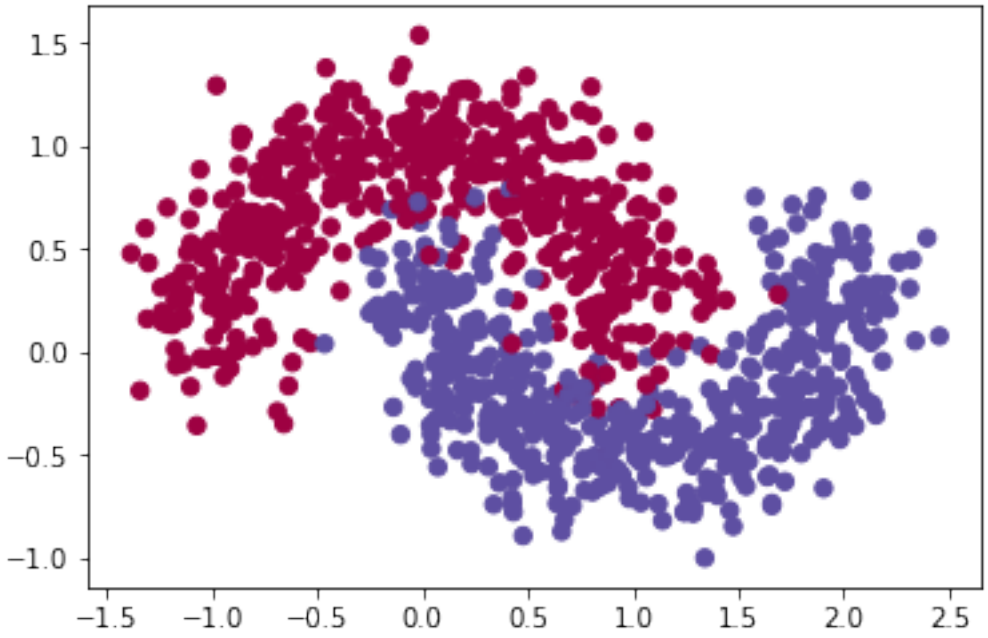
# ROC-AUC_moons-1

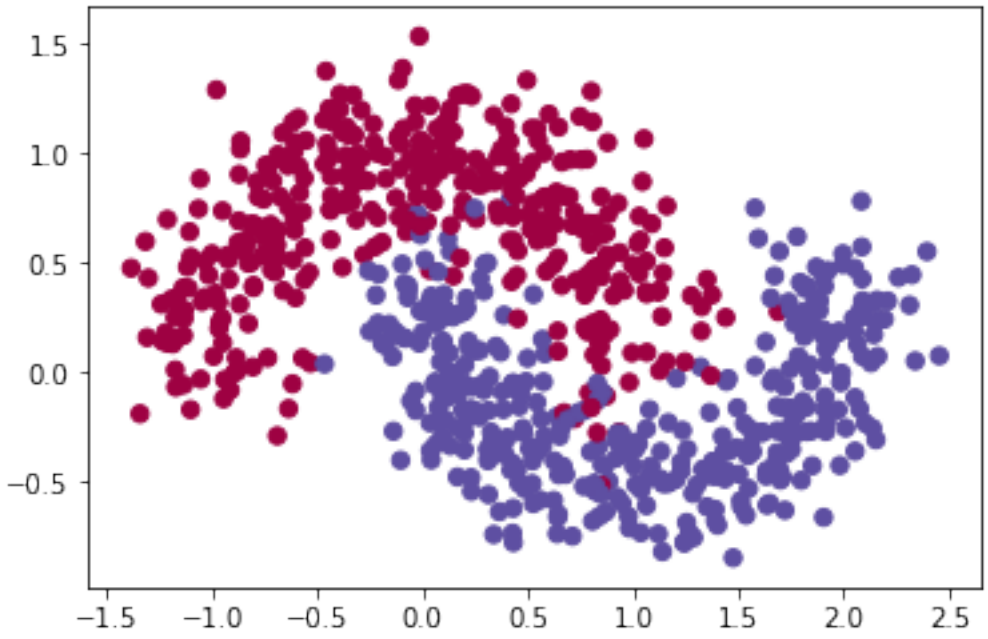
July 19, 2018

```
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.linear_model import LogisticRegression

In [5]: from sklearn.svm import SVC

In [6]: from sklearn.ensemble import RandomForestClassifier

In [7]: from sklearn.neighbors import KNeighborsClassifier

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 0x1a1dbffe50>
```

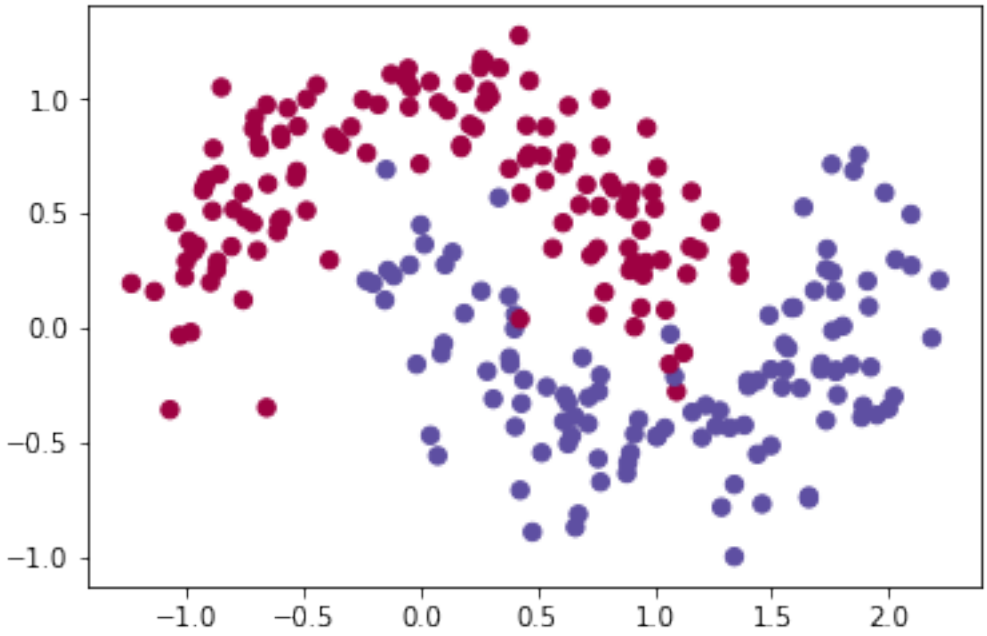



```
In [15]: plt.scatter(X_train[:,0], X_train[:,1], s=40, c=y_train, cmap=plt.cm.Spectral)
```

```
Out[15]: <matplotlib.collections.PathCollection at 0x1a1dd30650>
```



```
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 0x1a1ddb6c50>
```




```
In [17]: classifiers=[(LogisticRegression(),"Logistic Regression"),
                      (SVC(probability=True),"Support Vector Machine"),
                      (RandomForestClassifier(n_estimators=100),"Random Forest"),
                      (KNeighborsClassifier(),"Nearest Neighbor")]

In [18]: classifiers

Out[18]: [(LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                      intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
                      penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
                      verbose=0, warm_start=False), 'Logistic Regression'),
          (SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
             max_iter=-1, probability=True, random_state=None, shrinking=True,
             tol=0.001, verbose=False), 'Support Vector Machine'),
          (RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                      oob_score=False, random_state=None, verbose=0,
                      warm_start=False), 'Random Forest'),
```

```
          (KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                     weights='uniform'), 'Nearest Neighbor')]
```

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

In [19]:
```
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()
```