In [56]:
```
import pandas as pd
```

In [3]:
```
import matplotlib.pyplot as plt
```

In [4]:
```
import seaborn as sns
```

In [6]:
```
data=pd.read_csv("D:/Riya/SIMPLILEARN/DS Capstone/My project Healthcare/Project_2/health care diabetes.csv")
data.head()
```

Out[6]:

	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

Week1

In [7]:
```
data.isnull().any()
```

Out[7]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

In [8]:
```
data.describe().transpose()
```

Out[8]:

	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 [9]:
```
positive=data[data['Outcome']==1]
positive.head()
```

Out[9]:

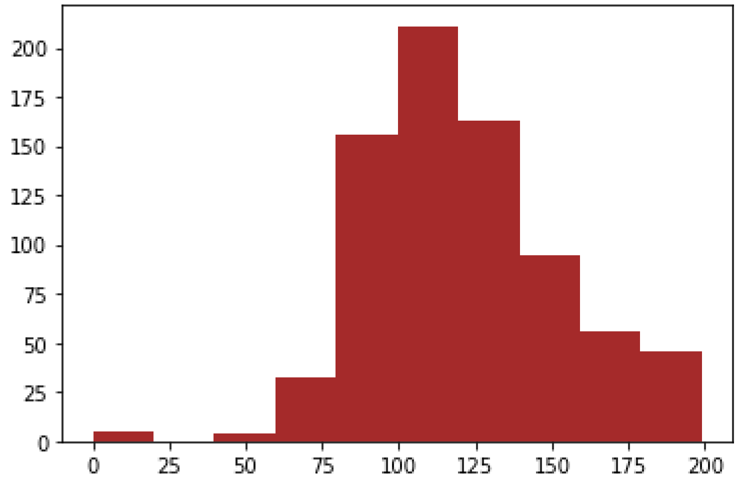
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
2	8	183	64	0	0	23.3	0.672	32	1
4	0	137	40	35	168	43.1	2.288	33	1
6	3	78	50	32	88	31.0	0.248	26	1
8	2	197	70	45	543	30.5	0.158	53	1

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

Out[10]:
```
100    17
99     17
129    14
125    14
111    14
106    14
95     13
108    13
105    13
102    13
Name: Glucose, dtype: int64
```

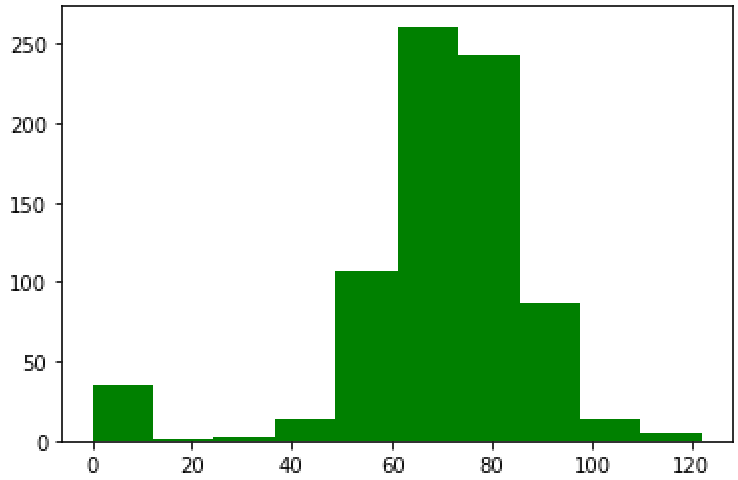
In [11]:
```
plt.hist(data['Glucose'],color='Brown')
```

Out[11]:
```
(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>)
```



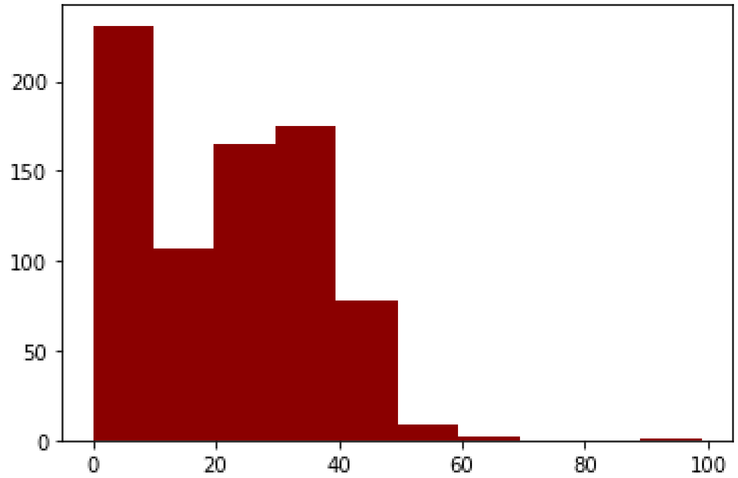
In [12]:
```
plt.hist(data['BloodPressure'],color='Green')
```

