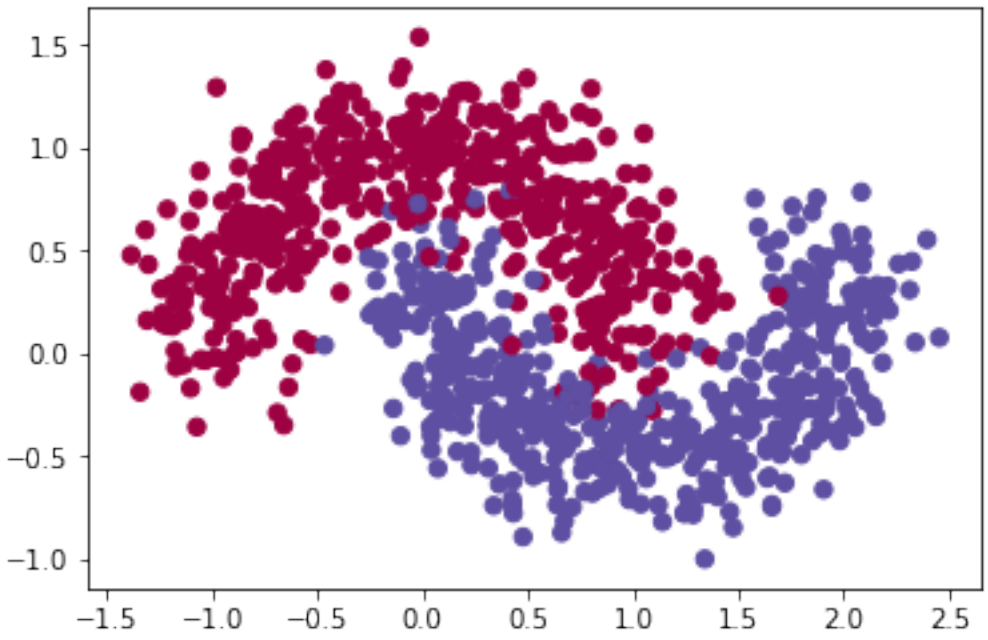
ROC-AUC_arbol

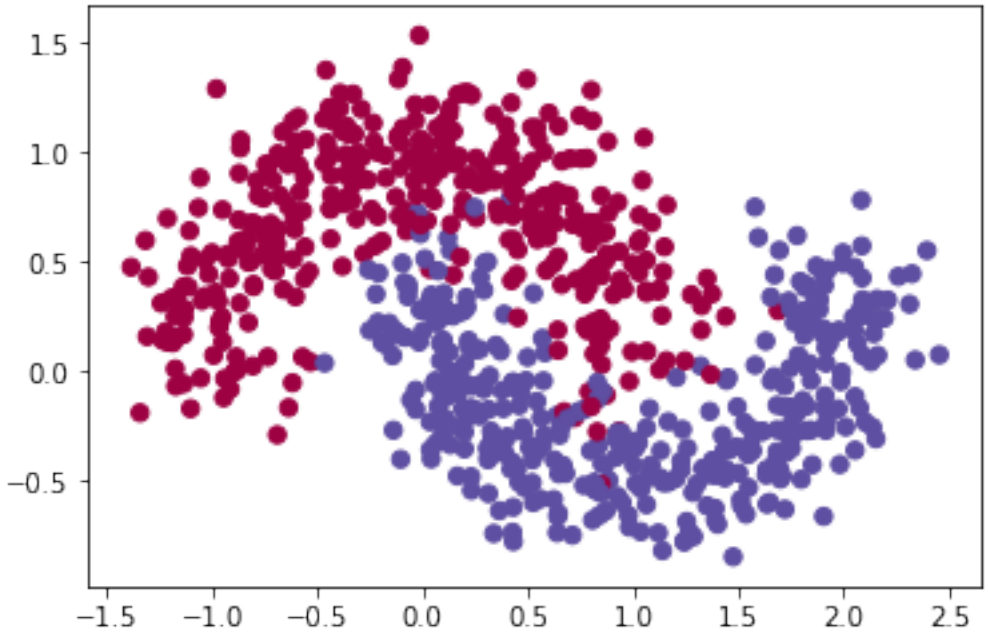
July 28, 2018

```
In [1]: import numpy as np

In [2]: import matplotlib.pyplot as plt

In [3]: from sklearn.metrics import roc_auc_score

In [4]: from sklearn import tree

In [5]: from sklearn import datasets #Importamos el conjunto de datos

In [6]: from sklearn.model_selection import train_test_split

In [7]: np.random.seed(0)

In [8]: X, y = datasets.make_moons(1000, noise=0.20)

In [9]: #Dividimos nuestros datos en "conjunto de entrenamiento y de prueba

In [10]: X_train, X_test, y_train, y_test = train_test_split(X, y)

In [11]: plt.scatter(X[:,0], X[:,1], s=40, c=y, cmap=plt.cm.Spectral)

Out[11]: <matplotlib.collections.PathCollection at 0x11082a650>
```



