In [1]:

```
import numpy as np
```

In [2]:

```
import pandas as pd
```

In [3]:

```
import matplotlib.pyplot as plt
```

In [4]:

```
from sklearn.model_selection import cross_val_predict,StratifiedKFold,GridSearchC
```

In [5]:

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

In [6]:

```
from sklearn import tree
```

In [7]:

```
from sklearn.neighbors import KernelDensity
```

In [8]:

```
from sklearn import datasets #Importamos el conjunto de datos

```

In [9]:

```
from sklearn.model_selection import train_test_split
```

In [10]:

```
np.random.seed(0)
```

In [11]:

```
X, y = datasets.make_moons(1000, noise=0.20)
```

In [12]:

```
#Dividimos nuestros datos en "conjunto de entrenamiento y de prueba
```

In [13]:

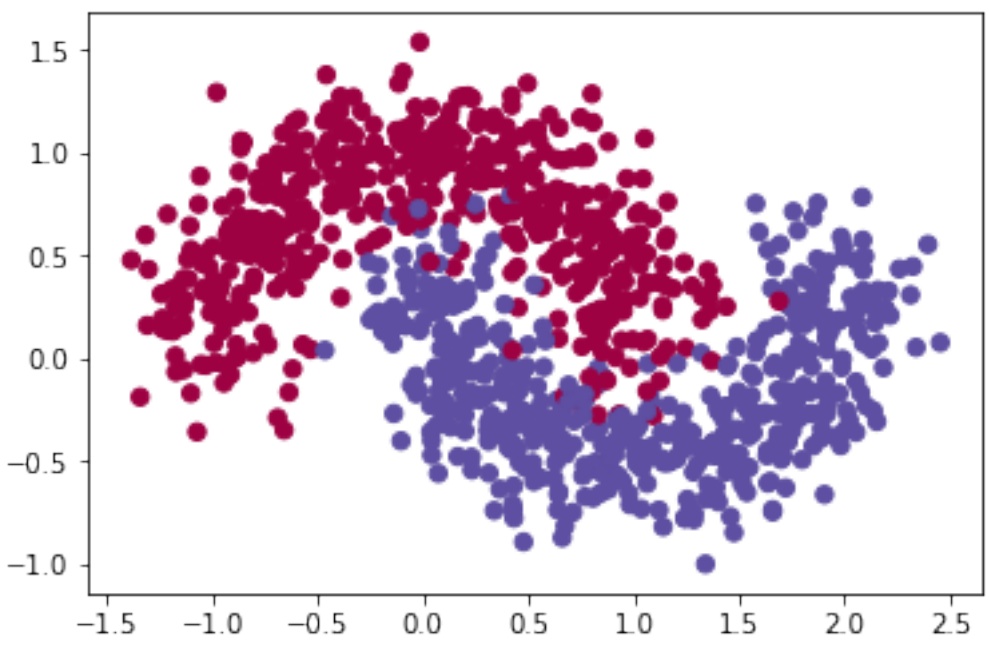
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [14]:

```
plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=plt.cm.Spectral)
```

Out[14]:

```
<matplotlib.collections.PathCollection at 0x1a1372ea90>
```

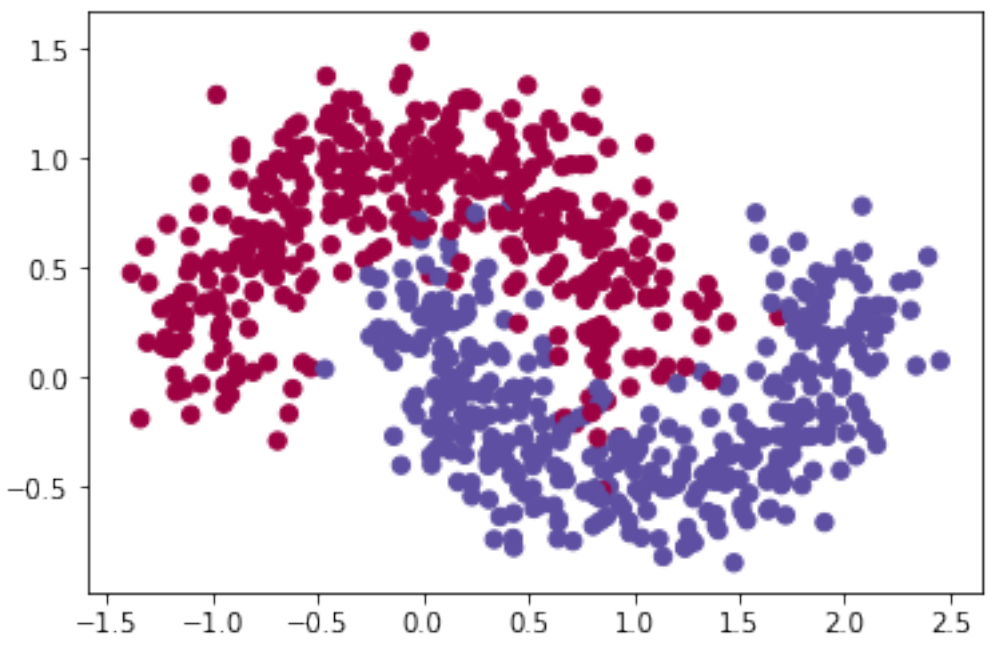


In [15]:

```
plt.scatter(X_train[:,0], X_train[:,1], s=40, c=y_train, cmap=plt.cm.Spectral)
```

Out[15]:

```
<matplotlib.collections.PathCollection at 0x1a13856b90>
```

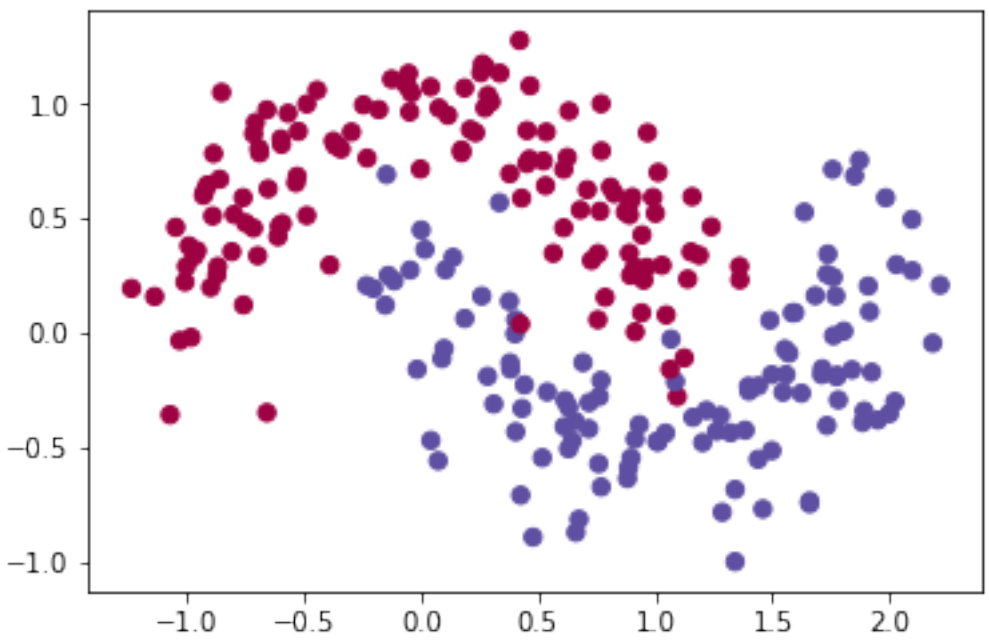


In [16]:

```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=y_test, cmap=plt.cm.Spectral)
```

Out[16]:

```
<matplotlib.collections.PathCollection at 0x1a13943290>
```



In [117]:

```
#calculo las priors a partir del conjunto de entrenamiento
n1=((y_train+1)/2.0).sum()
n0=((1-y_train)/2.0).sum()
P1=n1/(n1+n0) #cakculo priors a partir de conjunto de train
P0=1-P1

#P0=P1=1/2.0
```

In [118]:

```
P0
```

Out[118]:

```
0.2446666666666667
```

In [119]:

```
# estimo densidades
params = {'bandwidth': np.logspace(-1, 1, 20)}
grid = GridSearchCV(KernelDensity(), params)
grid.fit(X_train[y_train==1])
kde1= grid.best_estimator_
#kde1=KernelDensity().fit(X_train[y_train==1])
proba1=np.exp(kde1.score_samples(X_test)+np.log(P1))
```

In [120]:

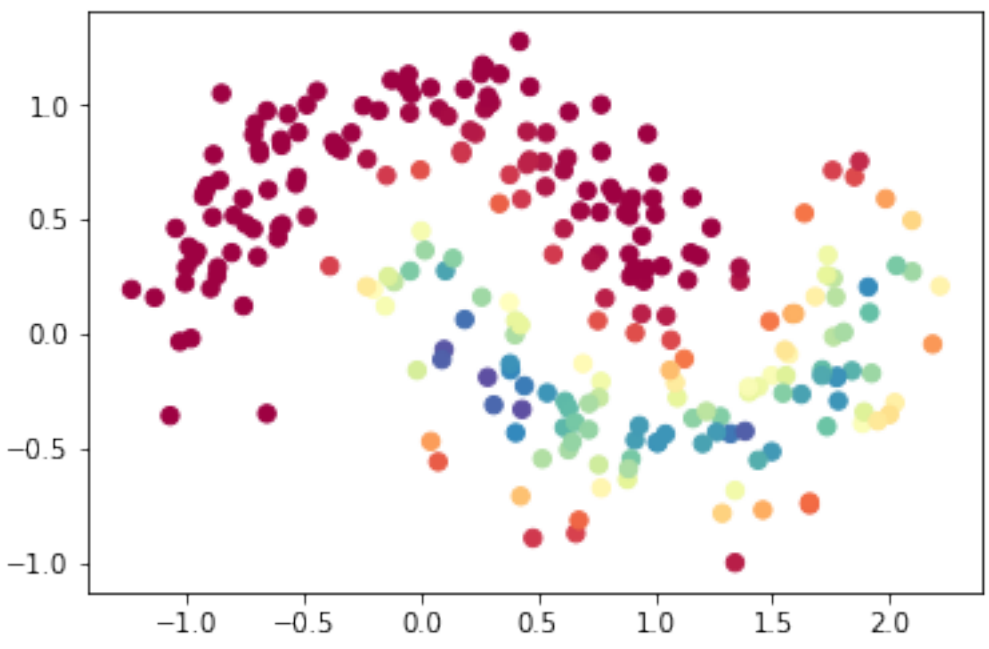
```
params = {'bandwidth': np.logspace(-1, 1, 20)}
grid = GridSearchCV(KernelDensity(), params)
grid.fit(X_train[y_train==0])
kde0= grid.best_estimator_

proba0=np.exp(kde0.score_samples(X_test)+np.log(P0))
```

In [121]:

```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=proba1, cmap=plt.cm.Spectral)
```

Out[121]:

```
<matplotlib.collections.PathCollection at 0x1a15e426d0>
```

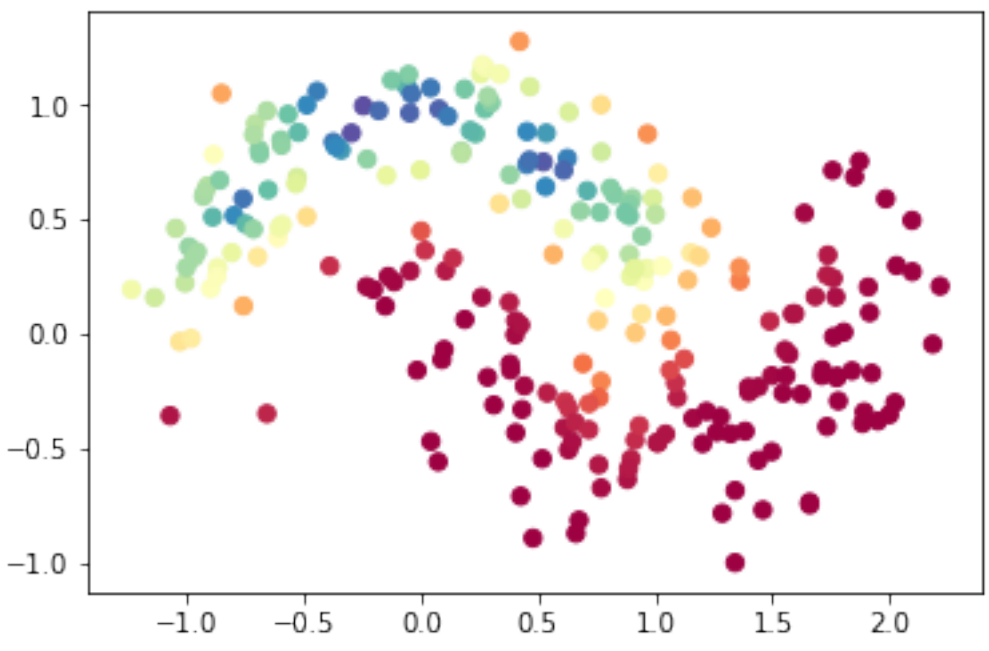


In [122]:

```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=proba0, cmap=plt.cm.Spectral)
```

Out[122]:

```
<matplotlib.collections.PathCollection at 0x1a15eb3c10>
```



In [127]:

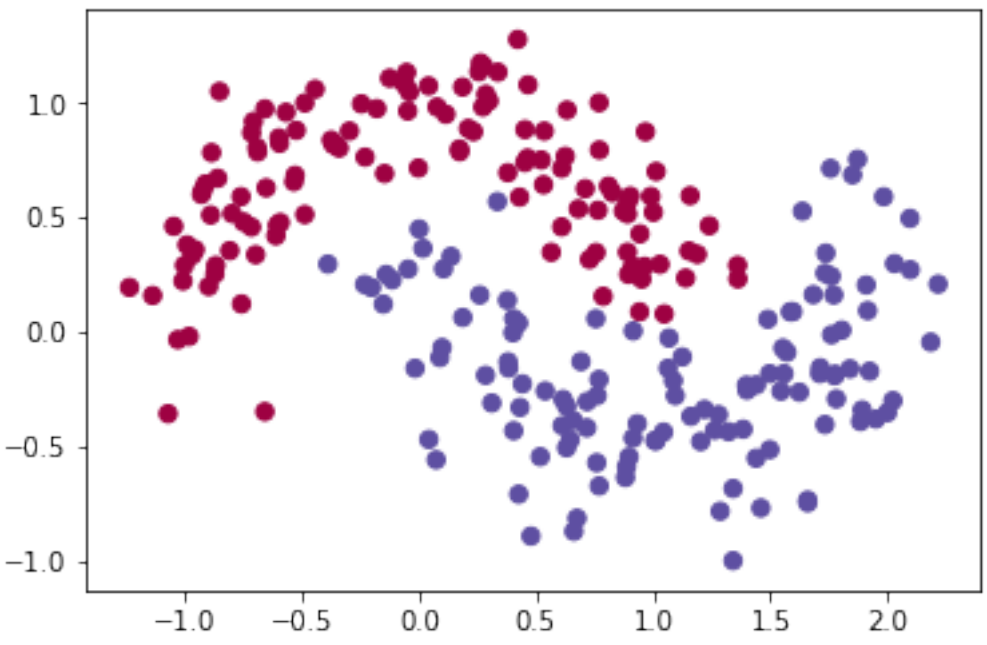
```
cost_01=1
cost_00=0
cost_10=1
cost_11=0

#Calculo riesgos para cada decision

Riesgo1=proba1*cost_11+proba0*cost_10
Riesgo0=proba1*cost_01+proba0*cost_00

y_pred_test_bmr=(Riesgo0-Riesgo1)

#si el riesgo de asignar 0 es mayor que el de 1 decido 1

y_pred_test_bmr1=(np.sign(y_pred_test_bmr)+1)/2
```

In [128]:

```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=y_pred_test_bmr1, cmap=plt.cm.Spect
```

Out[128]:

```
<matplotlib.collections.PathCollection at 0x1a1618f650>
```



In [129]:

```
diff_test=y_test-y_pred_test_bmr1
```

In [130]:

```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=diff_test, cmap=plt.cm.Spectral)
```

Out[130]:

```
<matplotlib.collections.PathCollection at 0x1a1625ef90>
```