Out[12]:
```
(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>)
```



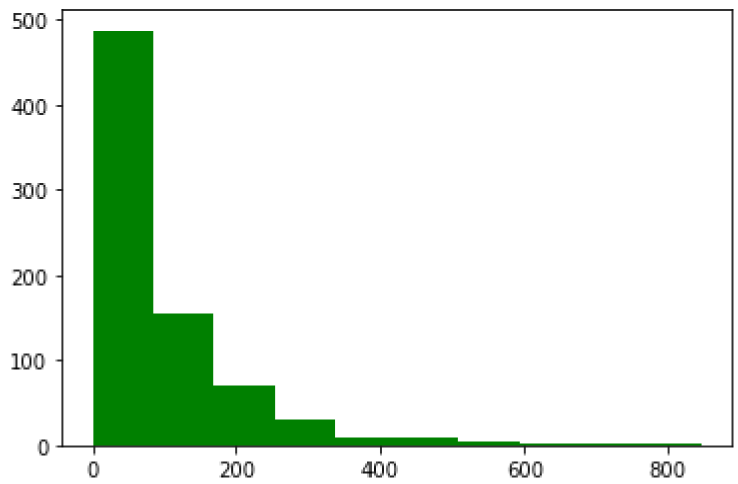
In [13]:
```
plt.hist(data['SkinThickness'],color='DarkRed')
```

Out[13]:
```
(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>)
```



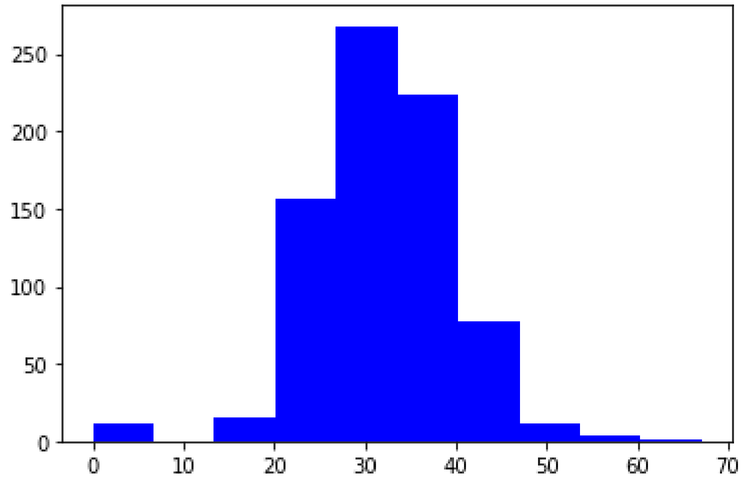
In [14]:
```
plt.hist(data['Insulin'],color='Green')
```

Out[14]:
```
(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 [15]:
```
plt.hist(data['BMI'],color='Blue')
```

Out[15]:
```
(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 [16]:
```
print(data['Glucose'].value_counts().head())
print(data['BloodPressure'].value_counts().head())
print(data['SkinThickness'].value_counts().head())
print(data['Insulin'].value_counts().head())
data['BMI'].value_counts().head()
```

```
100    17
99     17
129    14
125    14
111    14
Name: Glucose, dtype: int64
70    57
74    52
68    45
78    45
72    44
Name: BloodPressure, dtype: int64
0     227
32     31
30     27
27     23
23     22
Name: SkinThickness, dtype: int64
0      374
105     11
140      9
130      9
120      8
Name: Insulin, dtype: int64
```

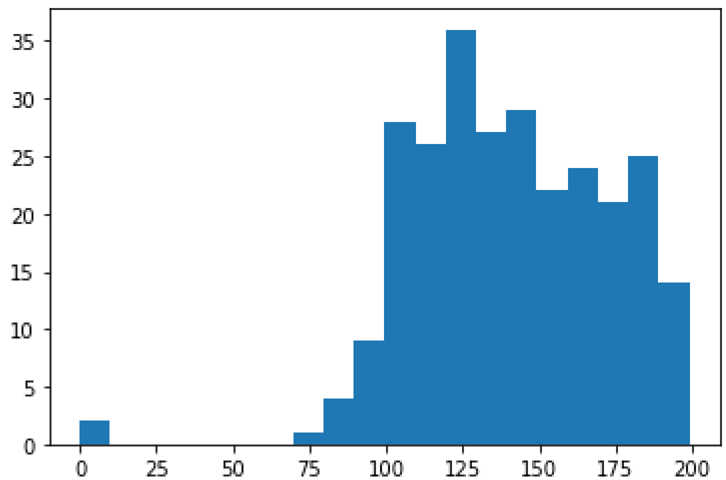
Out[16]:
```
32.0    13
31.6    12
31.2    12
0.0     11
33.3    10
Name: BMI, dtype: int64
```

Week2

In [17]:
```
print(positive['Glucose'].value_counts().head())
plt.hist(positive['Glucose'],histtype='stepfilled',bins=20)
```

```
125    7
158    6
128    6
115    6
129    6
Name: Glucose, dtype: int64
```

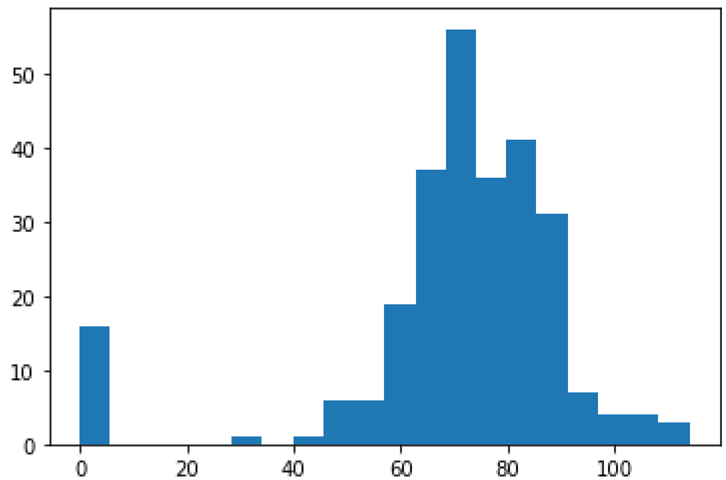
Out[17]:
```
(array([ 2.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  4.,  9., 28., 26., 36.,
        27., 29., 22., 24., 21., 25., 14.]),
 array([  0.  ,   9.95,  19.9 ,  29.85,  39.8 ,  49.75,  59.7 ,  69.65,
         79.6 ,  89.55,  99.5 , 109.45, 119.4 , 129.35, 139.3 , 149.25,
        159.2 , 169.15, 179.1 , 189.05, 199.  ]),
 [<matplotlib.patches.Polygon at 0x24ecb5ff070>])
```



In [18]:
```
print(positive['BloodPressure'].value_counts().head())
plt.hist(positive['BloodPressure'],histtype='stepfilled',bins=20)
```

```
70    23
76    18
78    17
74    17
72    16
Name: BloodPressure, dtype: int64
```

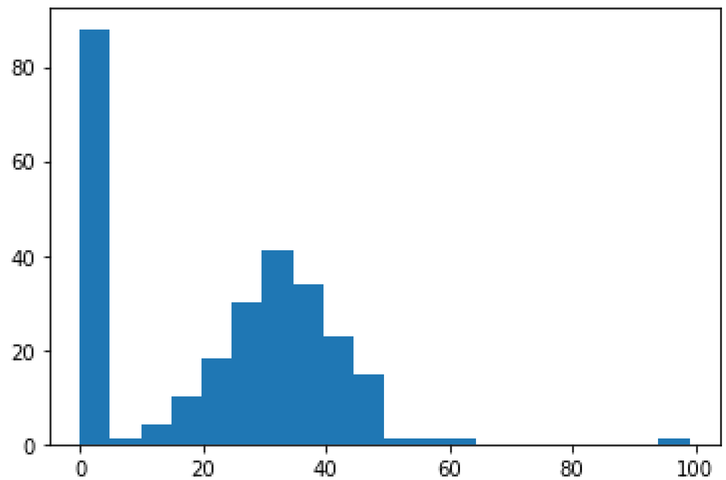
Out[18]:
```
(array([16.,  0.,  0.,  0.,  0.,  1.,  0.,  1.,  6.,  6., 19., 37., 56.,
        36., 41., 31.,  7.,  4.,  4.,  3.]),
 array([  0. ,   5.7,  11.4,  17.1,  22.8,  28.5,  34.2,  39.9,  45.6,
         51.3,  57. ,  62.7,  68.4,  74.1,  79.8,  85.5,  91.2,  96.9,
        102.6, 108.3, 114. ]),
 [<matplotlib.patches.Polygon at 0x24ecb657d30>])
```



In [19]:
```
print(positive['SkinThickness'].value_counts().head())
plt.hist(positive['SkinThickness'],histtype='stepfilled',bins=20)
```

```
0     88
32    14
33     9
30     9
39     8
Name: SkinThickness, dtype: int64
```

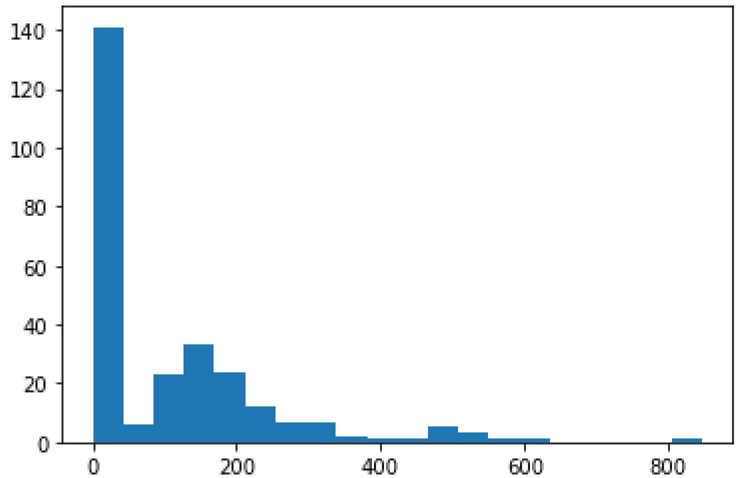
Out[19]:
```
(array([88.,  1.,  4., 10., 18., 30., 41., 34., 23., 15.,  1.,  1.,  1.,
         0.,  0.,  0.,  0.,  0.,  0.,  1.]),
 array([ 0.  ,  4.95,  9.9 , 14.85, 19.8 , 24.75, 29.7 , 34.65, 39.6 ,
        44.55, 49.5 , 54.45, 59.4 , 64.35, 69.3 , 74.25, 79.2 , 84.15,
        89.1 , 94.05, 99.  ]),
 [<matplotlib.patches.Polygon at 0x24ecb6a9f40>])
```



In [20]:
```
print(positive['Insulin'].value_counts().head())
plt.hist(positive['Insulin'],histtype='stepfilled',bins=20)
```

```
0      138
130      6
180      4
156      3
175      3
Name: Insulin, dtype: int64
```

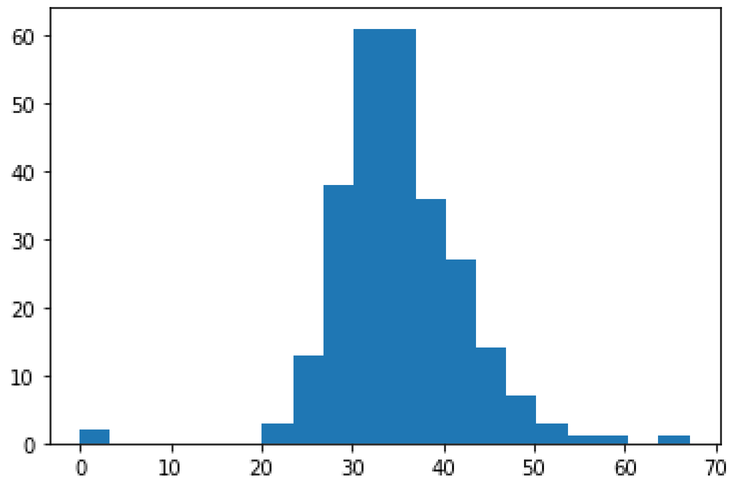
Out[20]:
```
(array([141.,   6.,  23.,  33.,  24.,  12.,   7.,   7.,   2.,   1.,   1.,
          5.,   3.,   1.,   1.,   0.,   0.,   0.,   0.,   1.]),
 array([  0. ,  42.3,  84.6, 126.9, 169.2, 211.5, 253.8, 296.1, 338.4,
        380.7, 423. , 465.3, 507.6, 549.9, 592.2, 634.5, 676.8, 719.1,
        761.4, 803.7, 846. ]),
 [<matplotlib.patches.Polygon at 0x24ecb6feeb0>])
```



In [21]:
```
print(positive['BMI'].value_counts().head())
plt.hist(positive['BMI'],histtype='stepfilled',bins=20)
```

```
32.9    8
31.6    7
33.3    6
30.5    5
32.0    5
Name: BMI, dtype: int64
```

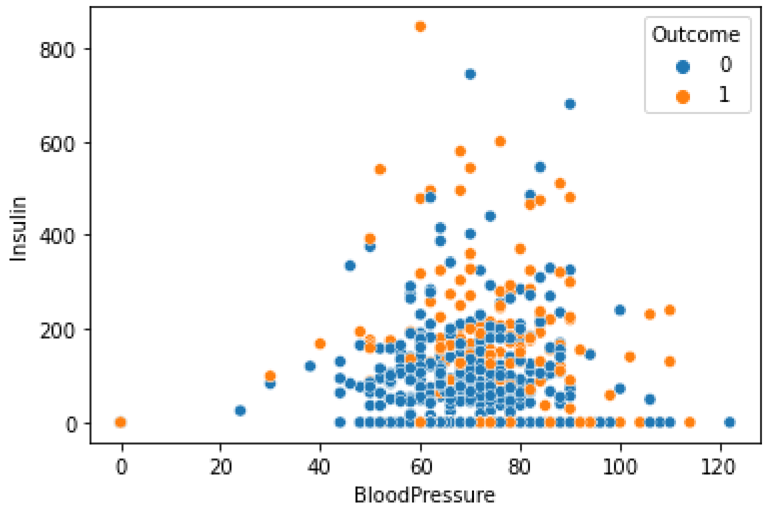
Out[21]:
```
(array([ 2.,  0.,  0.,  0.,  0.,  0.,  3., 13., 38., 61., 61., 36., 27.,
        14.,  7.,  3.,  1.,  1.,  0.,  1.]),
 array([ 0.   ,  3.355,  6.71 , 10.065, 13.42 , 16.775, 20.13 , 23.485,
        26.84 , 30.195, 33.55 , 36.905, 40.26 , 43.615, 46.97 , 50.325,
        53.68 , 57.035, 60.39 , 63.745, 67.1  ]),
 [<matplotlib.patches.Polygon at 0x24ecb75c400>])
```



In [22]:
```
BloodPressure = positive['BloodPressure']
Glucose = positive['Glucose']
SkinThickness = positive['SkinThickness']
Insulin = positive['Insulin']
BMI = positive['BMI']
```

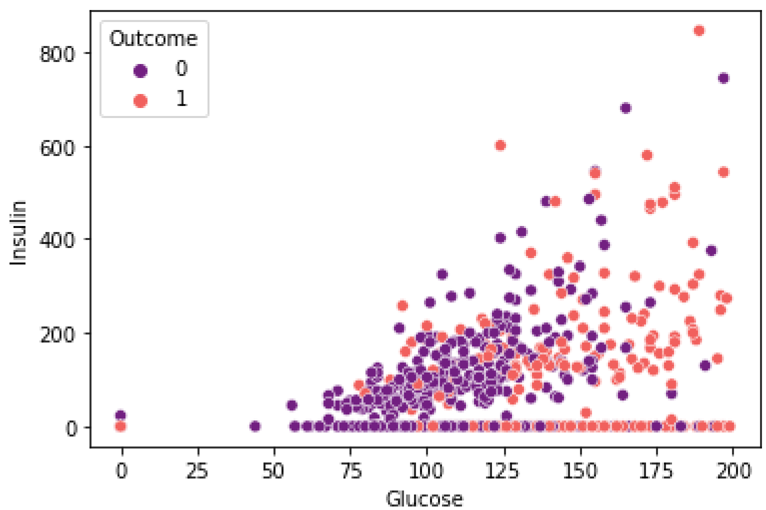
In [23]:
```
sns.scatterplot(x='BloodPressure',y='Insulin',hue='Outcome',data=data)
```

Out[23]: <AxesSubplot:xlabel='BloodPressure', ylabel='Insulin'>



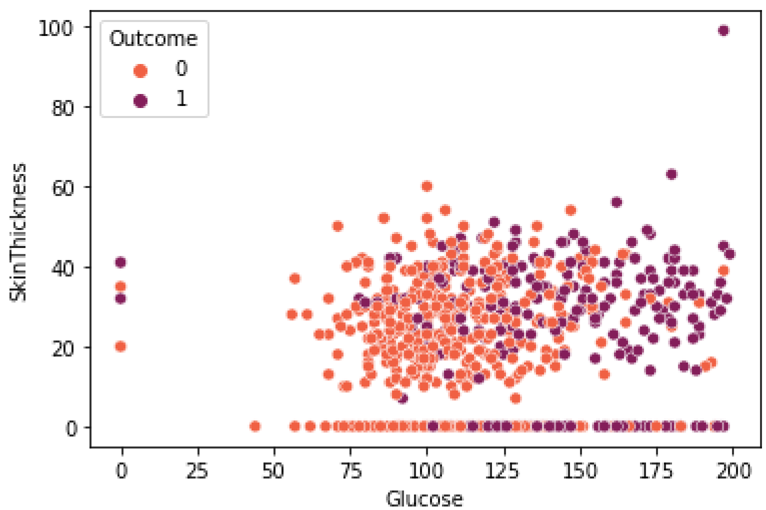
In [24]:
```
sns.scatterplot(x='Glucose',y='Insulin',hue='Outcome',data=data,palette='magma')
```

Out[24]: <AxesSubplot:xlabel='Glucose', ylabel='Insulin'>



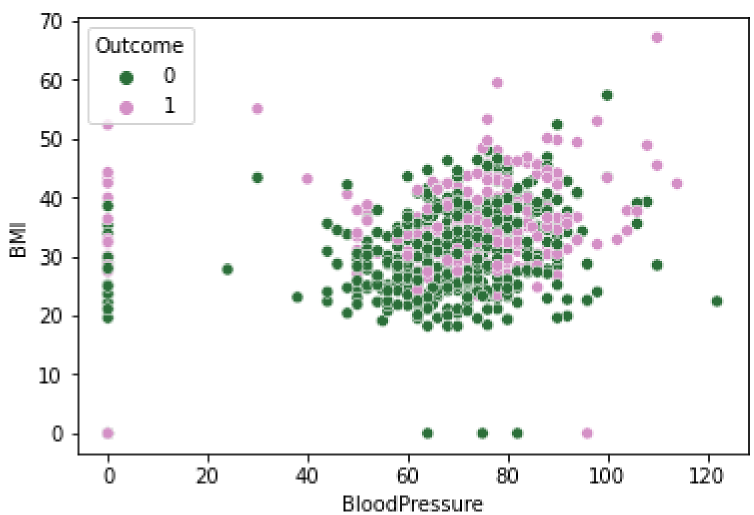
In [25]:
```
sns.scatterplot(x='Glucose',y='SkinThickness',hue='Outcome',data=data,palette='rocket_r')
```

Out[25]: <AxesSubplot:xlabel='Glucose', ylabel='SkinThickness'>



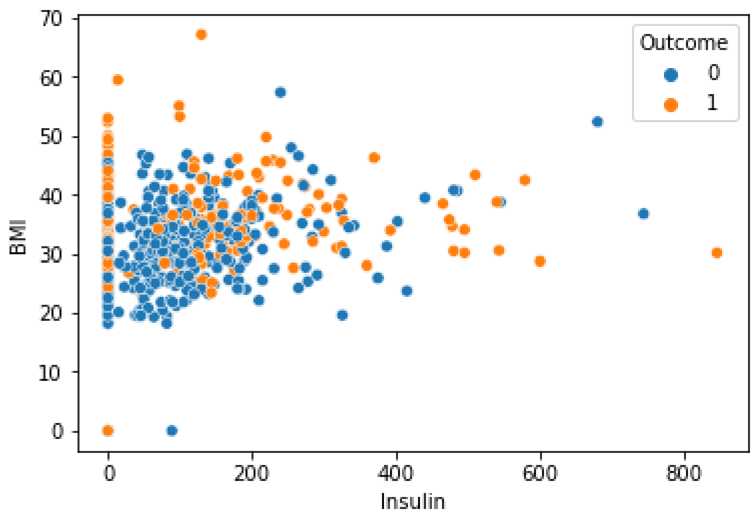
In [26]:
```
sns.scatterplot(x='BloodPressure',y='BMI',hue='Outcome',data=data,palette='cubehelix')
```

Out[26]: <AxesSubplot:xlabel='BloodPressure', ylabel='BMI'>



In [27]:
```
sns.scatterplot(x='Insulin',y='BMI',hue='Outcome',data=data,palette='tab10')
```

