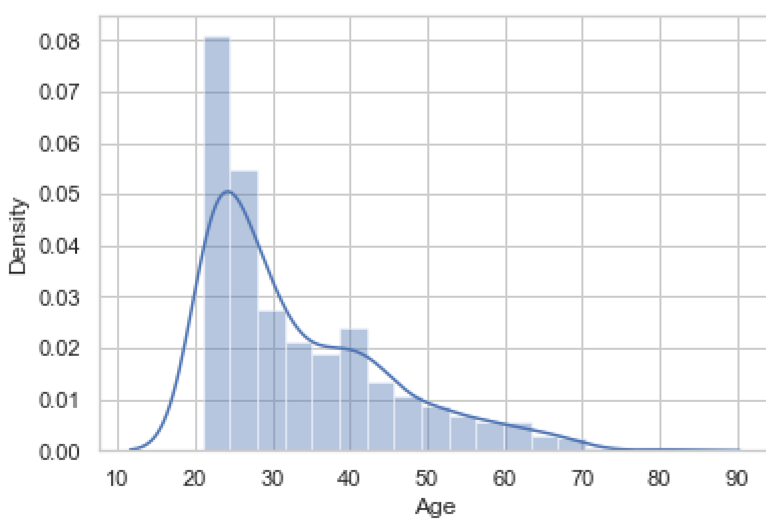

In [69]:
```
sns.distplot(data['Pregnancies'])
```

Out[69]: <AxesSubplot:xlabel='Pregnancies', ylabel='Density'>



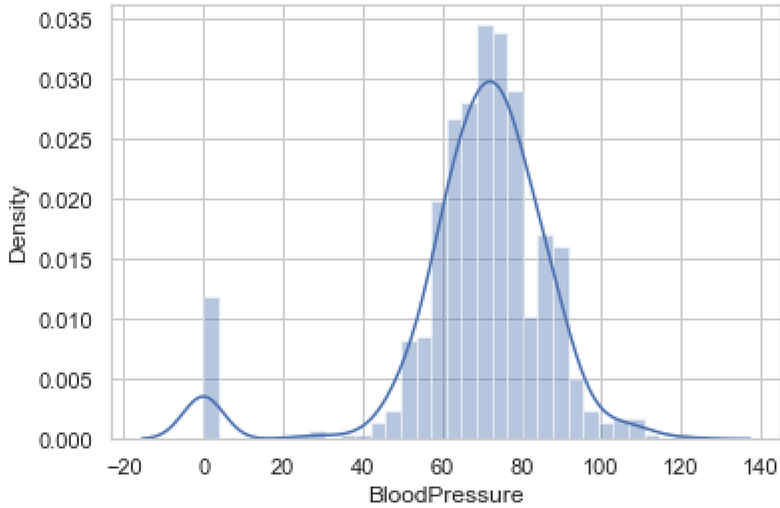

In [70]:
```
sns.distplot(data['Age'])
```

Out[70]: <AxesSubplot:xlabel='Age', ylabel='Density'>



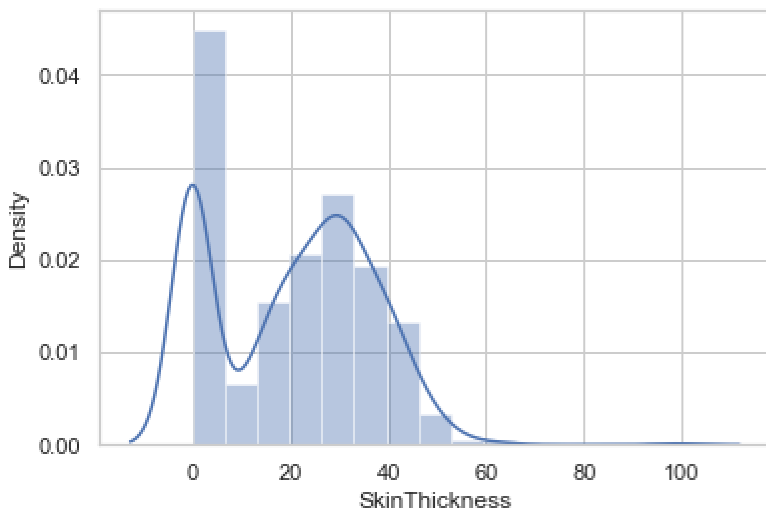

In [71]:
```
sns.distplot(data['BloodPressure'])
```

Out[71]: <AxesSubplot:xlabel='BloodPressure', ylabel='Density'>




In [72]:
```
sns.distplot(data['SkinThickness'])
```

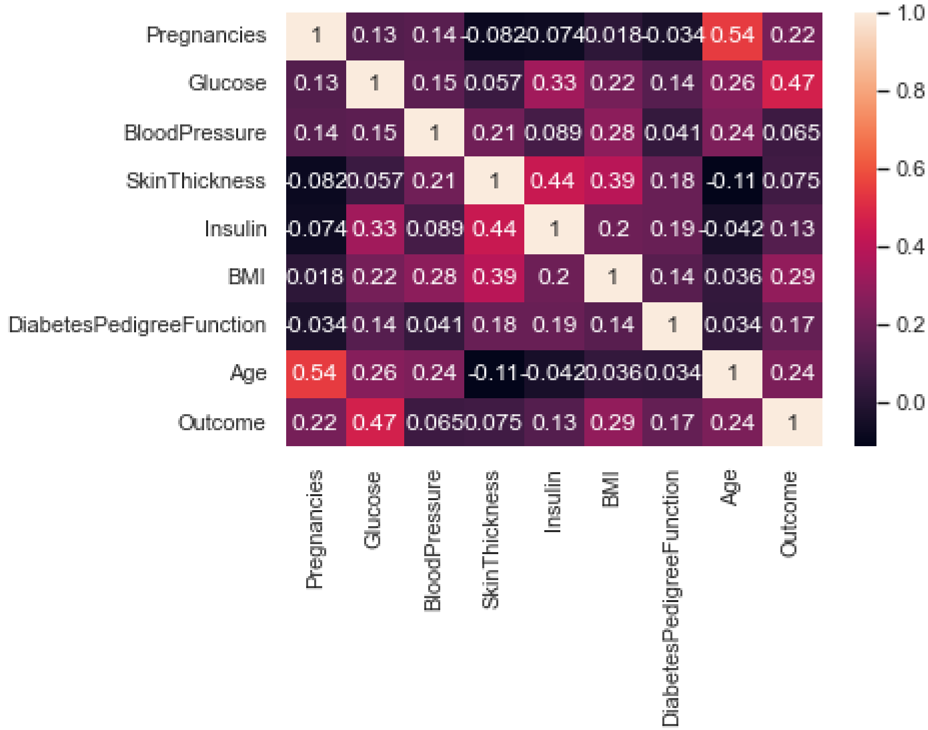
Out[72]:
```
<AxesSubplot:xlabel='SkinThickness', ylabel='Density'>
```




# Heatmap for expressing correlation

In [76]:
```
corr = data.corr()
sns.heatmap(corr,annot=True)
```

Out[76]:
```
<AxesSubplot:>
```




# Feature Selection

In [83]:

```
#lets extract features and targets
X = data.drop(columns=['Outcome'])
Y = data['Outcome']
print("Features Extraction Sucessfull")
```

```
Features Extraction Sucessfull
```

# Feature Importance

In [84]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()

from sklearn.ensemble import ExtraTreesClassifier
classifiern = ExtraTreesClassifier()
classifiern.fit(X,Y)
score = classifiern.feature_importances_
print(score)
```

```
[0.1098642  0.23582965 0.09835589 0.07997468 0.075298   0.14356218
 0.11860896 0.13850643]
```

# Splitting Dataset

In [85]:

```
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)
```

# Using Logistic Regression

In [86]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score,confusion_matrix
print("Accuracy Score:",accuracy_score(Y_test,Y_pred))
```

```
Accuracy Score: 0.8181818181818182
```

In [87]:

```
confusion_mat = confusion_matrix(Y_test,Y_pred)
print(confusion_mat)
```

```
[[105   9]
 [ 19  21]]
```

# Using KNN

In [88]:

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

```
Accuracy Score: 0.7272727272727273
```

# Using SVC

In [89]:

```
from sklearn.svm import SVC
model = SVC()
model.fit(X_train,Y_train)
pred_y = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,pred_y))
```

```
Accuracy Score: 0.8116883116883117
```

# Using Decision Tree

In [90]:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='entropy',random_state=7)
model.fit(X_train,Y_train)
y_pred = model.predict(X_test)

from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(Y_test,y_pred))
```

```
Accuracy Score: 0.7012987012987013
```