In [1]:
```
%config IPCompleter.greedy=True
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import seaborn as sns
%matplotlib inline
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics


from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
C:\Users\kietd\Anaconda3\lib\site-packages\sklearn\externals\six.py:3
1: DeprecationWarning: The module is deprecated in version 0.21 and w
ill be removed in version 0.23 since we've dropped support for Python
2.7. Please rely on the official version of six (https://pypi.org/pro
ject/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

0. Compute the mean value and standard deviation for attributes 2-6. Remove 0's that do not make sense prior to computing these statistics.

In [2]:
```
diabetes = pd.read_csv('diabetes.csv')

diabetes= diabetes[diabetes['Glucose'] != 0]
diabetes= diabetes[diabetes['BloodPressure'] != 0]
diabetes= diabetes[diabetes['SkinThickness'] != 0]
diabetes= diabetes[diabetes['Insulin'] != 0]
diabetes= diabetes[diabetes['BMI'] != 0]
diabetes
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2
6	3	78	50	32	88	31.0	0
8	2	197	70	45	543	30.5	0
13	1	189	60	23	846	30.1	0
14	5	166	72	19	175	25.8	0
16	0	118	84	47	230	45.8	0
18	1	103	30	38	83	43.3	0
19	1	115	70	30	96	34.6	0
20	3	126	88	41	235	39.3	0
24	11	143	94	33	146	36.6	0
25	10	125	70	26	115	31.1	0
27	1	97	66	15	140	23.2	0
28	13	145	82	19	110	22.2	0
31	3	158	76	36	245	31.6	0
32	3	88	58	11	54	24.8	0
35	4	103	60	33	192	24.0	0
39	4	111	72	47	207	37.1	1
40	3	180	64	25	70	34.0	0
43	9	171	110	24	240	45.4	0
50	1	103	80	11	82	19.4	0
51	1	101	50	15	36	24.2	0
52	5	88	66	21	23	24.4	0
53	8	176	90	34	300	33.7	0
54	7	150	66	42	342	34.7	0
56	7	187	68	39	304	37.7	0
57	0	100	88	60	110	46.8	0
59	0	105	64	41	142	41.5	0
63	2	141	58	34	128	25.4	0
68	1	95	66	13	38	19.6	0
...	...	...	...	...	...	...	
707	2	127	46	21	335	34.4	0
709	2	93	64	32	160	38.0	0
710	3	158	64	13	387	31.2	0
711	5	126	78	27	22	29.6	0

In [3]:
```
diabetes_26 = pd.read_csv('diabetes.csv', usecols = ['Glucose','BloodPr
essure','SkinThickness','Insulin','BMI'])
diabetes_26.describe()
```

Out[3]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	120.894531	69.105469	20.536458	79.799479	31.992578
std	31.972618	19.355807	15.952218	115.244002	7.884160
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	99.000000	62.000000	0.000000	0.000000	27.300000
50%	117.000000	72.000000	23.000000	30.500000	32.000000
75%	140.250000	80.000000	32.000000	127.250000	36.600000
max	199.000000	122.000000	99.000000	846.000000	67.100000

1. Compute the covariance matrix for attributes 2-6 next, compute the correlations for each of the 10 pairs of the 5 attributes. Interpret the statistical findings! Remove 0's that do not make sense prior to computing the covariance matrix and correlations.

In [4]:
```
print('CORRELATION MATRIX')
print(diabetes_26.corr())
```

```
CORRELATION MATRIX
                Glucose  BloodPressure  SkinThickness   Insulin
BMI
Glucose        1.000000       0.152590       0.057328  0.331357  0.22
1071
BloodPressure  0.152590       1.000000       0.207371  0.088933  0.28
1805
SkinThickness  0.057328       0.207371       1.000000  0.436783  0.39
2573
Insulin        0.331357       0.088933       0.436783  1.000000  0.19
7859
BMI            0.221071       0.281805       0.392573  0.197859  1.00
0000
```

In [5]:
```
print('COVARIANCE MATRIX')
diabetes_26.cov()
```

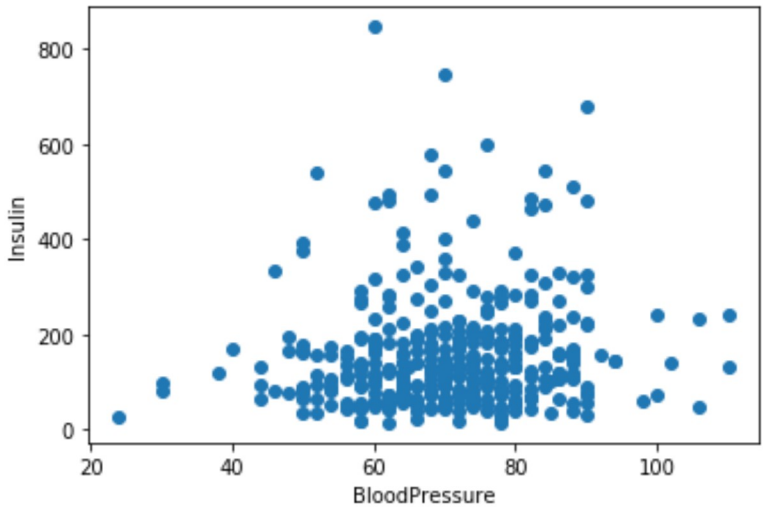
```
COVARIANCE MATRIX
```

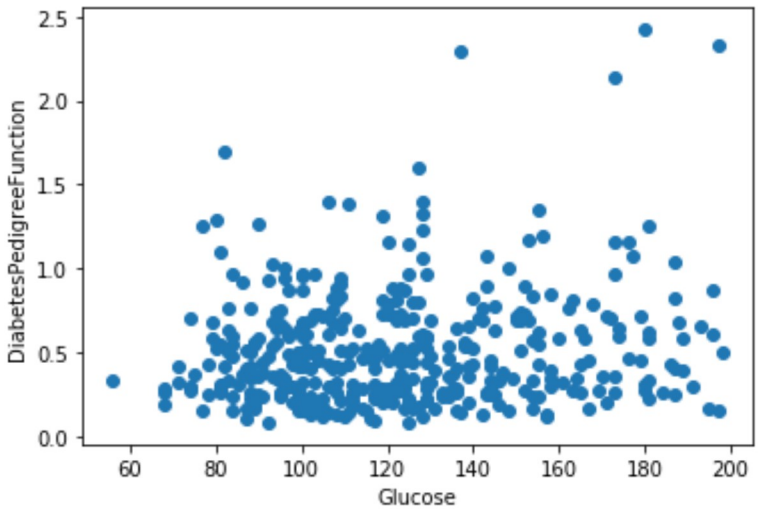
Out[5]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
Glucose	1022.248314	94.430956	29.239183	1220.935799	55.726987
BloodPressure	94.430956	374.647271	64.029396	198.378412	43.004695
SkinThickness	29.239183	64.029396	254.473245	802.979941	49.373869
Insulin	1220.935799	198.378412	802.979941	13281.180078	179.775172
BMI	55.726987	43.004695	49.373869	179.775172	62.159984

2. Create a scatter plot for attributes 3 and 6 of your dataset and a second scatter plot for attributes 2 and 7. Interpret the two scatter plots!

In [6]:
```
plt.scatter(diabetes.iloc[:,2:3],diabetes.iloc[:,4:5])
plt.xlabel('BloodPressure')
plt.ylabel('Insulin')
plt.show()

plt.clf()

plt.scatter(diabetes.iloc[:,1:2],diabetes.iloc[:,6:7])
plt.xlabel('Glucose')
plt.ylabel('DiabetesPedigreeFunction')
plt.show()
```

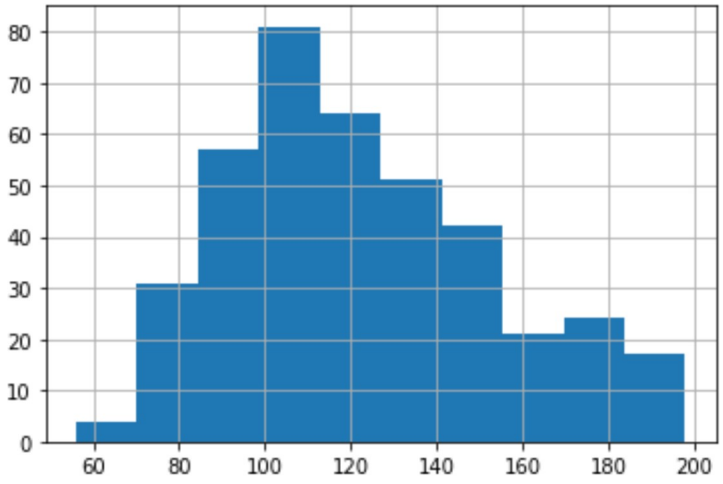




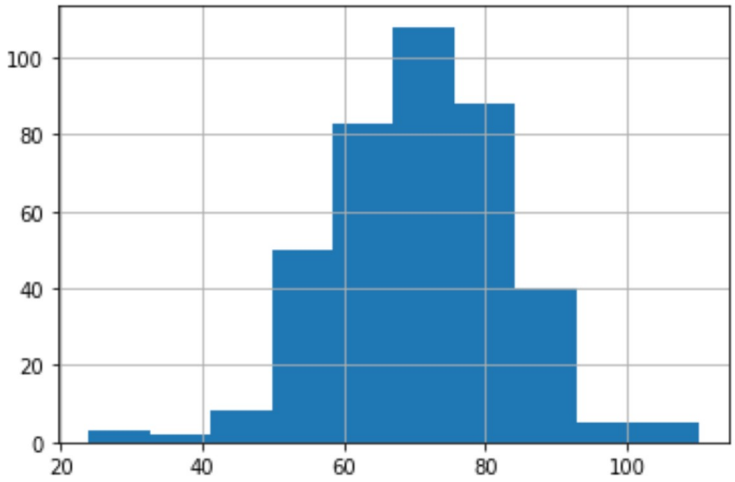
3. Create histograms for attributes 2, 3 and 6. Then create the same histograms for the 3 attributes for the instances of class 1 and for the instances of class 0; interpret the obtained 9 histograms.

Histograms without condition

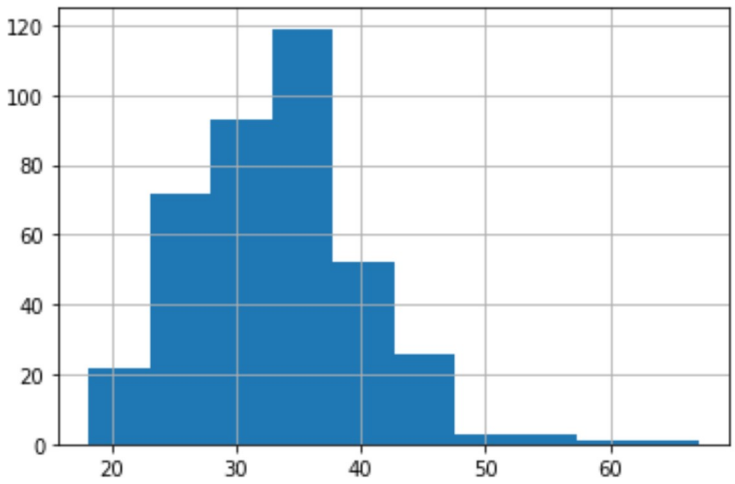
In [7]:
```
diabetes['Glucose'].hist()
plt.show()
plt.clf()
print('\t\tGlucose')
diabetes['BloodPressure'].hist()
plt.show()
plt.clf()
print('\t\tBloodPressure')
diabetes['BMI'].hist()
plt.show()
plt.clf()
print('\t\tBMI')
```




```
Glucose
```



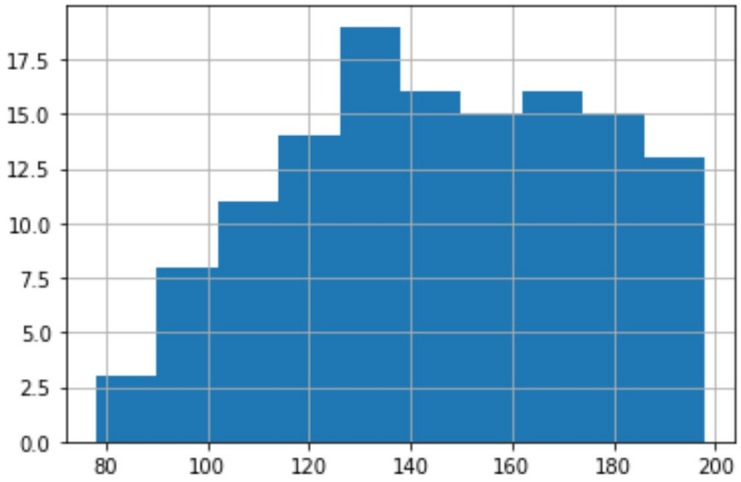

```
BloodPressure
```



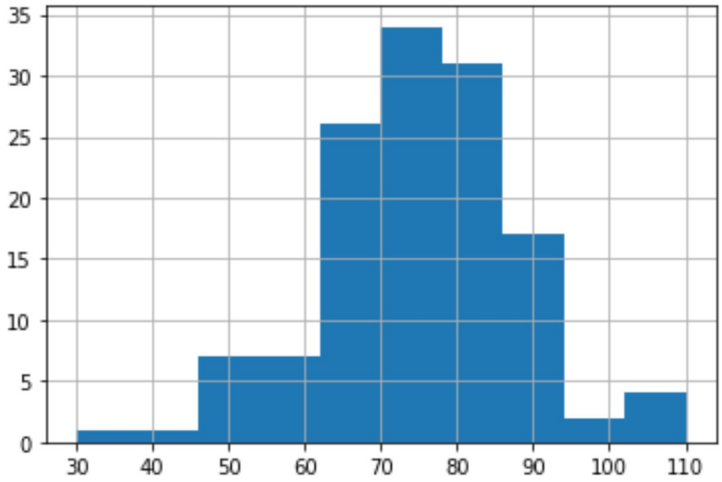

```
BMI
<Figure size 432x288 with 0 Axes>
```

Histograms with class 1

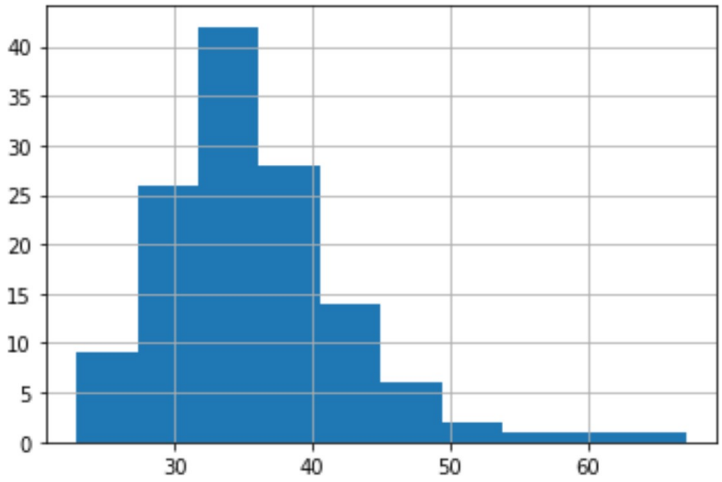
In [8]:
```
diabetes_class_1 = diabetes[diabetes['Outcome']==1]
diabetes_class_1['Glucose'].hist()
plt.show()
plt.clf()
print('\t\tGlucose')
diabetes_class_1['BloodPressure'].hist()
plt.show()
plt.clf()
print('\t\tBloodPressure')
diabetes_class_1['BMI'].hist()
plt.show()
plt.clf()
print('\t\tBMI')
```