Out[27]: <AxesSubplot:xlabel='Insulin', ylabel='BMI'>



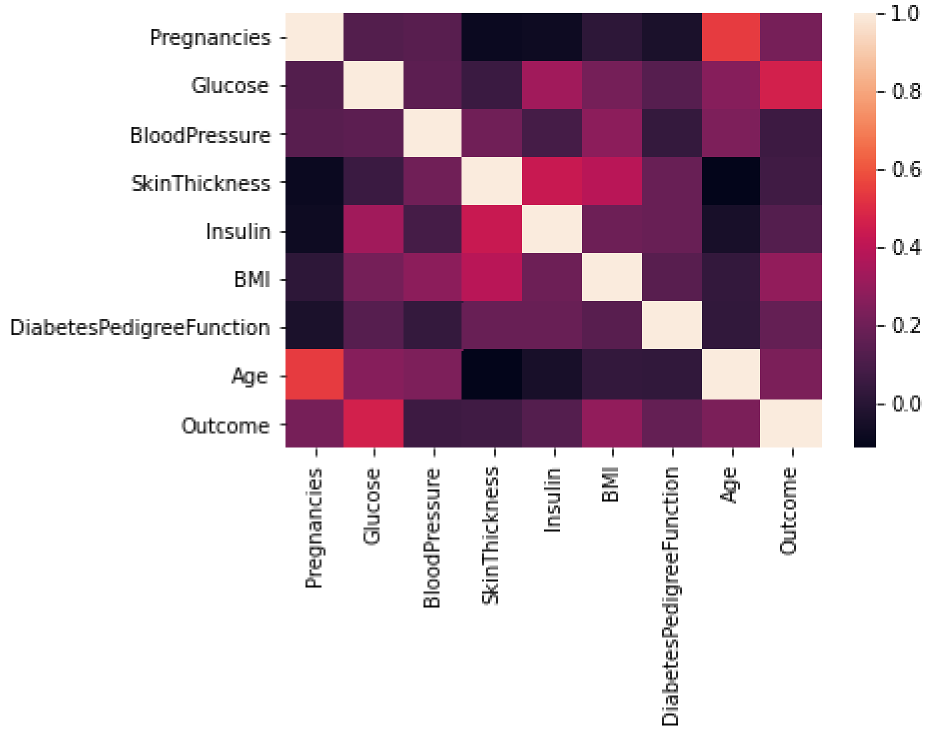
In [28]:
```
data.corr()
```

Out[28]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356

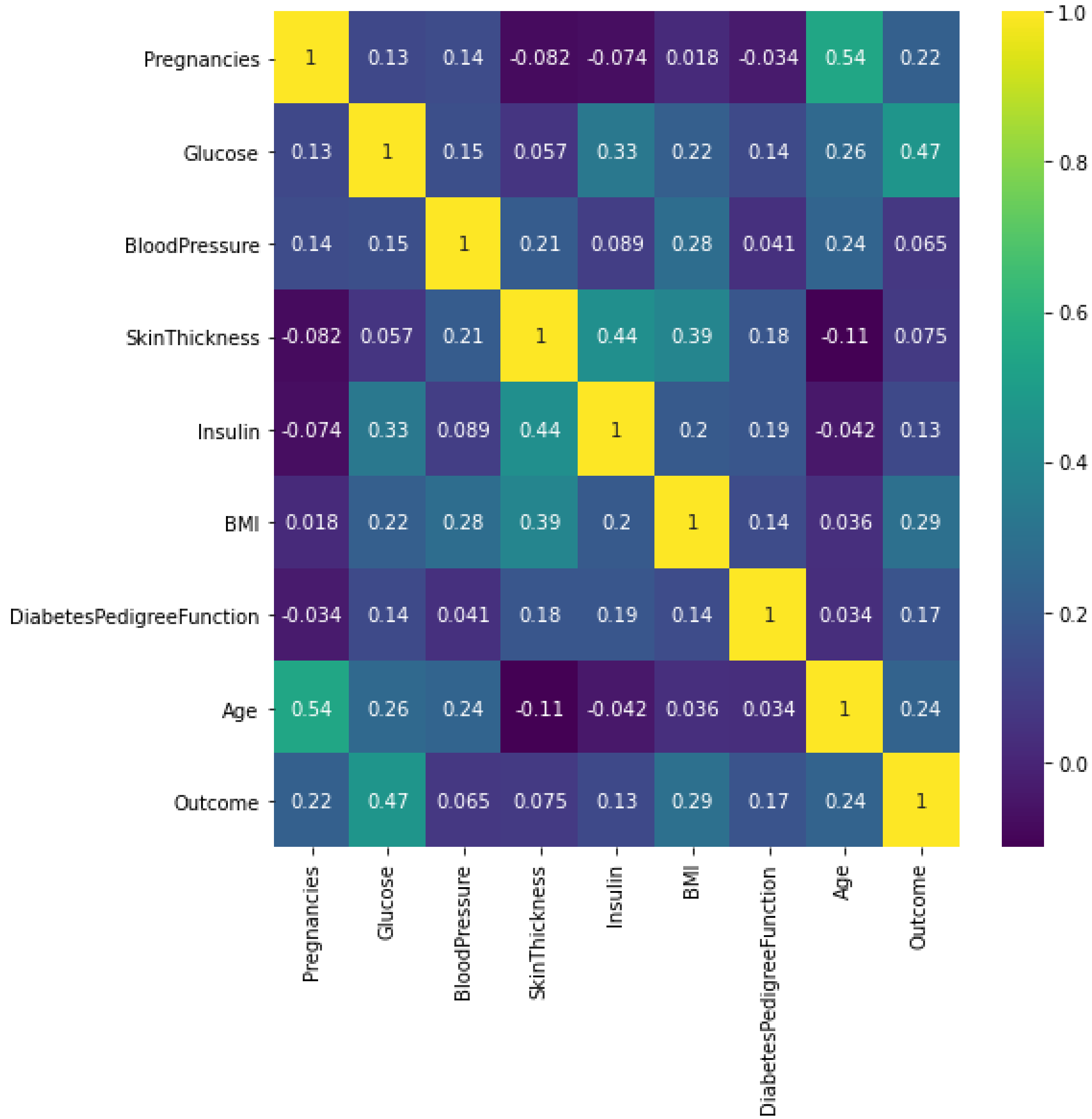
In [29]:
```
sns.heatmap(data.corr())
```