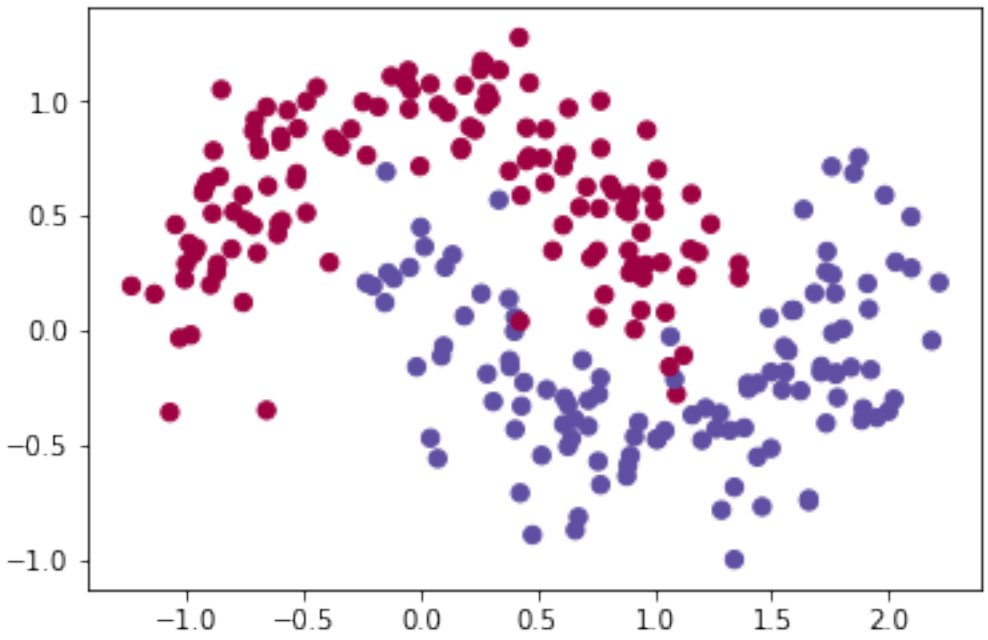
In [12]:
```
plt.scatter(X_train[:,0], X_train[:,1], s=40, c=y_train, cmap=plt.cm.Spectral)
```

Out[12]: <matplotlib.collections.PathCollection at 0x11094bfd0>



In [13]:
```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=y_test, cmap=plt.cm.Spectral)
```

Out[13]: <matplotlib.collections.PathCollection at 0x110a3b710>

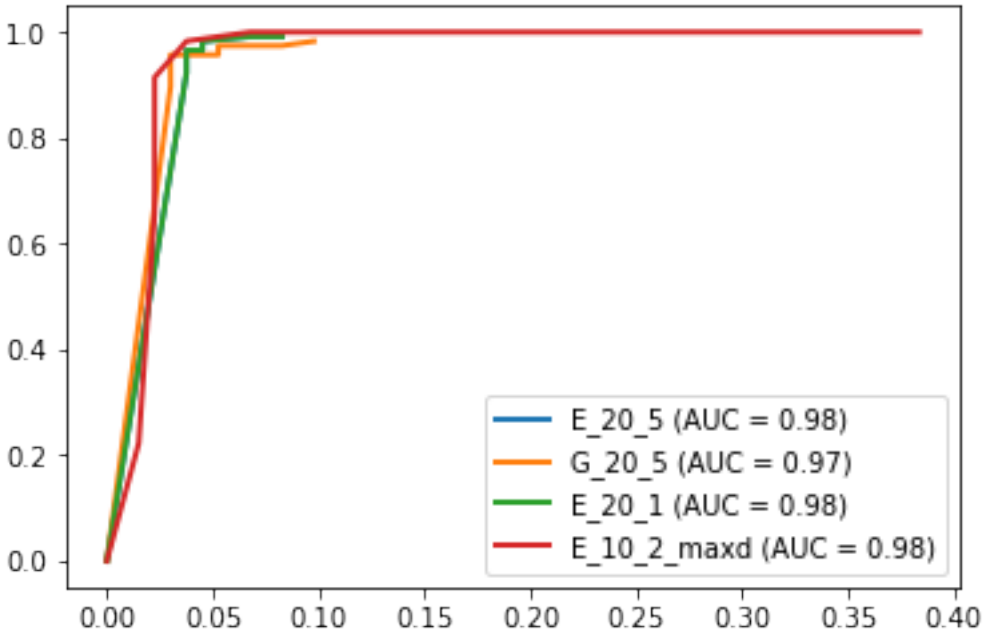



```
In [30]: classifiers=[(tree.DecisionTreeClassifier(criterion='entropy',
                                                   min_samples_split=20,
                                                   min_samples_leaf=5,
                                                   max_depth = 10),"E_20_5"),
                      (tree.DecisionTreeClassifier(criterion='gini',
                                                   min_samples_split=20,
                                                   min_samples_leaf=5,
                                                   max_depth = 10),"G_20_5"),
                      (tree.DecisionTreeClassifier(criterion='entropy',
                                                   min_samples_split=20,
                                                   min_samples_leaf=1,
                                                   max_depth = 10),"E_20_1"),
                      (tree.DecisionTreeClassifier(criterion='entropy',
                                                   min_samples_split=10,
                                                   min_samples_leaf=2,
                                                   min_impurity_decrease=.01
                                                  ),"E_10_2_maxd")]

In [15]: classifiers

Out[15]: [(DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=10,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=5, min_samples_split=20,
                          min_weight_fraction_leaf=0.0, presort=False, random_state=None,
                          splitter='best'), 'E_20_5'),
```

```
(DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=20,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), 'G_20_5'),
(DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=20,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), 'E_20_1'),
(DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=20,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'), 'E_20_5_maxd')]
```

In [16]:
```
#Para cada clasificador se grafica la ROC
```

In [31]:
```
for clf, name in classifiers:
    clf.fit (X_train,y_train)
    ROC=[]
    for gamma in np.linspace(0,1,1000):
        err1=np.count_nonzero(clf.predict_proba(X_test[y_test==0,:])[:,1]<=gamma)
        err2=np.count_nonzero(clf.predict_proba(X_test[y_test==1,:])[:,1]>gamma)
        err1=float(err1)/np.count_nonzero(y_test==0)
        err2=float(err2)/np.count_nonzero(y_test==1)
        ROC.append([err1,err2])
    ROC=np.array(ROC)
    ROC=ROC[::-1,:]
    auc=roc_auc_score(y_test,clf.predict_proba(X_test)[:,1])
    plt.plot(1-ROC[:,0],ROC[:,1], linewidth=2, label="%s (AUC = %.2f)" %(name,auc))
    plt.legend()
```



In [34]:
```
diff_test=y_test-clf.predict(X_test)
```

In [35]:
```
plt.scatter(X_test[:,0], X_test[:,1], s=40, c=diff_test, cmap=plt.cm.Spectral)
```

Out[35]:
```
<matplotlib.collections.PathCollection at 0x110ba6bd0>
```