Glucose



BloodPressure

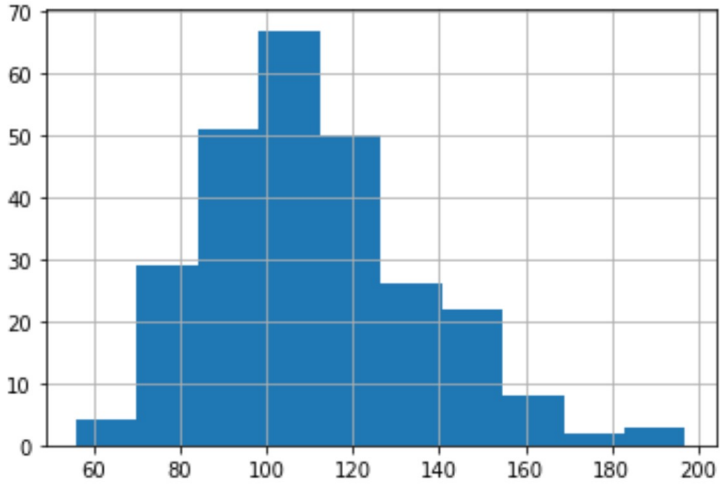


BMI

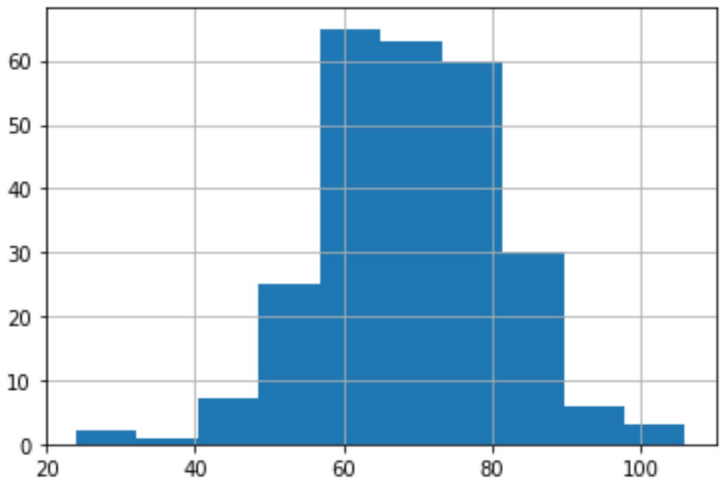
<Figure size 432x288 with 0 Axes>

Histograms with class 0

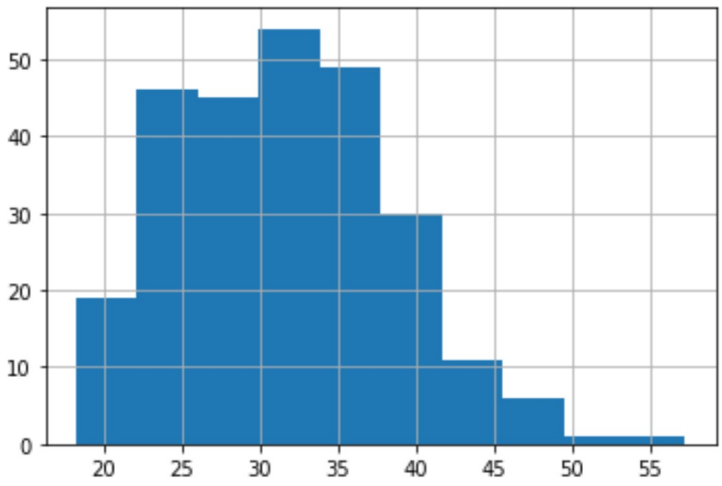
In [9]:
```
diabetes_class_0 = diabetes[diabetes['Outcome']==0]
diabetes_class_0['Glucose'].hist()
plt.show()
plt.clf()
print('\t\tGlucose')
diabetes_class_0['BloodPressure'].hist()
plt.show()
plt.clf()
print('\t\tBloodPressure')
diabetes_class_0['BMI'].hist()
plt.show()
plt.clf()
print('\t\tBMI')
```



Glucose



BloodPressure



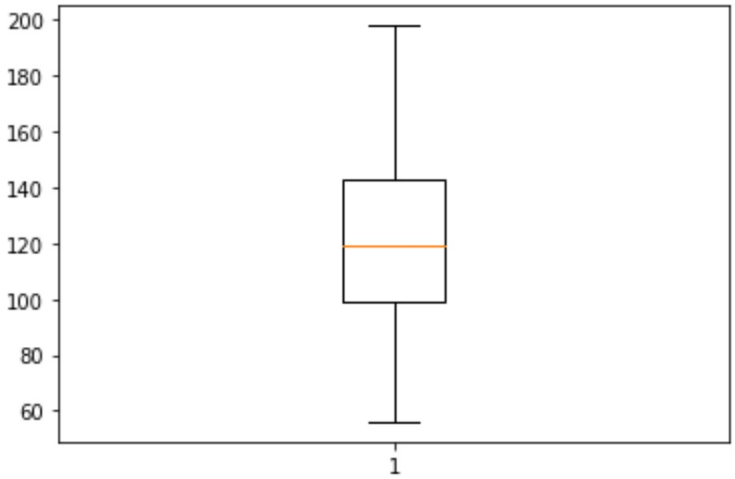
BMI

<Figure size 432x288 with 0 Axes>

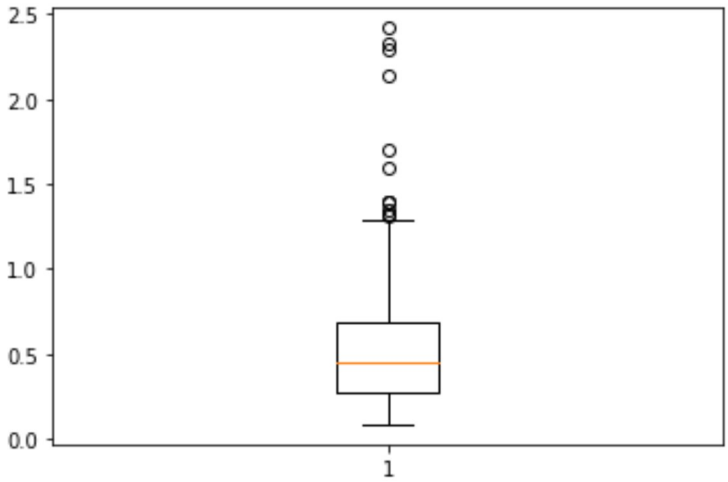
4. Create box plots for the 2nd , 7th and 8th attribute; one for the whole dataset and one each for the instances of the two classes. Remove 0's that do not make sense prior to computing the box plots. Interpret and compare the obtained 9 boxplots!

Box plots wihout condition

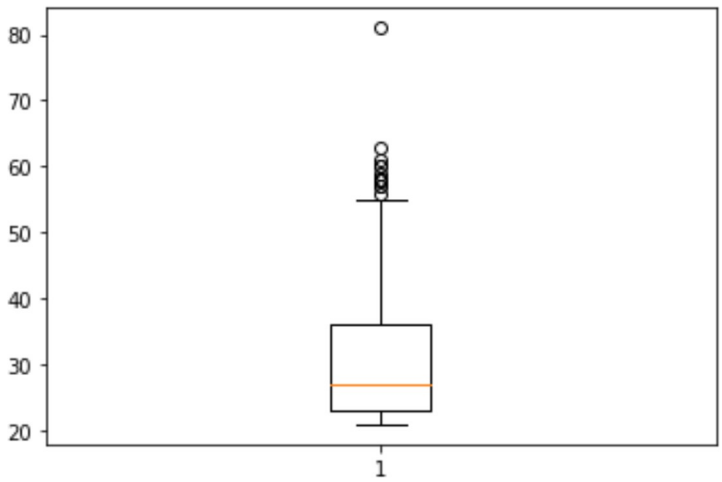
In [10]:
```
plt.boxplot(diabetes.Glucose)
plt.show()
plt.clf()
print('\t\tGlucose')
plt.boxplot(diabetes.DiabetesPedigreeFunction)
plt.show()
plt.clf()
print('\t\tDiabetesPedigreeFunction')
plt.boxplot(diabetes.Age)
plt.show()
plt.clf()
print('\t\tAge')
```




```
Glucose
```




```
DiabetesPedigreeFunction
```

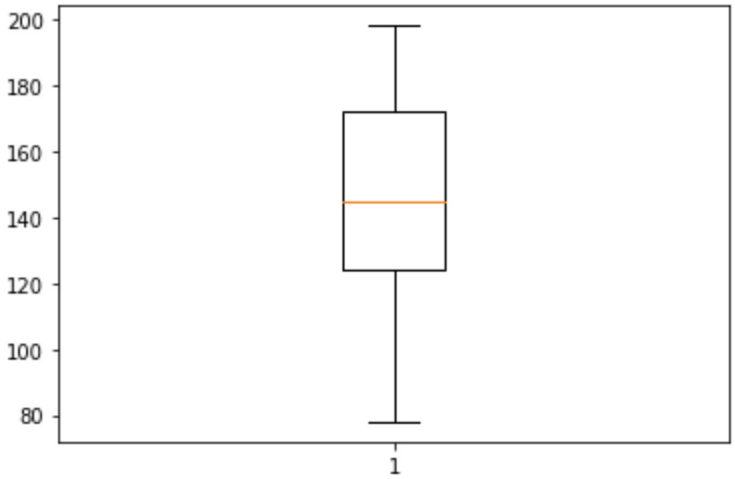



```
Age
```

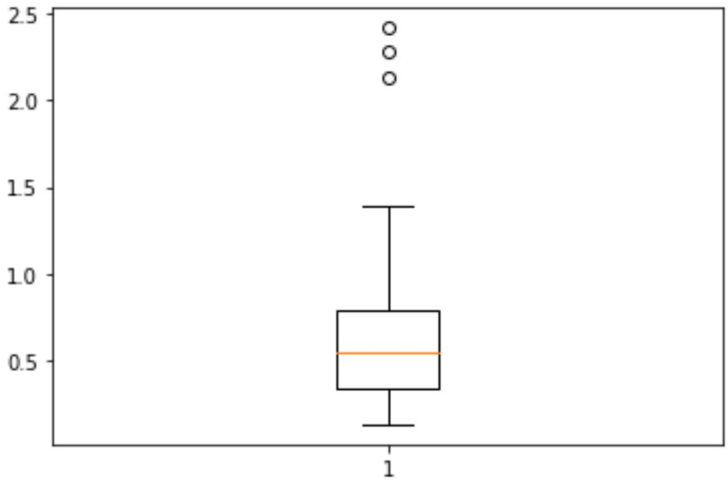
```
<Figure size 432x288 with 0 Axes>
```

Box plots with class 1

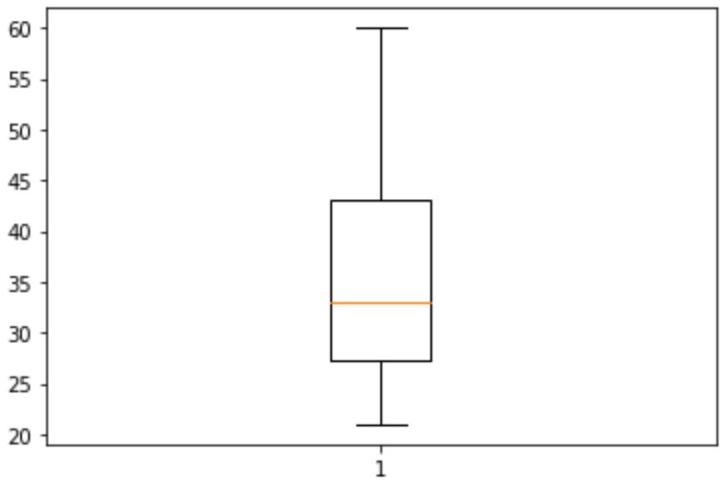
In [11]:
```
plt.boxplot(diabetes_class_1.Glucose)
plt.show()
plt.clf()
print('\t\tGlucose')
plt.boxplot(diabetes_class_1.DiabetesPedigreeFunction)
plt.show()
plt.clf()
print('\t\tDiabetesPedigreeFunction')
plt.boxplot(diabetes_class_1.Age)
plt.show()
plt.clf()
print('\t\tAge')
```




```
Glucose
```




```
DiabetesPedigreeFunction
```

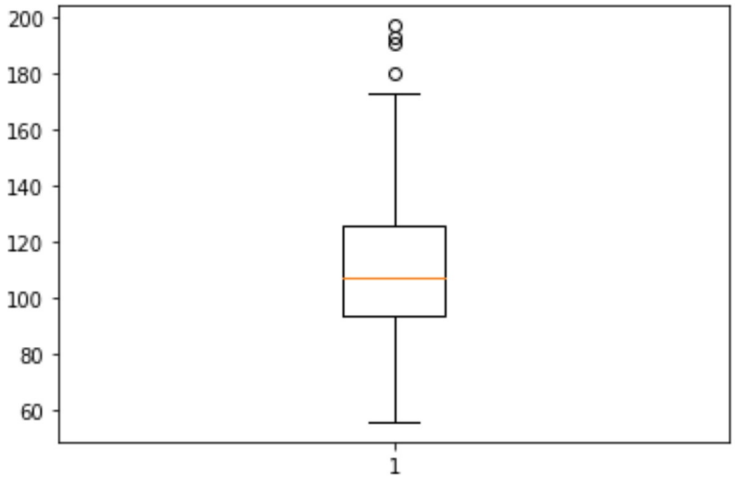



```
Age
```

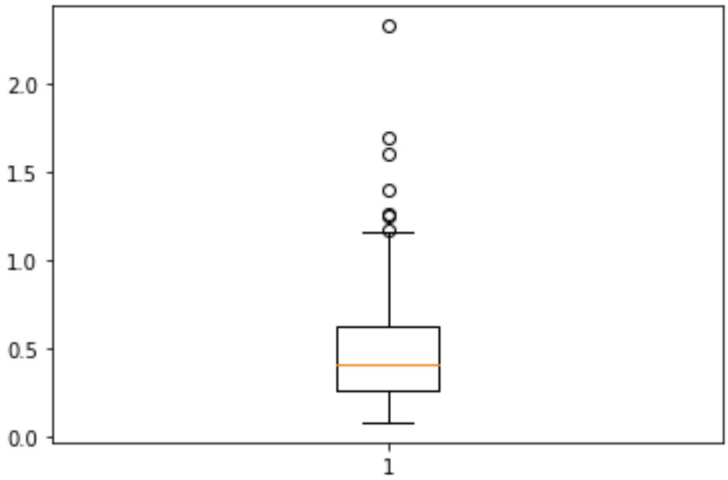
```
<Figure size 432x288 with 0 Axes>
```

Box plots with class 0

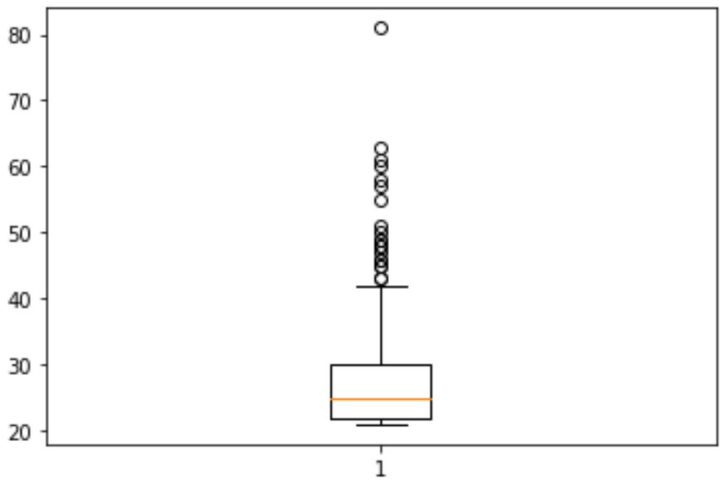
In [12]:
```
plt.boxplot(diabetes_class_0.Glucose)
plt.show()
plt.clf()
print('\t\tGlucose')
plt.boxplot(diabetes_class_0.DiabetesPedigreeFunction)
plt.show()
plt.clf()
print('\t\tDiabetesPedigreeFunction')
plt.boxplot(diabetes_class_0.Age)
plt.show()
plt.clf()
print('\t\tAge')
```




```
Glucose
```




```
DiabetesPedigreeFunction
```



