In [1]:
```
import pandas as pd
```

In [13]:
```
data = pd.read_csv('health care diabetes.csv')
```

In [14]:
```
data.head()
```

Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

Week 1

In [16]:
```
data.isnull().any()
```

Out[16]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

In [17]:
```
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
```

In [18]:
```
positive = data[data['Outcome']==1]
positive.head()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
2	8	183	64	0	0	23.3	0.672	32
4	0	137	40	35	168	43.1	2.288	33
6	3	78	50	32	88	31.0	0.248	26
8	2	197	70	45	543	30.5	0.158	53

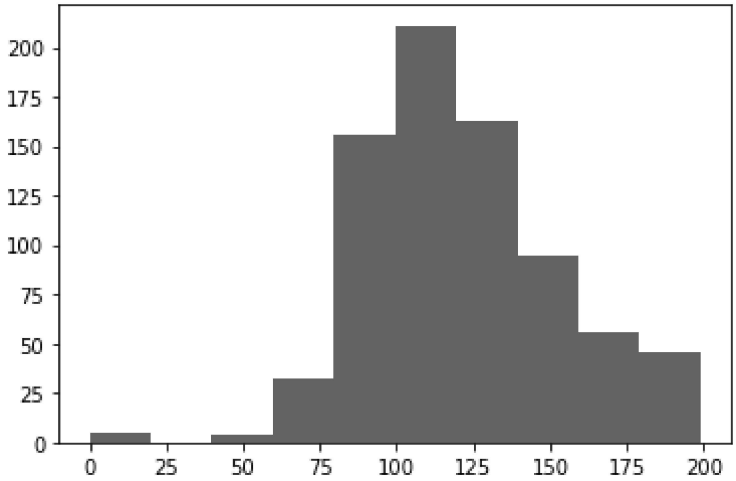
In [19]:
```
data['Glucose'].value_counts().head(10)
```

Out[19]:
```
99     17
100    17
111    14
129    14
125    14
106    14
112    13
108    13
95     13
105    13
Name: Glucose, dtype: int64
```

In [20]:
```
import matplotlib.pyplot as plt
```

In [21]:
```
plt.hist(data['Glucose'])
```

Out[21]:
```
(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 [22]:
```
data['BloodPressure'].value_counts().head(10)
```

Out[22]:
```
70    57
74    52
78    45
68    45
72    44
64    43
80    40
```

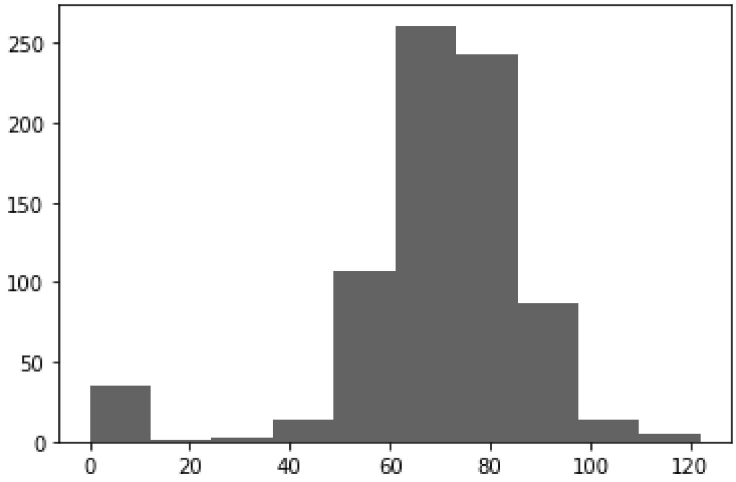
```
76    39
60    37
0     35
Name: BloodPressure, dtype: int64
```

In [23]:

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

Out[23]:

```
(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 [24]:

```
data['SkinThickness'].value_counts().head(10)
```

Out[24]:

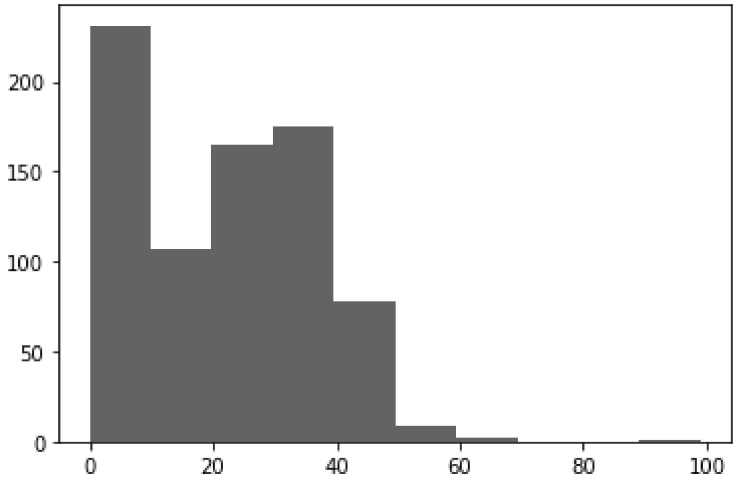
```
0     227
32     31
30     27
27     23
23     22
33     20
28     20
18     20
31     19
19     18
Name: SkinThickness, dtype: int64
```

In [25]:

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

Out[25]:

```
(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>)
```

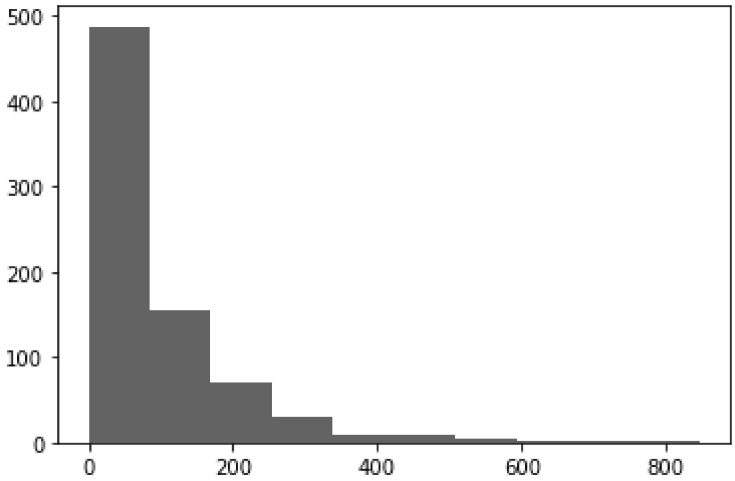


In [26]:
```
data['Insulin'].value_counts().head(10)
```

Out[26]:
```
0      374
105     11
130      9
140      9
120      8
94       7
180      7
100      7
135      6
115      6
Name: Insulin, dtype: int64
```

In [27]:
```
plt.hist(data['Insulin'])
```

Out[27]:
```
(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>)
```

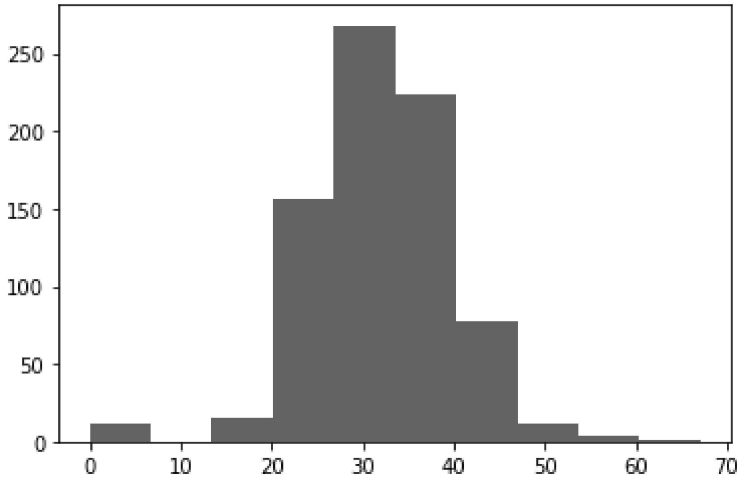


In [29]:
```
data['BMI'].value_counts().head(10)
```

Out[29]:
```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
33.3    10
30.1     9
32.8     9
32.9     9
30.8     9
Name: BMI, dtype: int64
```

In [30]:
```
plt.hist(data['BMI'])
```

Out[30]:
```
(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 [31]:
```
data.describe().transpose()
```

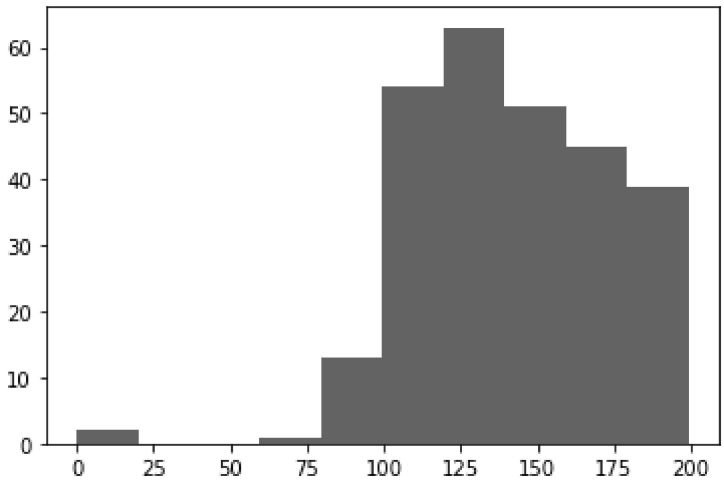
Out[31]:

	count	mean	std	min	25%	50%	75%	m
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1

Week 2

In [47]:
```
plt.hist(positive['Glucose'],histtype='stepfilled',bins=10)
```

Out[47]:
```
(array([ 2.,  0.,  0.,  1., 13., 54., 63., 51., 45., 39.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 [<matplotlib.patches.Polygon at 0x1b5473c2850>])
```

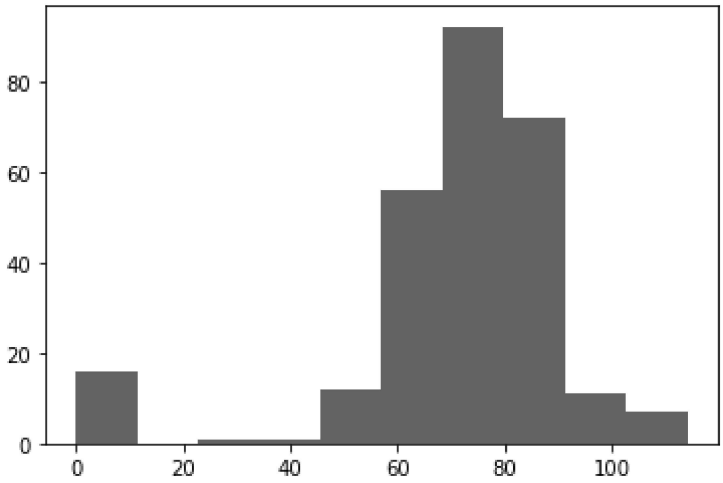


In [39]:
```
positive['Glucose'].value_counts().head(10)
```

Out[39]:
```
125    7
128    6
129    6
115    6
158    6
146    5
124    5
162    5
173    5
109    5
Name: Glucose, dtype: int64
```

In [33]:
```
plt.hist(positive['BloodPressure'],histtype='stepfilled',bins=10)
```

