In [37]:
```
import pandas as pd
df = pd.read_csv('diabetes.csv')
df
```

Out[37]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	720	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 [38]:
```
df.isna().sum()
```

Out[38]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         2
Outcome                     0
dtype: int64
```

In [39]:
```
mean_age = round(df['Age'].mean())
mean_age
```

Out[39]:
```
33
```

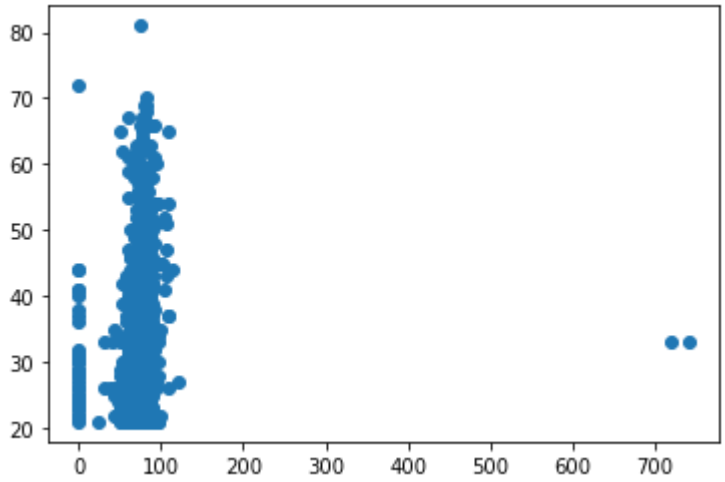
In [40]:
```
df['Age'].fillna(mean_age, inplace=True)
```

In [41]:
```
df.isna().sum()
```

Out[41]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [42]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(df['BloodPressure'], df['Age'])
```

Out[42]:
```
<matplotlib.collections.PathCollection at 0x1a8ba5509a0>
```



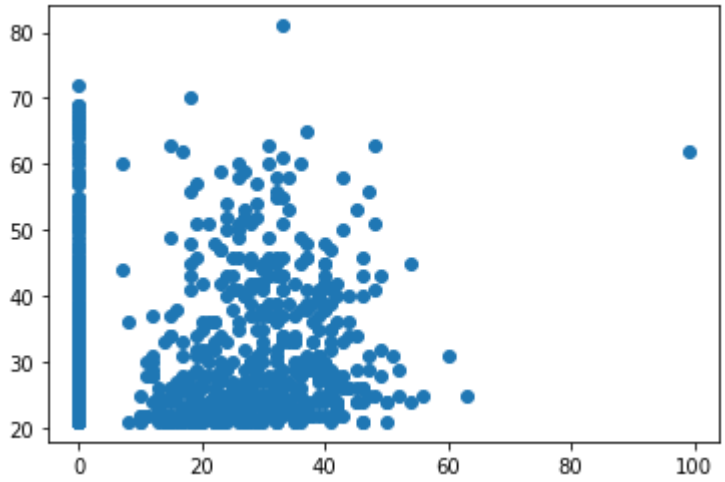
In [43]:
```
mean_pressure = round(df['BloodPressure'].mean())
for x in range(0,768):
    if df['BloodPressure'][x] >= 700:
        df['BloodPressure'][x] = mean_pressure
```

```
<ipython-input-43-530d90125613>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['BloodPressure'][x] = mean_pressure
```

In [44]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(df['SkinThickness'], df['Age'])
```

Out[44]:
```
<matplotlib.collections.PathCollection at 0x1a8ba5b7220>
```



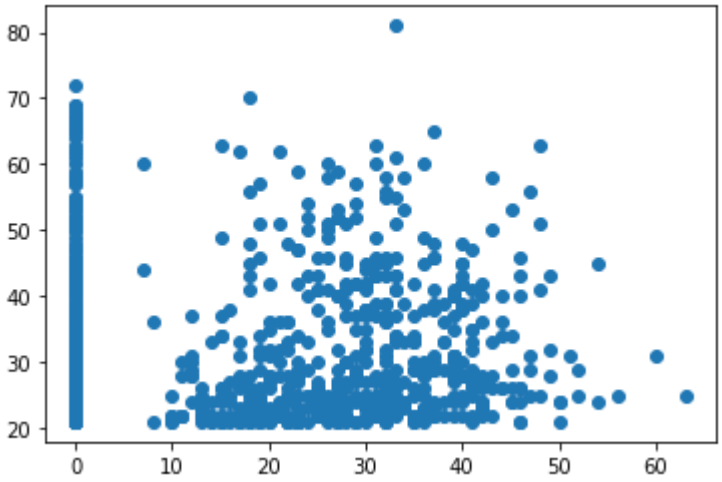
In [45]:
```
mean_skin = round(df['SkinThickness'].mean())
for x in range(0,768):
    if df['SkinThickness'][x] >= 80:
        df['SkinThickness'][x] = mean_skin
```

```
<ipython-input-45-df13c6698117>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['SkinThickness'][x] = mean_skin
```

In [46]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(df['SkinThickness'], df['Age'])
```

Out[46]:
```
<matplotlib.collections.PathCollection at 0x1a8ba606ca0>
```



In [47]:
```
x = df.drop(['Outcome'], axis=1)
y = df['Outcome']
```

In [48]:
```
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_stat
```

In [49]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [50]:
```
model.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
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.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[50]:
```
LogisticRegression()
```

In [51]:
```
y_pred = model.predict(x_test)
y_pred
```

Out[51]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
      dtype=int64)
```

In [52]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
cm
```

Out[52]:
```
array([[79, 20],
       [18, 37]], dtype=int64)
```

In [53]:
```
accuracy_score(y_test, y_pred)
```

Out[53]:
```
0.7532467532467533
```