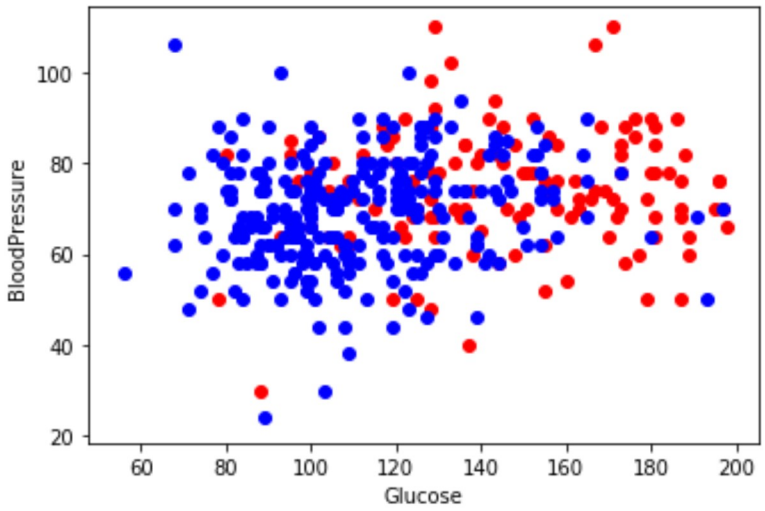

```
Age
```

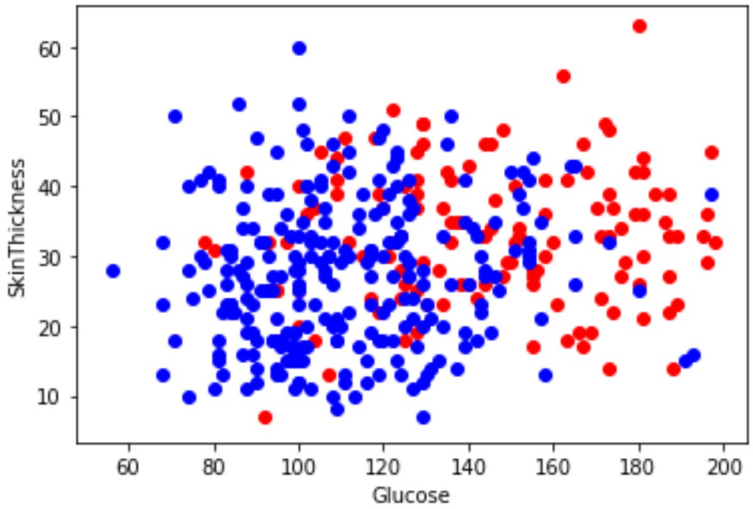
```
<Figure size 432x288 with 0 Axes>
```

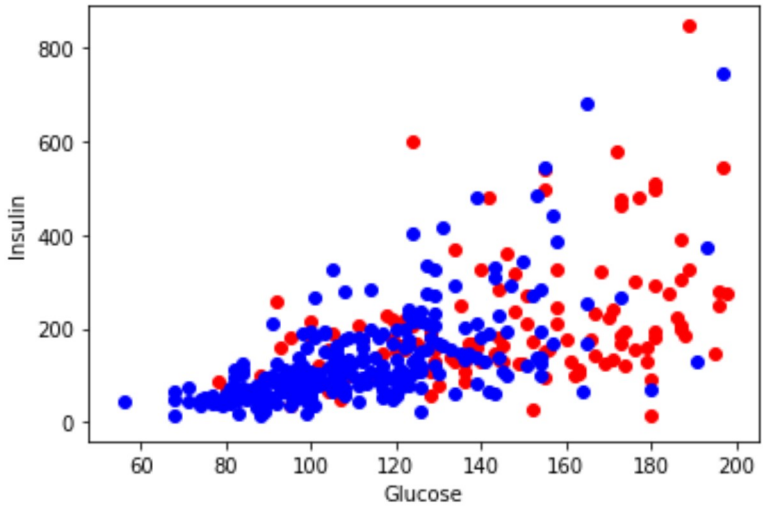
5. Create supervised scatter plots for all pairs of attributes 2-6—these are 10 plots. Next create two 3D-scatterplots: one for attributes 2, 3, 6 and one for attributes 2, 4, 6. Interpret the obtained scatter plots; in particular address what can be said about the difficulty in predicting diabetes. Assess the usefulness of the 3D scatterplot compared to the 2D plots!

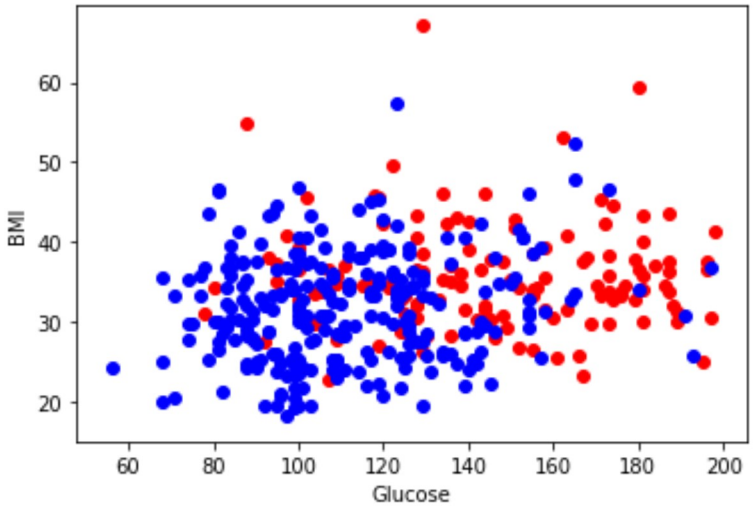
In [13]:
```
print('Class 1: red', end='\t')
print('Class 0: blue')
for i in range(1,5):
    for j in range(i+1,6):
        a = plt.scatter(diabetes_class_1.iloc[:,i:i+1],diabetes_class_
1.iloc[:,j:j+1], c='red')
        b = plt.scatter(diabetes_class_0.iloc[:,i:i+1],diabetes_class_
0.iloc[:,j:j+1], c='blue')
        plt.xlabel(diabetes_class_1.columns[i])
        plt.ylabel(diabetes_class_1.columns[j])
        plt.show()
        plt.clf()
```