Out[29]: <AxesSubplot:>



In [30]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(data.corr(),annot=True,cmap='viridis')
```

Out[30]: <AxesSubplot:>



In [31]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(data.corr(),annot=True)
```

Out[31]: <AxesSubplot:>

Week3

In [32]:
```
data.head(5)
```

Out[32]:

	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 [33]:
```
features = data.iloc[:,[0,1,2,3,4,5,6,7]].values
label = data.iloc[:,8].values
```

In [34]:
```
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_state =10)
```

In [35]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[35]:
```
LogisticRegression()
```

In [36]:
```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7719869706840391
0.7662337662337663
```

In [37]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

Out[37]:
```
array([[446,  54],
       [122, 146]], dtype=int64)
```

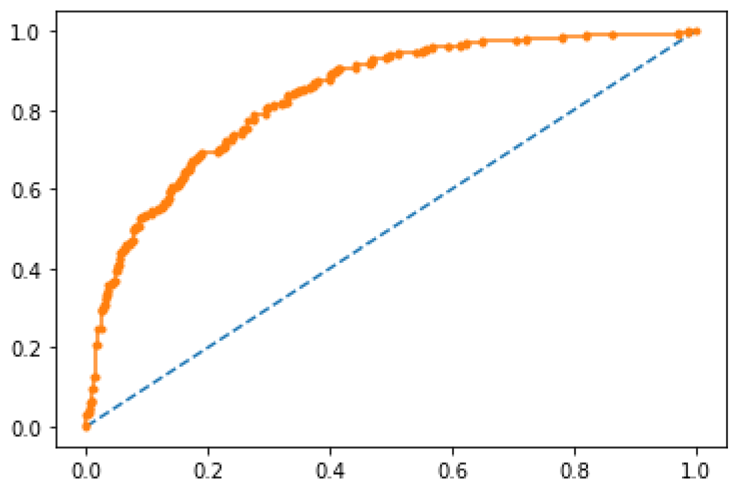
In [38]:
```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       500
           1       0.73      0.54      0.62       268

    accuracy                           0.77       768
   macro avg       0.76      0.72      0.73       768
weighted avg       0.77      0.77      0.76       768
```

In [39]:
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
```

```
AUC: 0.837
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x24ecd707310>]
```



In [40]:
```
#Applying Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

Out[40]:
```
DecisionTreeClassifier(max_depth=5)
```

In [41]:
```
model3.score(X_train,y_train)
```

Out[41]:
```
0.8289902280130294
```

In [42]:
```
model3.score(X_test,y_test)
```

Out[42]:
```
0.7727272727272727
```

In [43]:
```
#Applying Random Forest
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(X_train,y_train)
```

Out[43]:
```
RandomForestClassifier(n_estimators=11)
```

In [44]:
```
model4.score(X_train,y_train)
```

Out[44]:
```
0.99185667752443
```

In [45]:
```
model4.score(X_test,y_test)
```

Out[45]:
```
0.7532467532467533
```

In [46]:
```
#Support Vector Classifier

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

Out[46]:
```
SVC(gamma='auto')
```

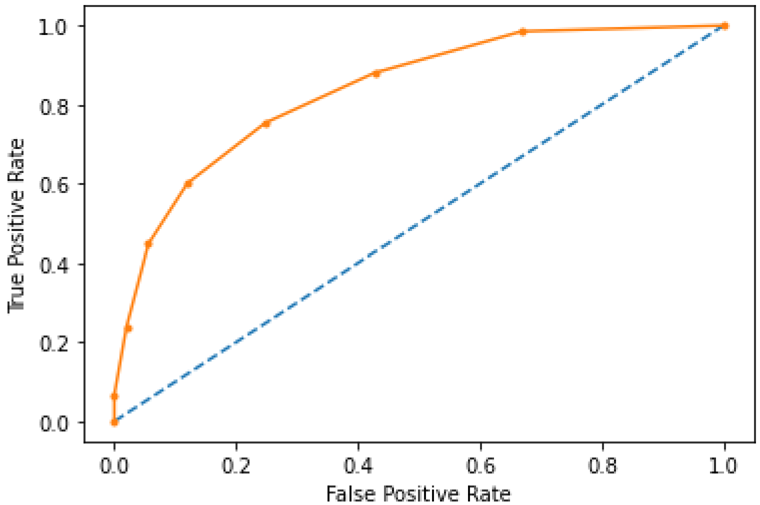
In [49]:
```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=7, metric='minkowski', p = 2)
model2.fit(X_train,y_train)
```

Out[49]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [50]:
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".format(tpr,fpr,thresholds))
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.836
True Positive Rate - [0.         0.06716418 0.23880597 0.44776119 0.60074627 0.75373134
 0.88059701 0.98507463 1.        ], False Positive Rate - [0.    0.    0.02  0.056 0.12  0.248 0.428 0.668 1.   
] Thresholds - [2.         1.         0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

Out[50]:
```
Text(0, 0.5, 'True Positive Rate')
```


True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0

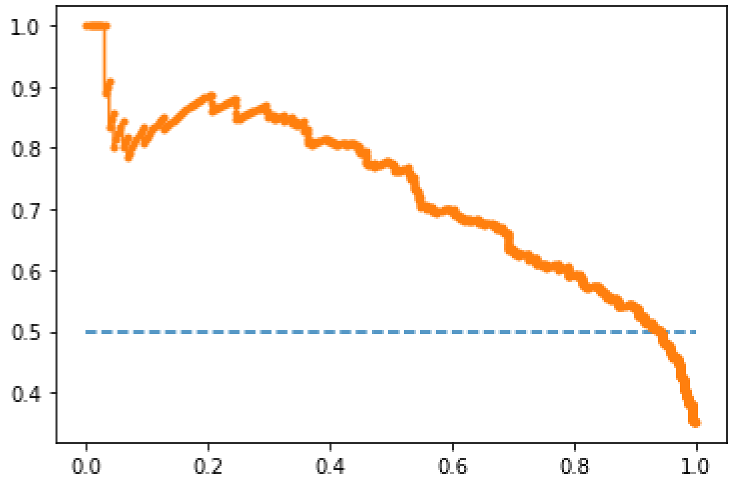
In [51]:

```
#Precision Recall Curve for Logistic Regression

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
# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.624 auc=0.726 ap=0.727
```

Out[51]:

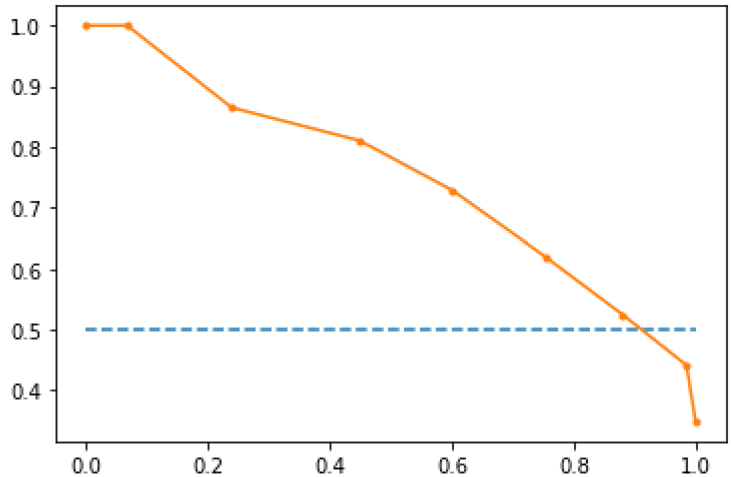
```
[<matplotlib.lines.Line2D at 0x24ecda306a0>]
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

In [52]:
```
#Precision Recall Curve for KNN

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
# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.658 auc=0.752 ap=0.709
```

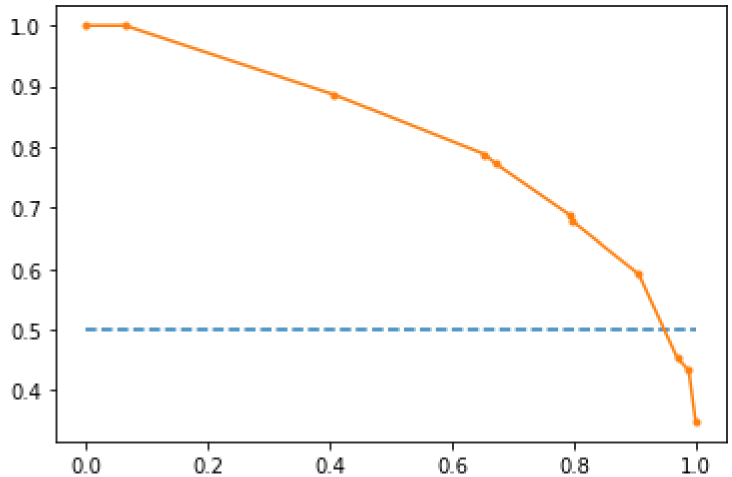
Out[52]:
```
[<matplotlib.lines.Line2D at 0x24ecda8f7c0>]
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

In [53]:
```
#Precision Recall Curve for Decission Tree Classifier

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
# predict probabilities
probs = model3.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.714 auc=0.815 ap=0.768
```

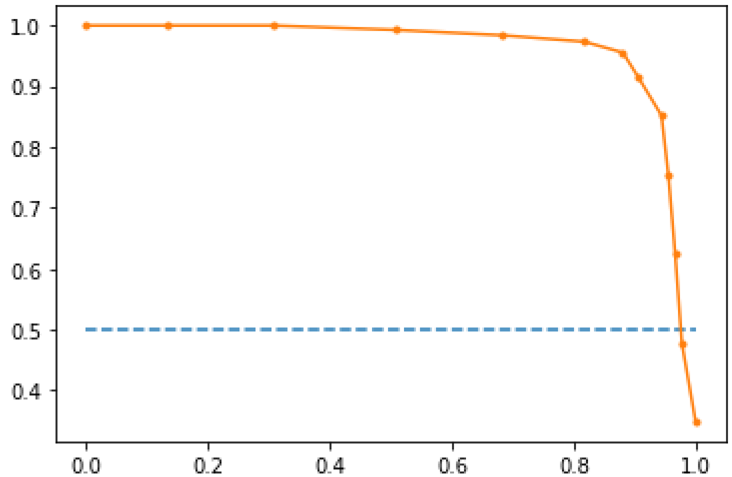
Out[53]:
```
[<matplotlib.lines.Line2D at 0x24ecdae9910>]
```


1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
0.2
0.4
0.6
0.8
1.0

In [54]:
```
#Precision Recall Curve for Random Forest

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
# predict probabilities
probs = model4.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model4.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.917 auc=0.962 ap=0.954
```

Out[54]:
```
[<matplotlib.lines.Line2D at 0x24ecdb44a60>]
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: