<https://www.youtube.com/watch?v=RtrBtAKwcxQ>

* Se hace sin bagging decisión tree
* Luego con bagging
* Por ultimo lo hace con random forest

Con random forest da mejor exactitud, porque internamente random forest hace bagging

import pandas as pd

df = pd.read\_csv("diabetes.csv")

df.describe()

|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 768.0000 | 768.000 | 768.0000 | 768.000 | 768.00 | 768.00 | 768.000 | 768.0 | 768.00 |
| **mean** | 3.8450 | 120.8945 | 69.1054 | 20.536 | 79.799 | 31.99 | 0.471876 | 33.24 | 0.3489 |
| **std** | 3.3695 | 31.9726 | 19.3558 | 15.952 | 115.24 | 7.81 | 0.331329 | 11.76 | 0.4769 |
| **min** | 0.0000 | 0.0000 | 0.0000 | 0.000 | 0.000 | 0.000 | 0.078000 | 21.00 | 0.0000 |
| **25%** | 1.0000 | 99.000 | 62.000 | 0.000 | 0.000 | 27.30 | 0.243750 | 24.00 | 0.0000 |
| **50%** | 3.0000 | 117.00 | 72.000 | 23.000 | 30.50 | 32.00 | 0.372500 | 29.00 | 0.0000 |
| **75%** | 6.0000 | 140.250 | 80.000 | 32.000 | 127.25 | 36.60 | 0.626250 | 41.00 | 1.0000 |
| **max** | 17.0000 | 199.000 | 122.000 | 99.000 | 846.00 | 67.10 | 2.420000 | 81.00 | 1.0000 |

X = df.drop("Outcome",axis="columns")

y = df.Outcome

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, stratify=y, random\_state=10)

from sklearn.model\_selection import cross\_val\_score

from sklearn.tree import DecisionTreeClassifier

scores = cross\_val\_score(DecisionTreeClassifier(), X, y, cv=5)

**OUTPUT 0.71**

**Train using Bagging**

from sklearn.ensemble import BaggingClassifier

bag\_model = BaggingClassifier(

    base\_estimator=DecisionTreeClassifier(),

    n\_estimators=100,

    max\_samples=0.8,

    oob\_score=True,

    random\_state=0

)

bag\_model.fit(X\_train, y\_train)

bag\_model.oob\_score\_

**OUTPUT 0.75**

**Haciendo bagging mejoro mucho el decisión tree, mejoraremos parámetros de Bagging**

bag\_model.score(X\_test, y\_test)

bag\_model = BaggingClassifier(

    base\_estimator=DecisionTreeClassifier(),

    n\_estimators=100,

    max\_samples=0.8,

    oob\_score=True,

    random\_state=0

)

scores = cross\_val\_score(bag\_model, X, y, cv=5)

scores

**OUTPUT**

**array([0.75324675, 0.72727273, 0.74675325, 0.82352941, 0.73856209])**

scores.mean()

**OUTPUT 0.75**

**No mejoro mucho, usaremos en vez de Decision Tree el Random Forest**

### Train using Random Forest

from sklearn.ensemble import RandomForestClassifier

scores = cross\_val\_score(RandomForestClassifier(n\_estimators=50), X, y, cv=5)

scores.mean()

**OUTPUT 0.76**

**Al final, un Random Forest dio mejor exactitud que usar Bagging**

**Random Forest internamente hace bagging.**