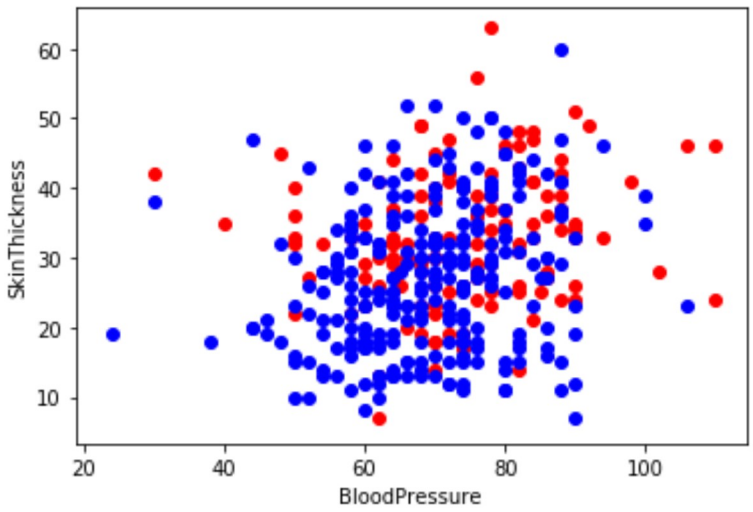
```
Class 1: red    Class 0: blue
```

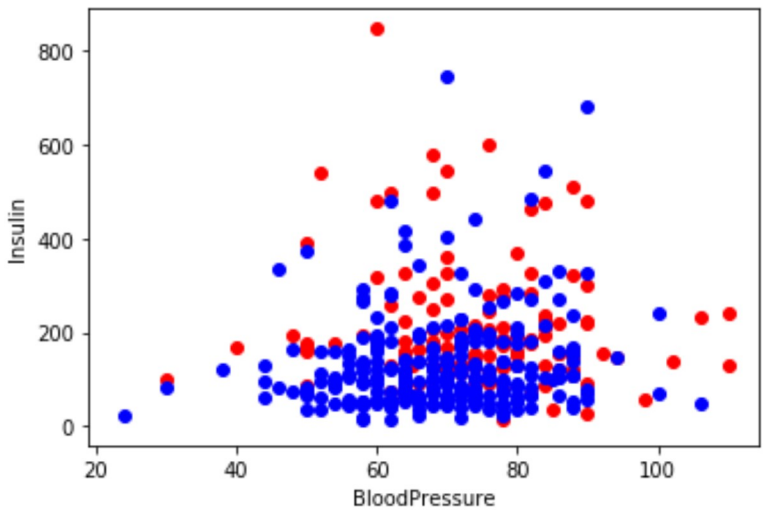


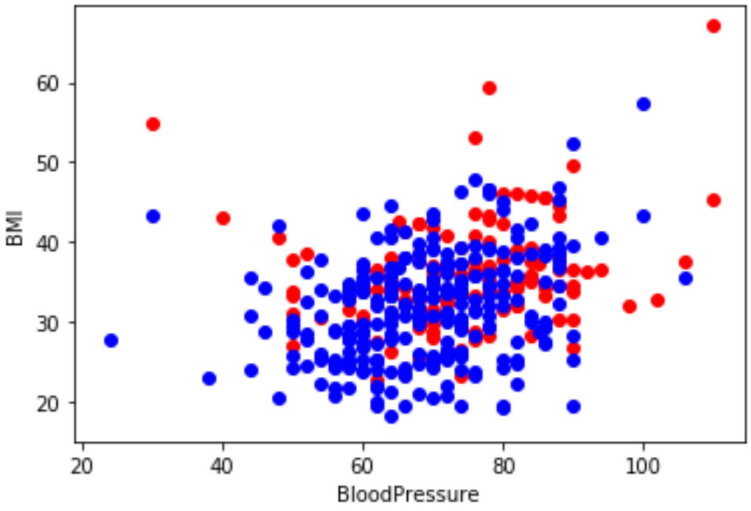


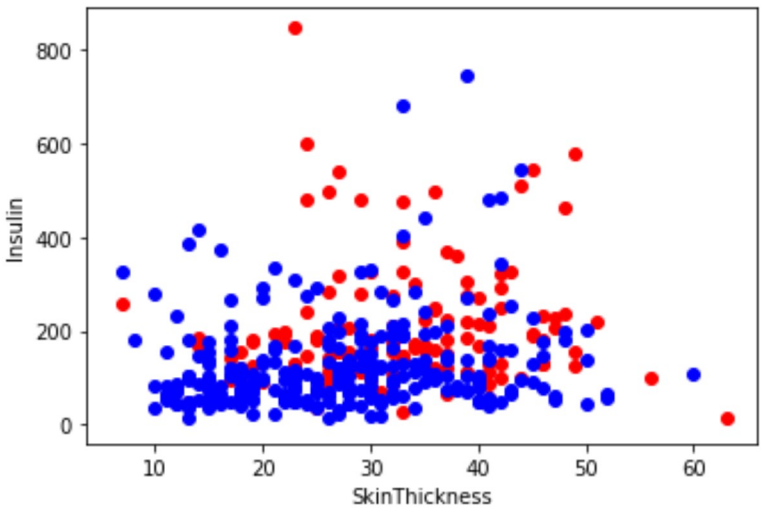


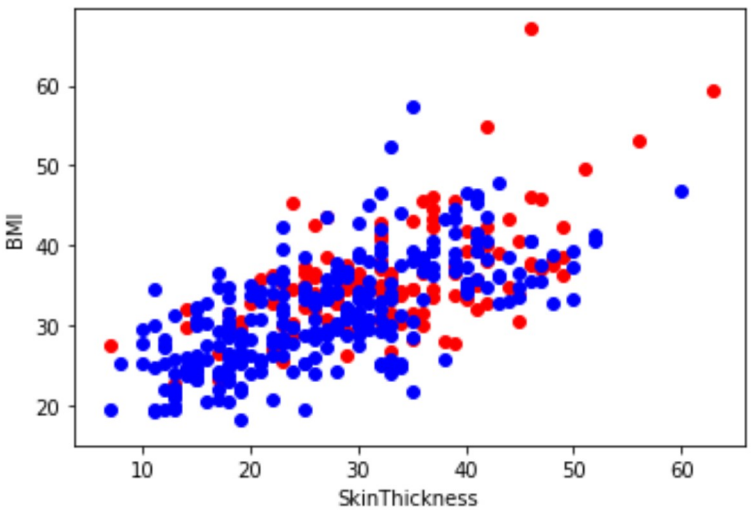

BMI
Glucose
60
50
40
30
20
60
80
100
120
140
160
180
200

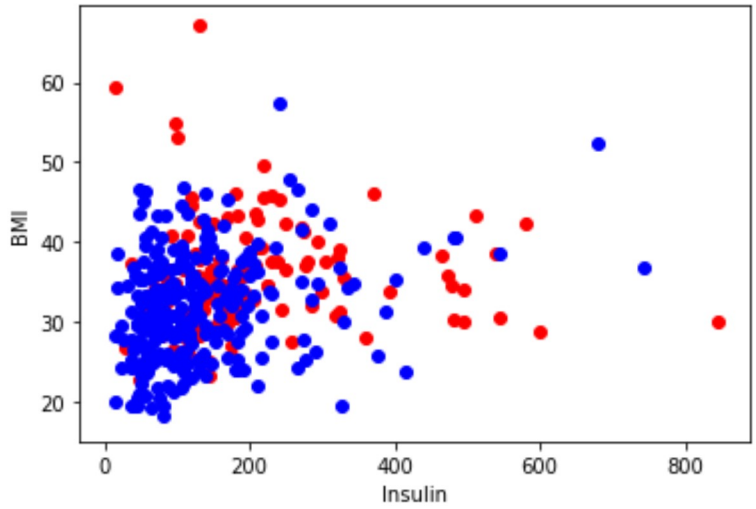

SkinThickness
BloodPressure
60
50
40
30
20
10
20
40
60
80
100


Insulin
BloodPressure
800
600
400
200
0
20
40
60
80
100


BMI
BloodPressure
20
30
40
50
60
20
40
60
80
100


Insulin
SkinThickness
0
200
400
600
800
10
20
30
40
50
60

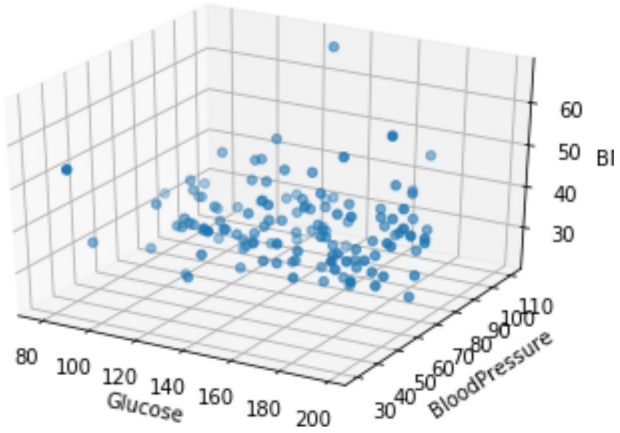

BMI
SkinThickness
20
30
40
50
60
10
20
30
40
50
60

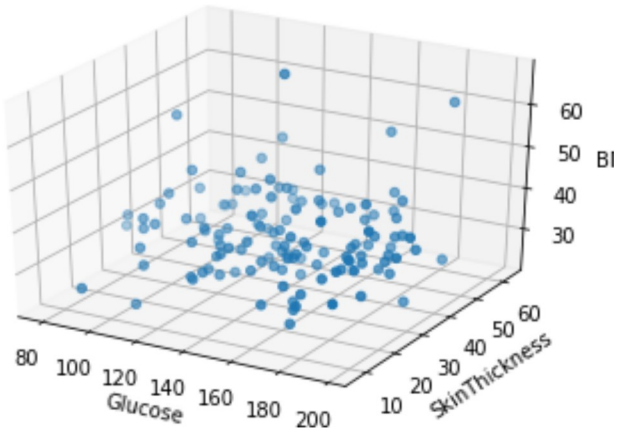



```
<Figure size 432x288 with 0 Axes>
```

In [14]:
```
graph = plt.figure().gca(projection='3d')
graph.scatter(diabetes_class_1.iloc[:,1:2],diabetes_class_1.iloc[:,2:
3], diabetes_class_1.iloc[:,5:6])
graph.set_xlabel(diabetes_class_1.columns[1])
graph.set_ylabel(diabetes_class_1.columns[2])
graph.set_zlabel(diabetes_class_1.columns[5])

graph2 = plt.figure().gca(projection='3d')
graph2.scatter(diabetes_class_1.iloc[:,1:2],diabetes_class_1.iloc[:,3:
4], diabetes_class_1.iloc[:,5:6])
graph2.set_xlabel(diabetes_class_1.columns[1])
graph2.set_ylabel(diabetes_class_1.columns[3])
graph2.set_zlabel(diabetes_class_1.columns[5])
```

Out[14]:
```
Text(0.5, 0, 'BMI')
```



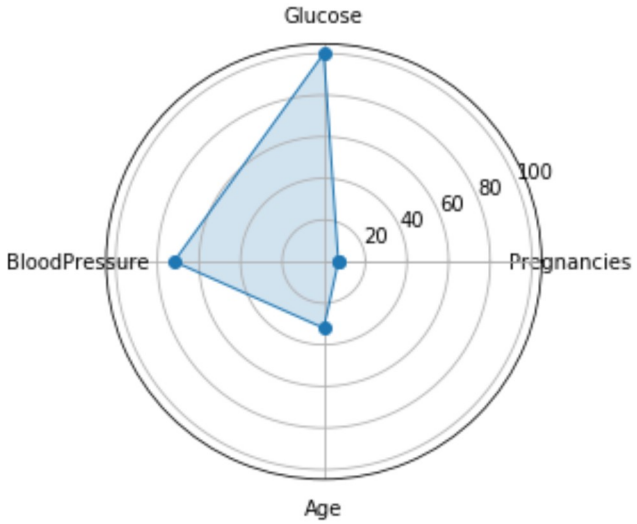


6. Create a Star plot for the first 10 instances of class 0 and the first 10 instances of class 1 for attributes 1, 2, 3, 8 in the cleaned dataset. Interpret the obtained plots.

In [15]:
```
diabetes_clean = pd.read_csv('diabetes_cleaned.csv',names = ['Pregnanci
es','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Diabetes
PedigreeFunction','Age','Outcome'])

diabetes_clean_class_0 = diabetes_clean[diabetes_clean['Outcome']==0]
diabetes_clean_class_1 = diabetes_clean[diabetes_clean['Outcome']==1]

diabetes_clean_20 = pd.concat([diabetes_clean_class_0.head(10),diabetes
_clean_class_1.head(10)])
print(diabetes_clean_20)
column_label=np.array(['Pregnancies', 'Glucose', 'BloodPressure', 'Age
'])
stats=diabetes_clean_20.loc[15,column_label].values

angles=np.linspace(0, 2*np.pi, len(column_label), endpoint=False)

stats=np.concatenate((stats,[stats[0]]))
angles=np.concatenate((angles,[angles[0]]))


fig=plt.figure()
graph = fig.add_subplot(111, polar=True)
graph.plot(angles, stats, 'o-', linewidth=1)
graph.fill(angles, stats, alpha=0.2)
graph.set_thetagrids(angles * 180/np.pi, column_label)
# graph.set_title([diabetes_clean_20.loc[15,"Pregnancies"]])
graph.grid(True)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
1	1	85	66	29	156	26.6
3	1	89	66	23	94	28.1
5	5	116	74	29	156	25.6
7	10	115	72	29	156	35.3
10	4	110	92	29	156	37.6
12	10	139	80	29	156	27.1
18	1	103	30	38	83	43.3
20	3	126	88	41	235	39.3
21	8	99	84	29	156	35.4
27	1	97	66	15	140	23.2
0	6	148	72	35	156	33.6
2	8	183	64	29	156	23.3
4	0	137	40	35	168	43.1
6	3	78	50	32	88	31.0
8	2	197	70	45	543	30.5
9	8	125	96	29	156	32.0
11	10	168	74	29	156	38.0
13	1	189	60	23	846	30.1
14	5	166	72	19	175	25.8
15	7	100	72	29	156	30.0

	DiabetesPedigreeFunction	Age	Outcome
1	0.351	31	0
3	0.167	21	0
5	0.201	30	0
7	0.134	29	0
10	0.191	30	0
12	1.441	57	0
18	0.183	33	0
20	0.704	27	0
21	0.388	50	0
27	0.487	22	0
0	0.627	50	1
2	0.672	32	1
4	2.288	33	1
6	0.248	26	1
8	0.158	53	1
9	0.232	54	1
11	0.537	34	1
13	0.398	59	1
14	0.587	51	1
15	0.484	32	1



7. Fit a linear model that predicts the class attribute using the 8 z-scored, continuous attributes of the cleaned dataset as independent variables . Report the R2 of the linear model and the coefficients of each attribute in the obtained regression function. Next, drop the two attributes, whose coefficients are the closest to 0, and obtain a linear model using the six remaining attributes as independent variable. Do the coefficients tell you anything about the importance of the attribute in predicting diabetes? What about negative and positive coefficients? Also compare the two regression functions!

In [16]:
```
df = diabetes_clean
cols = list(df.columns)
cols.remove('Outcome')
for col in cols:
    col_zscore = col + '_zscore'
    df[col_zscore] = (df[col] - df[col].mean())/df[col].std(ddof=0)
df

import statsmodels.formula.api as sm
result = sm.ols(formula="Outcome ~ Pregnancies_zscore + Glucose_zscore
+ BloodPressure_zscore + SkinThickness_zscore + Insulin_zscore + BMI_zs
core + DiabetesPedigreeFunction_zscore + Age_zscore", data=df).fit()
print(result.params)
print(result.summary())
```

```
Intercept                            0.348958
Pregnancies_zscore                   0.069717
Glucose_zscore                       0.196633
BloodPressure_zscore                -0.014003
SkinThickness_zscore                 0.001426
Insulin_zscore                      -0.007557
BMI_zscore                           0.099028
DiabetesPedigreeFunction_zscore      0.042785
Age_zscore                           0.024679
dtype: float64
                            OLS Regression Results
==============================================================================
=========
Dep. Variable:                 Outcome   R-squared:
0.321
Model:                             OLS   Adj. R-squared:
0.314
Method:                  Least Squares   F-statistic:
44.85
Date:                 Wed, 18 Sep 2019   Prob (F-statistic):
4.86e-59
Time:                         23:52:20   Log-Likelihood:
-372.00
No. Observations:                  768   AIC:
762.0
Df Residuals:                      759   BIC:
803.8
Df Model:                            8
Covariance Type:            nonrobust
==============================================================================
==============================
                                   coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------
Intercept                          0.3490      0.014     24.478
0.000       0.321       0.377
Pregnancies_zscore                 0.0697      0.017      4.089
0.000       0.036       0.103
Glucose_zscore                     0.1966      0.017     11.858
0.000       0.164       0.229
BloodPressure_zscore              -0.0140      0.016     -0.882
0.378      -0.045       0.017
SkinThickness_zscore               0.0014      0.017      0.083
0.934      -0.032       0.035
Insulin_zscore                    -0.0076      0.016     -0.478
0.633      -0.039       0.023
BMI_zscore                         0.0990      0.018      5.545
0.000       0.064       0.134
DiabetesPedigreeFunction_zscore    0.0428      0.015      2.935
0.003       0.014       0.071
Age_zscore                         0.0247      0.018      1.361
0.174      -0.011       0.060
==============================================================================
=========
Omnibus:                       32.824   Durbin-Watson:
```

Drop SkinThickness_zscore and Insulin_zscore as two closest to 0

In [17]:
```
result_after_drop = sm.ols(formula="Outcome ~ Pregnancies_zscore + Gluc
ose_zscore + BloodPressure_zscore + BMI_zscore + DiabetesPedigreeFuncti
on_zscore + Age_zscore", data=df).fit()
print(result_after_drop.params)
print(result_after_drop.summary())
```

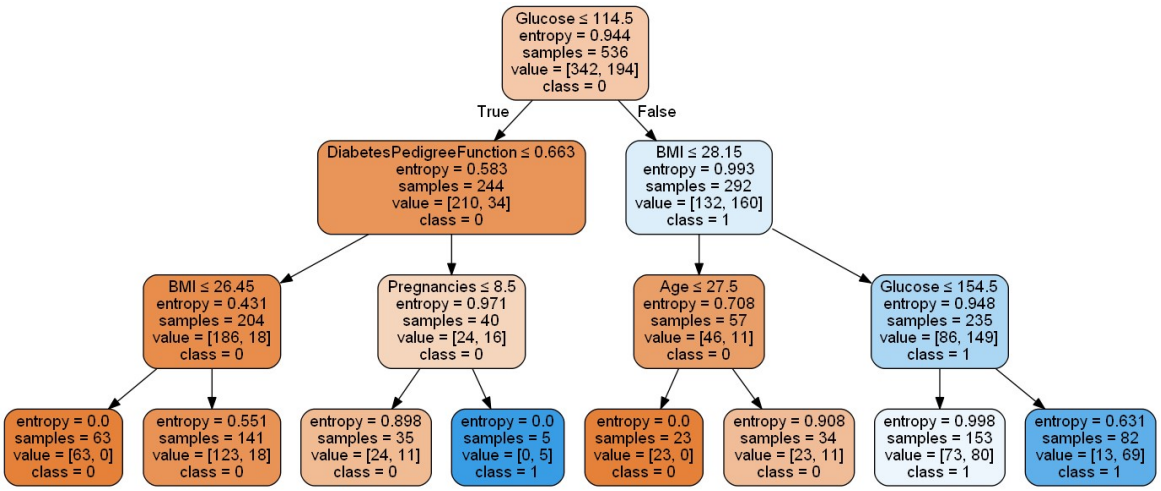
```
Intercept                           0.348958
Pregnancies_zscore                  0.069834
Glucose_zscore                      0.193703
BloodPressure_zscore               -0.013613
BMI_zscore                          0.099148
DiabetesPedigreeFunction_zscore     0.042585
Age_zscore                          0.024415
dtype: float64
                            OLS Regression Results
==============================================================================
=========
Dep. Variable:                 Outcome   R-squared:
0.321
Model:                             OLS   Adj. R-squared:
0.315
Method:                  Least Squares   F-statistic:
59.90
Date:                 Wed, 18 Sep 2019   Prob (F-statistic):
9.05e-61
Time:                         23:52:20   Log-Likelihood:
-372.12
No. Observations:                 768   AIC:
758.2
Df Residuals:                     761   BIC:
790.7
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
==============================
                                    coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------------
Intercept                           0.3490      0.014     24.506
0.000       0.321       0.377
Pregnancies_zscore                  0.0698      0.017      4.102
0.000       0.036       0.103
Glucose_zscore                      0.1937      0.015     12.630
0.000       0.164       0.224
BloodPressure_zscore               -0.0136      0.016     -0.860
0.390      -0.045       0.017
BMI_zscore                          0.0991      0.015      6.462
0.000       0.069       0.129
DiabetesPedigreeFunction_zscore     0.0426      0.015      2.926
0.004       0.014       0.071
Age_zscore                          0.0244      0.018      1.354
0.176      -0.011       0.060
==============================================================================
=========
Omnibus:                       31.401   Durbin-Watson:
1.963
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
22.706
Skew:                           0.315   Prob(JB):
1.17e-05
Kurtosis:                       2.441   Cond. No.
```

8. Create 3 decision tree models with 20 or less nodes using the cleaned dataset (total number of nodes should be less than 21 do not submit models with more than 20 nodes!); Explain how the 3 decision tree models were obtained. Report the training accuracy and the testing accuracy for each decision tree; interpret the learnt decision tree—what do they tell you about the importance of the 8 continuous attributes for the classification problem?

In [18]:
```
pima = pd.read_csv('diabetes_cleaned.csv',names = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome'])
#getting rid of the first row (header) : pregnancies, glucose, bloodpressure, skinthickness, ...
pima = pima.iloc[1:]
#split dataset in features and target variable
feature_cols = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
X = pima[feature_cols]
y = pima.Outcome
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)


##optimizing performance of decision tree
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

```
Accuracy: 0.7012987012987013
```

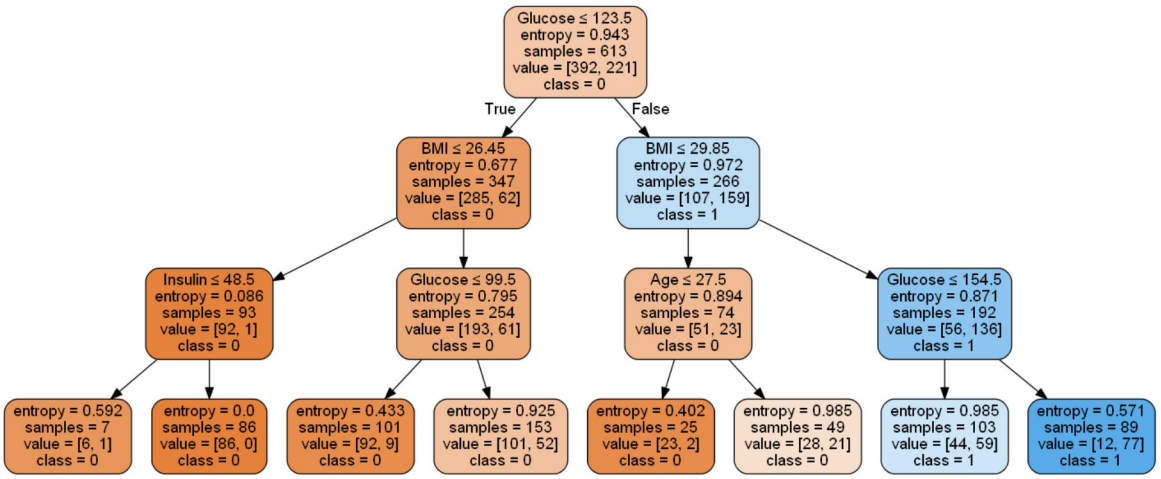
Out[18]:



In [23]:
```
pima = pd.read_csv('diabetes_cleaned.csv',names = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome'])
#getting rid of the first row (header) : pregnancies, glucose, bloodpressure, skinthickness, ...
pima = pima.iloc[1:]
#split dataset in features and target variable
feature_cols = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
X = pima[feature_cols]
y = pima.Outcome
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)


##optimizing performance of decision tree
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

```
Accuracy: 0.7337662337662337
```

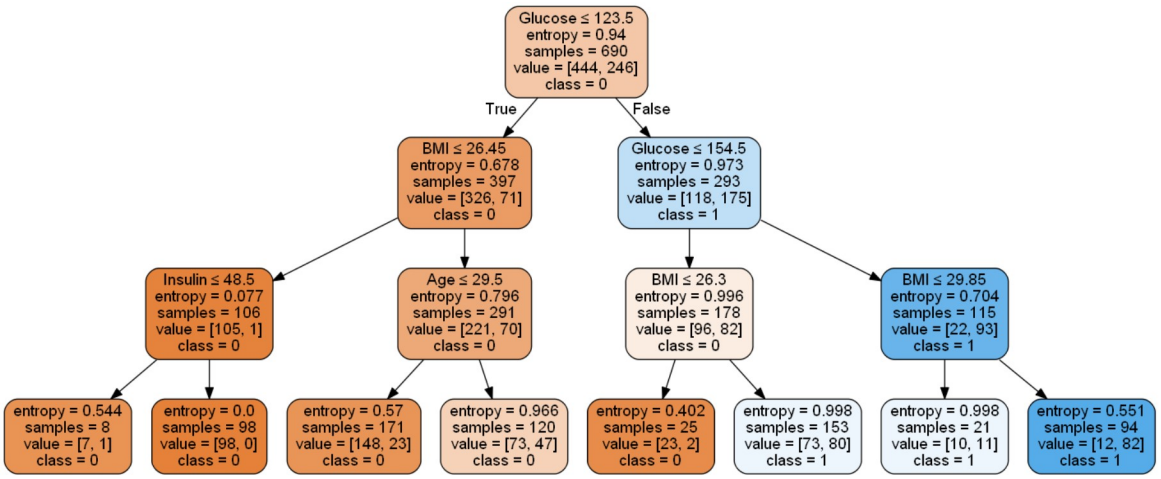
Out[23]:



In [22]:
```
pima = pd.read_csv('diabetes_cleaned.csv',names = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome'])
#getting rid of the first row (header) : pregnancies, glucose, bloodpressure, skinthickness, ...
pima = pima.iloc[1:]
#split dataset in features and target variable
feature_cols = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']
X = pima[feature_cols]
y = pima.Outcome
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)


##optimizing performance of decision tree
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names = feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```

```
Accuracy: 0.7012987012987013
```

Out[22]:



In [ ]:
```

```