Out[33]:
```
(array([16.,  0.,  1.,  1., 12., 56., 92., 72., 11.,  7.]),
 array([  0. ,  11.4,  22.8,  34.2,  45.6,  57. ,  68.4,  79.8,  91.2,
        102.6, 114. ]),
 [<matplotlib.patches.Polygon at 0x1b5470ee820>])
```

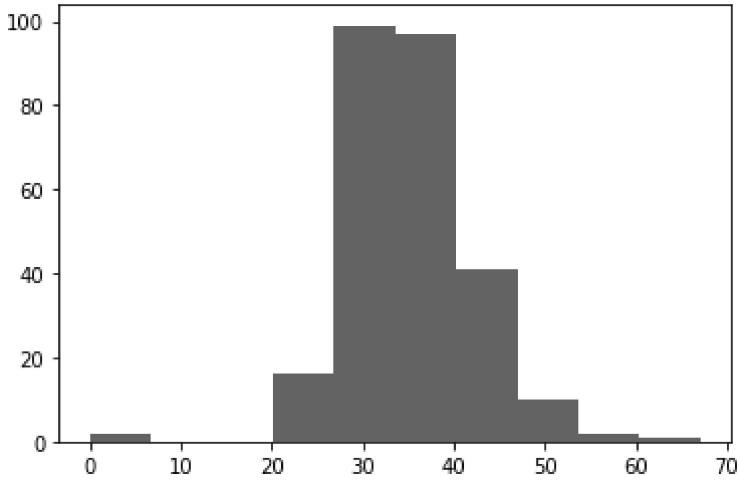


In [40]:
```
positive['BloodPressure'].value_counts().head(10)
```

Out[40]:
```
70    23
76    18
78    17
74    17
72    16
0     16
80    13
64    13
82    13
84    12
Name: BloodPressure, dtype: int64
```

In [34]:
```
plt.hist(positive['BMI'],histtype='stepfilled',bins=10)
```

Out[34]:
```
(array([ 2.,  0.,  0., 16., 99., 97., 41., 10.,  2.,  1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 [<matplotlib.patches.Polygon at 0x1b547144e80>])
```

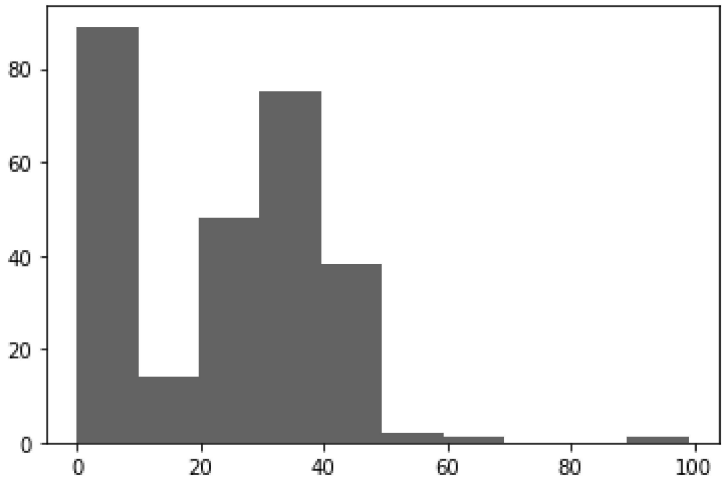


In [41]:
```
positive['BMI'].value_counts().head(10)
```

Out[41]:
```
32.9    8
31.6    7
33.3    6
31.2    5
30.5    5
32.0    5
34.3    4
30.4    4
32.4    4
43.3    4
Name: BMI, dtype: int64
```

In [43]:
```
plt.hist(positive['SkinThickness'],histtype='stepfilled',bins=10)
```

Out[43]:
```
(array([89., 14., 48., 75., 38.,  2.,  1.,  0.,  0.,  1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 [<matplotlib.patches.Polygon at 0x1b5472fca30>])
```



In [44]:
```
positive['SkinThickness'].value_counts().head(10)
```

Out[44]:
```
0     88
32    14
30     9
33     9
39     8
37     8
36     8
35     8
```

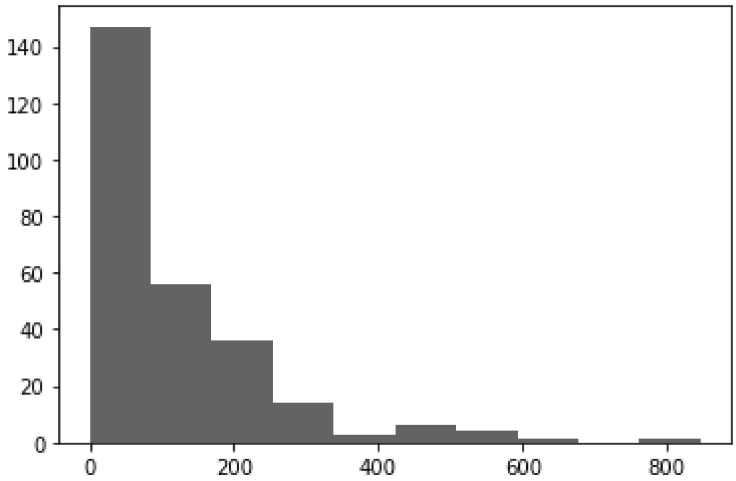
```
27     7
29     7
Name: SkinThickness, dtype: int64
```

In [45]:

```
plt.hist(positive['Insulin'],histtype='stepfilled',bins=10)
```

Out[45]:

```
(array([147.,  56.,  36.,  14.,   3.,   6.,   4.,   1.,   0.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 [<matplotlib.patches.Polygon at 0x1b5473633d0>])
```



In [46]:

```
positive['Insulin'].value_counts().head(10)
```

Out[46]:

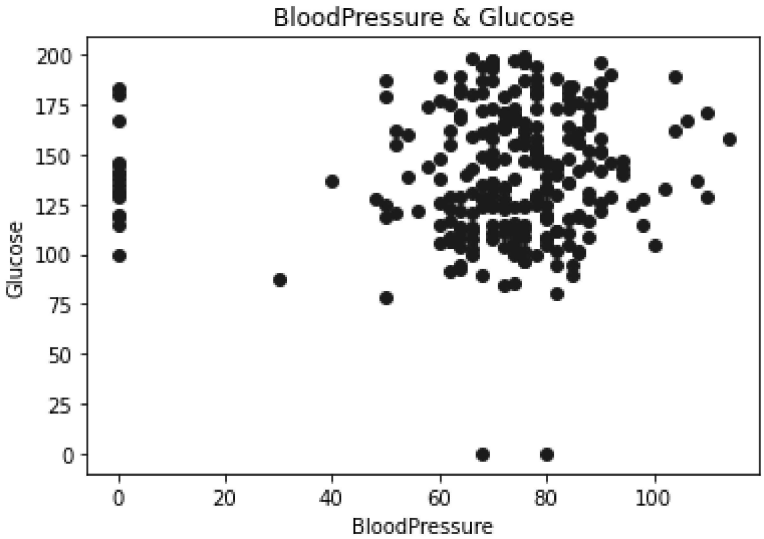
```
0      138
130      6
180      4
175      3
156      3
185      2
194      2
168      2
144      2
210      2
Name: Insulin, dtype: int64
```

In [48]:

```
BloodPressure = positive['BloodPressure']
Glucose = positive['Glucose']
Insulin = positive['Insulin']
BMI = positive['BMI']
Thickness = positive['SkinThickness']
```

In [52]:

```
plt.scatter(BloodPressure, Glucose, color=['b'])
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
plt.title('BloodPressure & Glucose')
plt.show()
```

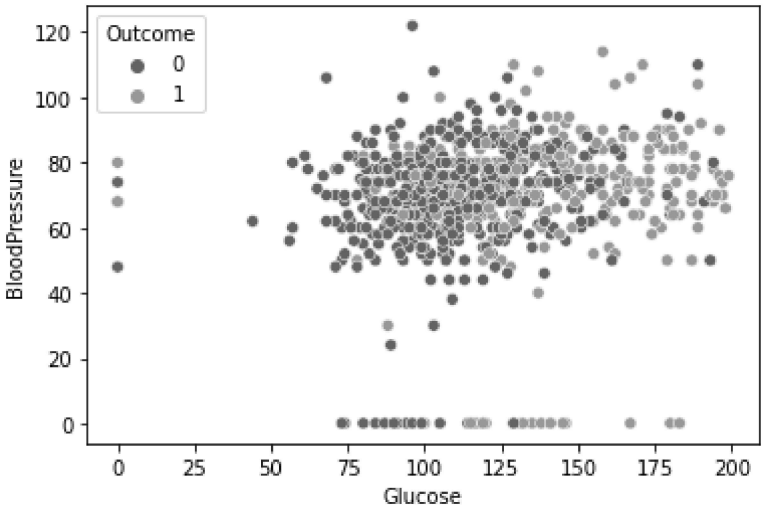


In [53]:

```
import seaborn as sns
```

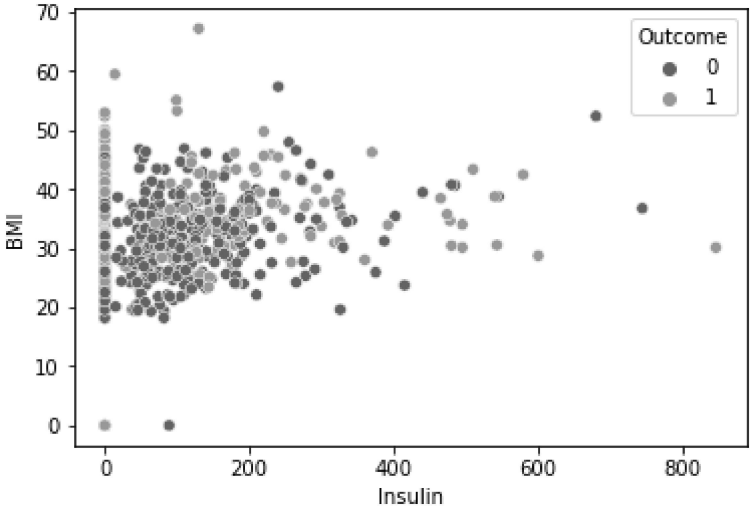
In [54]:

```
g =sns.scatterplot(x= "Glucose" ,y= "BloodPressure",
                   hue="Outcome",
                   data=data);
```



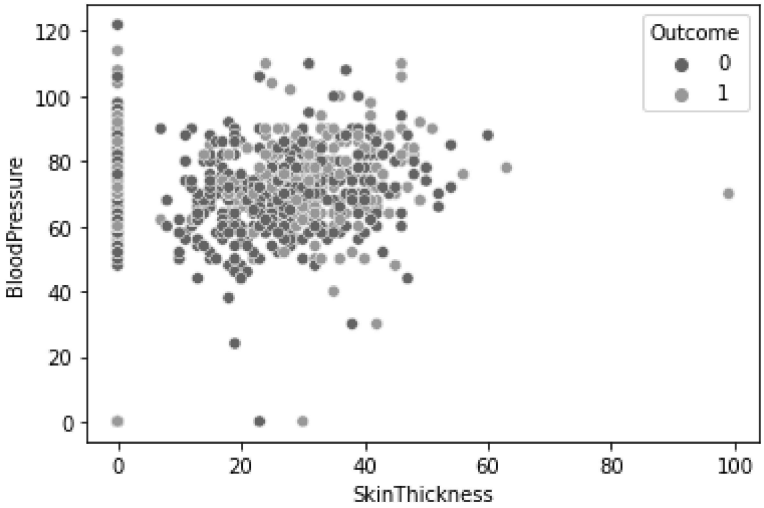
In [55]:

```
g =sns.scatterplot(x= "Insulin" ,y= "BMI",
                   hue="Outcome",
                   data=data);
```



In [56]:

```
g =sns.scatterplot(x= "SkinThickness" ,y= "BloodPressure",
                   hue="Outcome",
                   data=data);
```



In [57]:

```
data.corr()
```

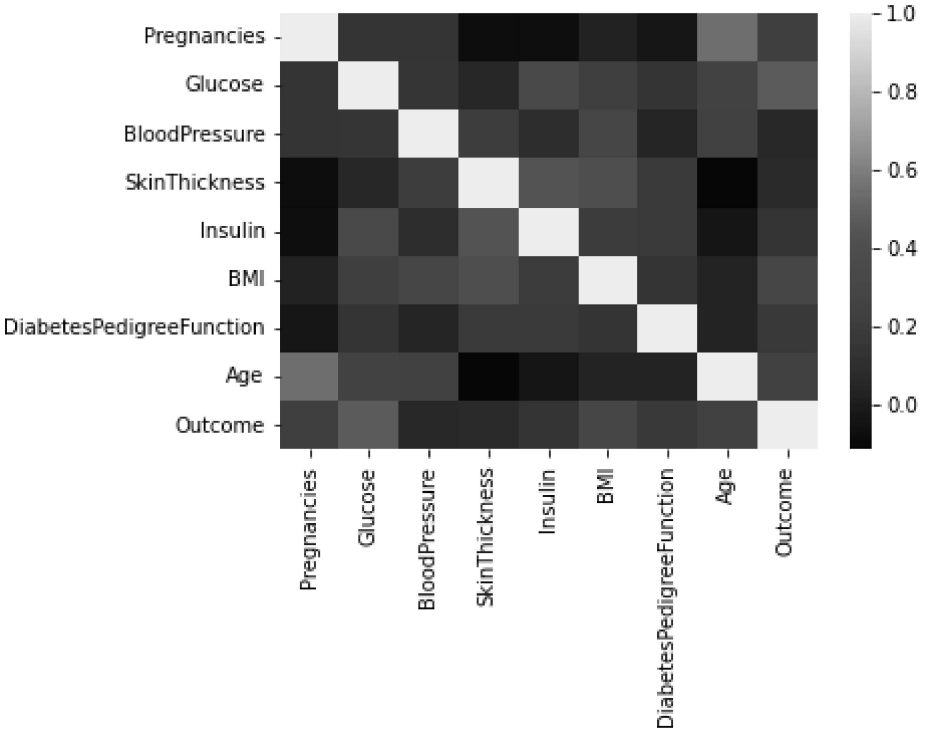
Out[57]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
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 [58]:

```
sns.heatmap(data.corr())
```

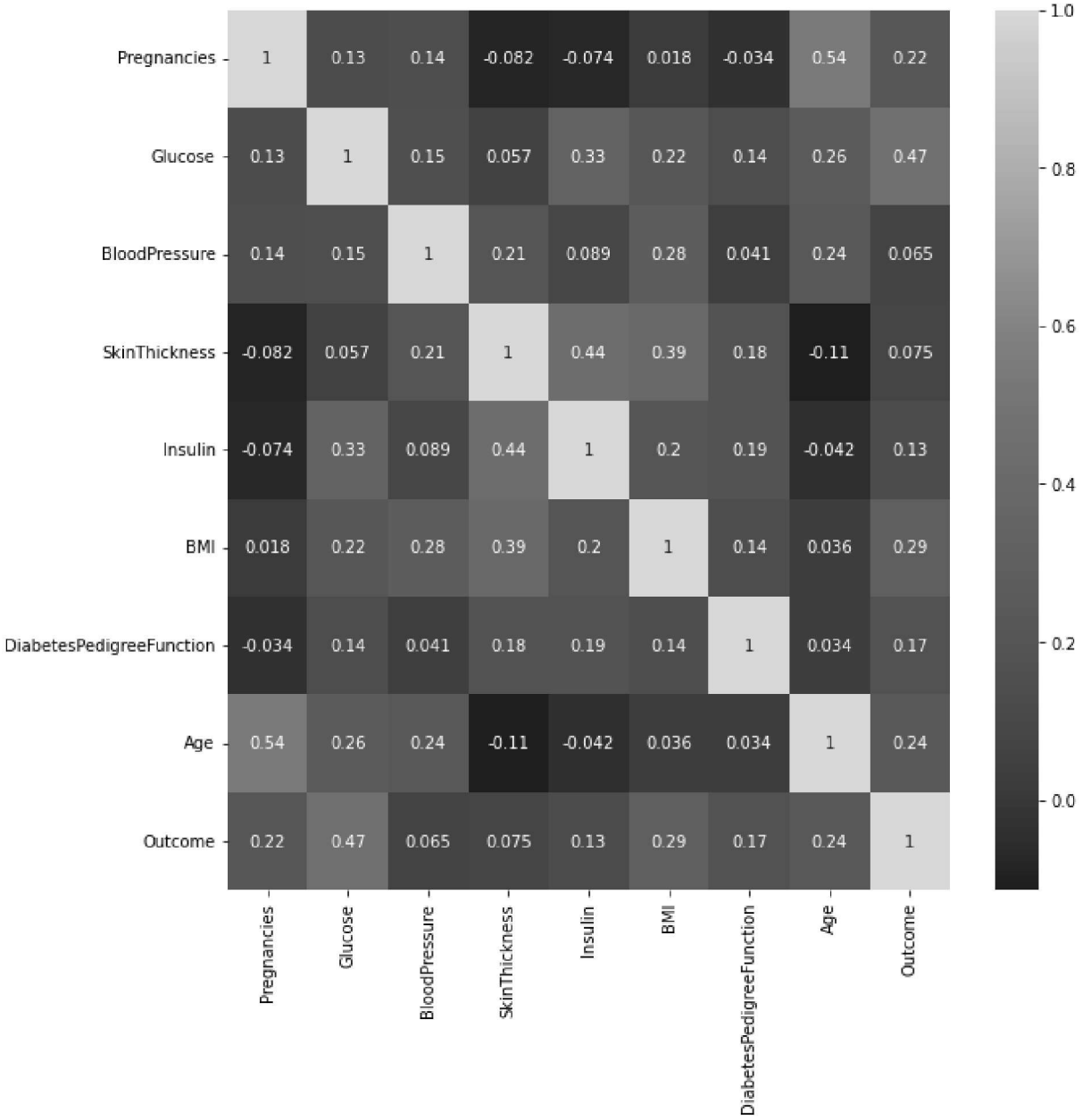
Out[58]: <AxesSubplot:>



In [62]:

```
plt.subplots(figsize=(10,10))
sns.heatmap(data.corr(),annot=True,cmap='viridis')
```

Out[62]: <AxesSubplot:>

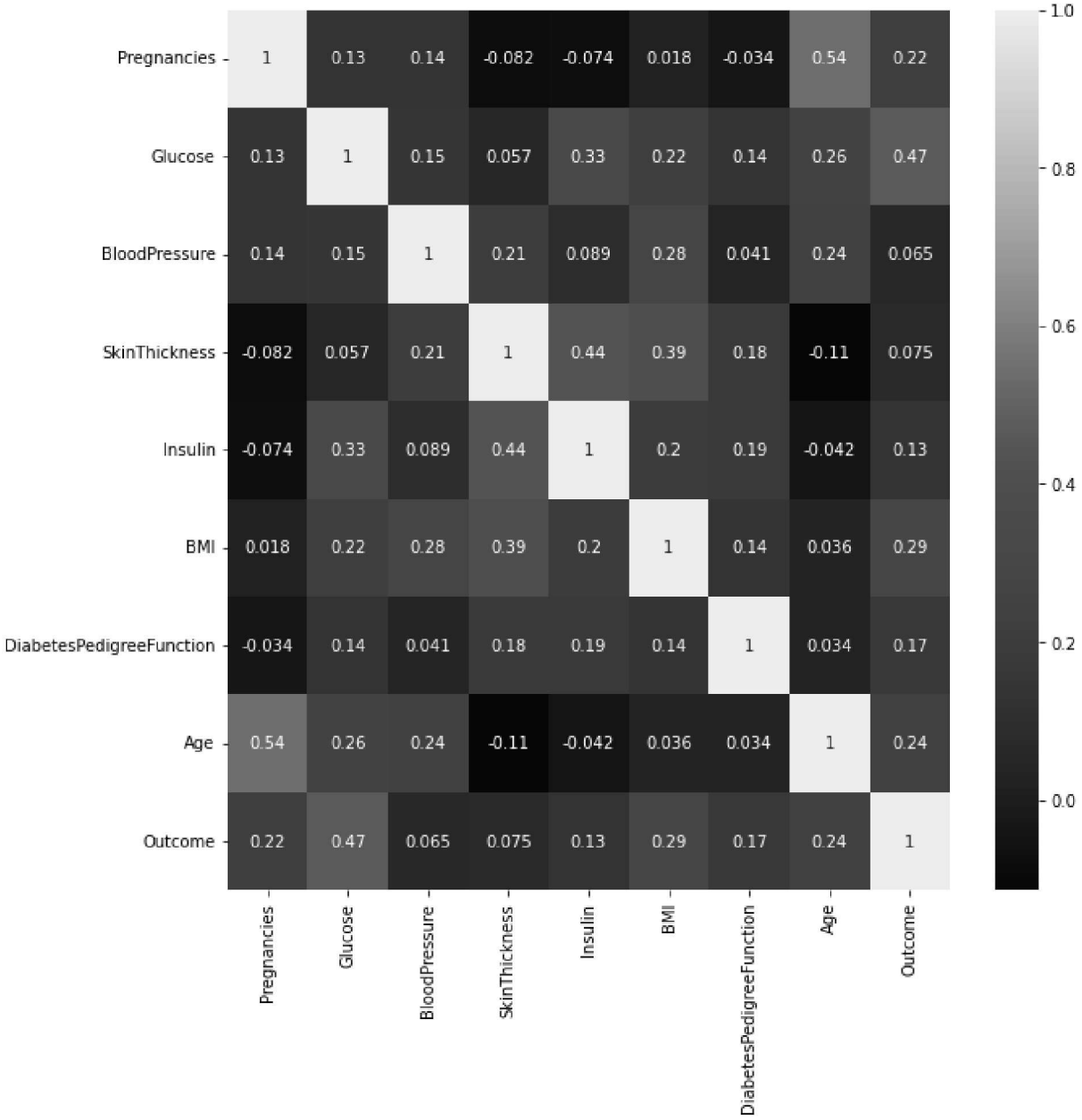


In [173…

```
plt.subplots(figsize=(10,10))
sns.heatmap(data.corr(),annot=True)
```

Out[173…

```
<AxesSubplot:>
```



Weak 3

In [63]:
```
data.head()
```

Out[63]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [80]:
```
features = data.iloc[:,[0,1,2,3,4,6,7]].values
label = data.iloc[:,8].values
```

In [84]:
```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test = train_test_split(features,label,test_size=0.2,random
```

In [85]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[85]:

```
LogisticRegression()
```

In [89]:

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

```
0.7736156351791531
0.6883116883116883
```

In [94]:

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

Out[94]:

```
array([[444,  56],
       [131, 137]], dtype=int64)
```

In [95]:

```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.77      0.89      0.83       500
           1       0.71      0.51      0.59       268

    accuracy                           0.76       768
   macro avg       0.74      0.70      0.71       768
weighted avg       0.75      0.76      0.75       768
```

Weak 4

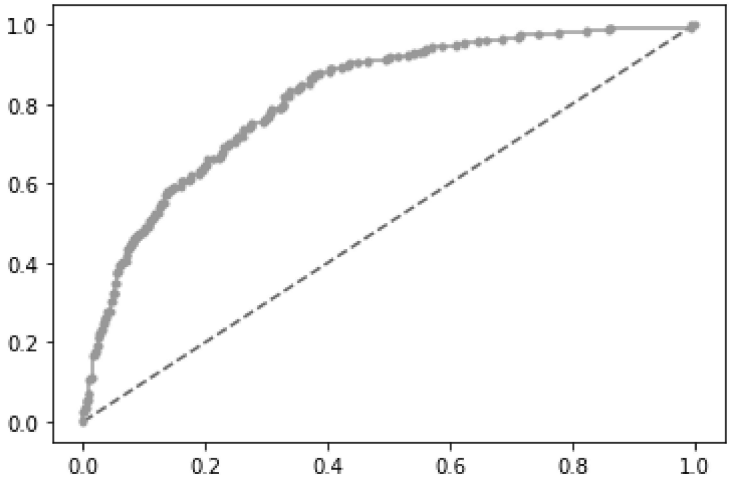
In [97]:

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

probs = model.predict_proba(features)

probs = probs[:, 1]

auc = roc_auc_score(label, probs)
print('AUC: %.3f' %auc)

fpr, tpr, threshold = roc_curve(label, probs)

plt.plot([0, 1], [0, 1], linestyle='--')

plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.818
```

Out[97]:

```
[<matplotlib.lines.Line2D at 0x1b54a2b52b0>]
```



In [100…

```
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

Out[100…

```
DecisionTreeClassifier(max_depth=5)
```

In [102…

```
model3.score(X_train,y_train)
```

Out[102…

```
0.8208469055374593
```

In [108…

```
model3.score(X_test,y_test)
```

Out[108…

```
0.7337662337662337
```

In [106…

```
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(X_train,y_train)
```

Out[106…

```
RandomForestClassifier(n_estimators=11)
```

In [107…

```
model4.score(X_train,y_train)
```

Out[107…

```
0.990228013029316
```

In [109…

```
model4.score(X_test,y_test)
```

Out[109…

```
0.7142857142857143
```

In [144…

```
from sklearn.svm import SVC
model5 = SVC(kernel='rbf',gamma='auto')
model5.fit(X_train,y_train)
```

Out[144…

```
SVC(gamma='auto')
```

In [119…

```
model5.score(X_test,y_test,)
```

```
0.6168831168831169
```

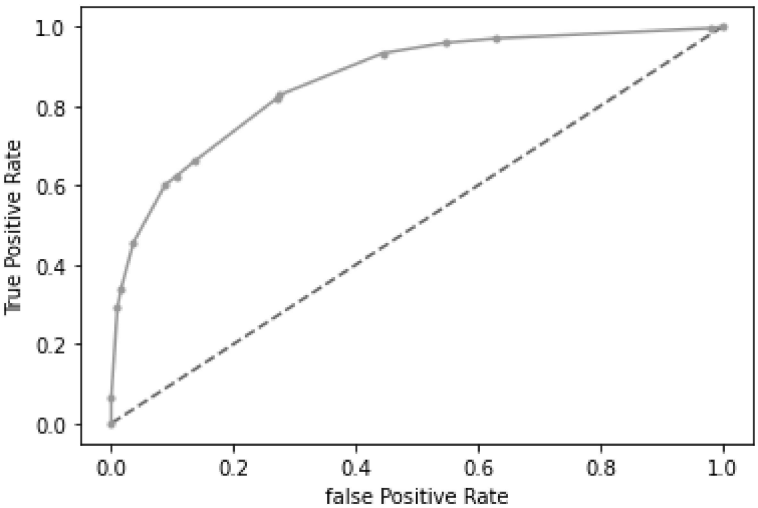
Out[119…

In [163…

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

probs = model3.predict_proba(features)

probs = probs[:, 1]

auc = roc_auc_score(label, probs)
print('AUC: %.3f' %auc)

fpr, tpr, thresholds = roc_curve(label, probs)
print("true Positive Rate - {}, False Positive Rate - {} Threshold - {}".format(tpr,

plt.plot([0, 1], [0, 1], linestyle='--')

plt.plot(fpr, tpr, marker='.')
plt.xlabel("false Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.863
true Positive Rate - [0.         0.06343284 0.29104478 0.33955224 0.45522388 0.60074
627
 0.62313433 0.66044776 0.82089552 0.82835821 0.93283582 0.95895522
 0.97014925 0.99626866 1.        ], False Positive Rate - [0.    0.    0.01  0.016
0.036 0.088 0.106 0.136 0.272 0.276 0.444 0.546
 0.628 0.982 1.   ] Threshold - [2.         1.         0.9245283  0.92307692 0.78378
378 0.58695652
 0.46153846 0.45       0.36363636 0.33333333 0.22222222 0.09090909
 0.07894737 0.02040816 0.        ]
Text(0, 0.5, 'True Positive Rate')
```

Out[163…



In [164…

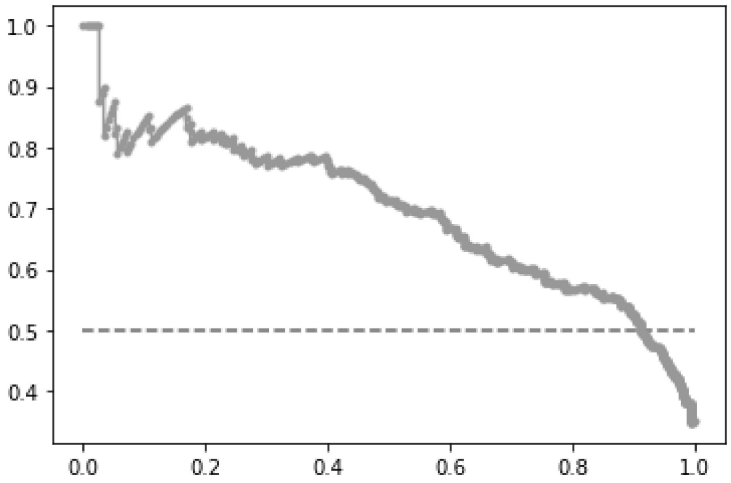
```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
probs = model.predict_proba(features)
probs = probs[:, 1]

yhat = model.predict(features)

precision, recall, thresholds = precision_recall_curve(label, probs)

f1 = f1_score(label, yhat)
```

```
auc = auc(recall, precision)

ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' %(f1, auc, ap))

print('f1=%.3f auc=%.3f ap=%.3f' %(f1, auc, ap))
plt.plot([0, 1], [0.5, 0.5], linestyle='--')

plt.plot(recall, precision, marker='.')
```

```
f1=0.594 auc=0.696 ap=0.697
```

Out[164…

```
[<matplotlib.lines.Line2D at 0x1b54a823be0>]
```



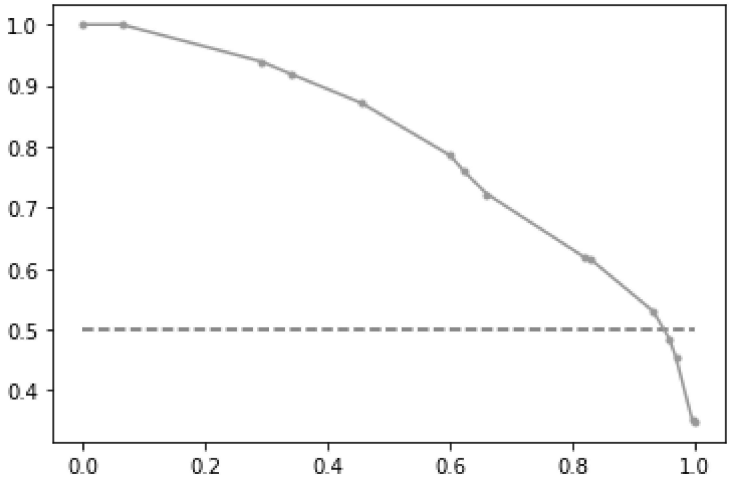
In [166…

```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
probs = model3.predict_proba(features)
probs = probs[:, 1]

yhat = model.predict(features)

precision, recall, thresholds = precision_recall_curve(label, probs)

f1 = f1_score(label, yhat)

auc = auc(recall, precision)

print('f1=%.3f auc=%.3f ap=%.3f' %(f1, auc, ap))

plt.plot([0, 1], [0.5, 0.5], linestyle='--')

plt.plot(recall, precision, marker='.')
```

```
f1=0.594 auc=0.801 ap=0.697
```

Out[166…

```
[<matplotlib.lines.Line2D at 0x1b54a739c70>]
```



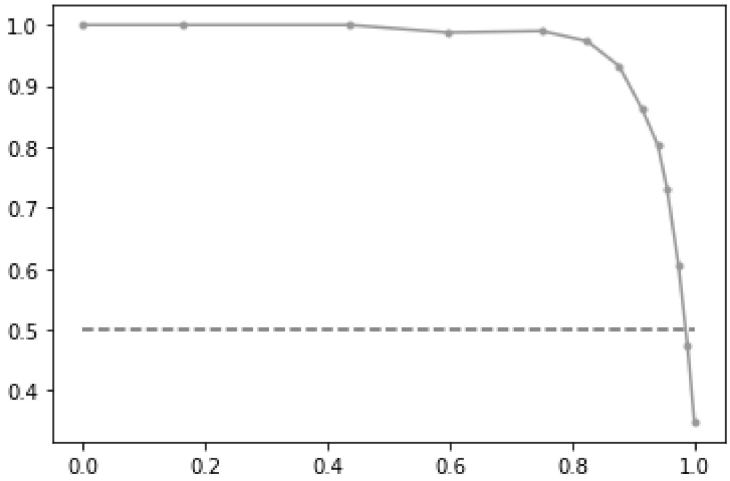
In [167…

```
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
probs = model4.predict_proba(features)
probs = probs[:, 1]

yhat = model.predict(features)

precision, recall, thresholds = precision_recall_curve(label, probs)

f1 = f1_score(label, yhat)

auc = auc(recall, precision)

print('f1=%.3f auc=%.3f ap=%.3f' %(f1, auc, ap))

plt.plot([0, 1], [0.5, 0.5], linestyle='--')

plt.plot(recall, precision, marker='.')
```

```
f1=0.594 auc=0.962 ap=0.697
```

Out[167…

```
[<matplotlib.lines.Line2D at 0x1b54a41edc0>]
```



In [ ]: