

# adaboost

September 7, 2018

## 0.1 Modelo de predicción de cancer de mama (Wisconsin uci)

```
In [1]: import pandas as pd
        from sklearn.cross_validation import train_test_split
        from sklearn.preprocessing import StandardScaler
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import AdaBoostClassifier
        from sklearn.pipeline import Pipeline
```

```
/home/rafa/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning:
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
In [2]: # Importamos los datos desde el repositorio oficial
        df = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-
        header=None)
```

```
In [3]: # Vemos la estructura del dataframe
        # Columna 0 --> ID
        # Columna 1 --> Clase
        # Columnas 2.. -> Datos
        df[:2]
```

```
Out[3]:
```

	0	1	2	3	4	5	6	7	8	9	\
0	842302	M	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	
1	842517	M	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	
...			22	23	24	25	26	27	28	29	\
0	...		25.38	17.33	184.6	2019.0	0.1622	0.6656	0.7119	0.2654	
1	...		24.99	23.41	158.8	1956.0	0.1238	0.1866	0.2416	0.1860	
			30	31							
0	0.4601	0.11890									
1	0.2750	0.08902									

```
[2 rows x 32 columns]
```

```
In [4]: # Separamos el dataset en (Conjunto de datos, Etiquetas)
        X = df.loc[:, 2:].values
        y = df.loc[:, 1].values
```

```

In [5]: # Separamos el conjunto total en dos subconjuntos (Train 80% / Test 20%)
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

In [6]: weak_clf = DecisionTreeClassifier(criterion='entropy', max_depth=None, random_state=0)

In [7]: # Creamos un flujo de trabajo (pipeline):
        # Paso 1 -> Escalar los datos
        # Paso 2 -> Analisis de componentes principales (Reduce la dimensionalidad)
        # Paso 3 -> Construimos un modelo utilizando Adaboost

        tree = DecisionTreeClassifier(random_state=0)

        adaboost = AdaBoostClassifier(base_estimator=tree,
                                       n_estimators=100,
                                       learning_rate=0.01,
                                       random_state=0)

        pipe_lr = Pipeline([('scl', StandardScaler()),
                             ('clf', adaboost)])

        # Ajustamos el modelo a los datos de entrenamiento
        pipe_lr.fit(X_train, y_train)

        #Calculamos la precisión del modelo utilizando los datos de test
        print('Test Accuracy: %.3f' % pipe_lr.score(X_test, y_test))

```

Test Accuracy: 0.904