```
# comparacion mediante pruebas en pareja para medir significancia estadistica


# arboles = 100, v1d PROM= 0.986607
# archivo=fork_of_rf_teziz_28k_v1d.ipynb
# acc de  n=10 de la validacion cruzada
rf = [0.99821492, 0.99964298, 1,         1,        1,        1,   1,        1,        1,    0


# arboles= 500 , v3  PROM= 0.98232
rf2 = [0.99393074, 0.99964298, 1  ,       1   ,      1  ,      1,   1  ,       1 ,        1
# archivo= fork-of-rf-teziz-28k-v3.ipynb
# arboles = 25,   v2 prom= 0.978500318763707

# fork-of-rf-teziz-28k-v2.ipynb
rf3 = [0.99143163, 0.99964298, 1    ,     0.99928571, 1,         1 , 1 ,       1   ,      1 ,


# de la prueba pareada entres los tres rf, ninguno es estadisticamete mas sigificativo, enton


from scipy import stats

stats.ttest_rel(rf2, rf)

# Ttest_relResult(statistic=-1.0625793669394634, pvalue=0.31565170839933626)
# 31.56
# c2 vs c3,  multiplicar por 100 para estar %

     Ttest_relResult(statistic=-1.1179939580979863, pvalue=0.2925225367973931)


#stats.ttest_rel(rf2, rf2)


#stats.ttest_rel(rf3, rf2)


stats.ttest_rel(rf3, rf)

# Ttest_relResult(statistic=-1.078184934383557, pvalue=0.30899927516686543)
# 30.89
# c1 vs c2,  multiplicar por 100 para estar %

 ⮡  Ttest_relResult(statistic=-1.109844996180234, pvalue=0.2958379924149413)


# svm RADIAL O RBF accPROM 0.904903771
# archivo = svm_28k_v2_teziz_rbf_v1d.ipynb
svm = [0.87325955, 0.82113531, 0.88964286, 0.90821429, 0.92892857, 0.94285714,  0.94428571, 0

# svm 2 lineal accPROM 0.85854763604835
# archivo = svm-28k-v5-teziz-linea.ipynb
```

```
svm2 = [0.86719029, 0.79900036, 0.84392857, 0.865 ,     0.8425  ,   0.85142857, 0.88107143, 0

# svm 3 poly alfa 8,  accPROM 0.7308073392155863
# archivo = svm-28k-v3-teziz-poly.ipynb
svm3 = [0.70581935, 0.68868261, 0.73892857, 0.73464286, 0.71392857, 0.72392857, 0.73642857, 0



from scipy import stats

#stats.ttest_rel(df['bp_before'], df['bp_after'])
stats.ttest_rel(svm2, svm) # svm radial  es mas significativo
# Ttest_relResult(statistic=-5.121446640281181, pvalue=0.0006267555728178404)
# 0.062
# c1 vs c2 de tabla,  multiplicar por 100 para estar %

     Ttest_relResult(statistic=-5.121446640281181, pvalue=0.0006267555728178404)


from scipy import stats

#stats.ttest_rel(df['bp_before'], df['bp_after'])
stats.ttest_rel(svm3, svm) # svm radial es mas significativo
#Ttest_relResult(statistic=-12.545525125959326, pvalue=5.26789191645834e-07)
# 5.26 e-5
# c2 vs c3 de tabla,  multiplicar por 100 para estar %

     Ttest_relResult(statistic=-12.545525125959326, pvalue=5.26789191645834e-07)


from scipy import stats

#stats.ttest_rel(df['bp_before'], df['bp_after'])
stats.ttest_rel(rf, svm)
#stats.ttest_rel(gh2, gh)
#Ttest_relResult(statistic=5.2117566811001295, pvalue=0.0005552702643440211)
# pvalue % = 0.05

# Ttest_relResult(statistic=5.2117566811001295, pvalue=0.0005552702643440211)
# 0.05
# RF VS .....SVM
# TABLA FINAL

     Ttest_relResult(statistic=4.904653541370277, pvalue=0.0008422251732582458)


#print(stats.__version__)

import scipy

print(scipy.__version__)

#print('scipy'.__version__)
```

```
1.4.1
```

```
stats.ttest_rel(svm, rf)
```

```
Ttest_relResult(statistic=-4.904653541370277, pvalue=0.0008422251732582458)
```

```
# El valor p devuelto (0.0005552) es menor que el umbral 'a' (0.05). Por lo tanto, se rechaza
# que el rendimiento de 'rf' es significativamente diferente y mejorado en comparación con el
```

```
# https://www.coursehero.com/file/p4hol16a/Aqu%C3%AD-prepresenta-la-proporci%C3%B3n-poblacion
```

```
ann = [0.92771084, 0.99776786, 0.92767857, 0.98883929, 0.99776786, 0.98660714,  0.925,      0
```

```
#99.74
ann77 = [1,  0.996875,   0.99821429, 0.99821429, 0.99866071 ,0.99642857, 1,  0.98928571, 0.99
```

```
#ann78 = [1, 1, 1 ,1, 1 ,1,  1, 1, 1, 1]
```

```
# ann78 = [1, 1, 1 ,1, 1 ,1,  1, 1, 1, 1,1,1,1,1,1,1,1]
```

```
# prom= 89.61651
#c1 v2
# archivo = tesis-ann-28k-v2.ipynb
annv2=[0.90138331 ,0.903125 ,  0.89553571, 0.89464286, 0.89642857, 0.89955357, 0.89375 ,   0.
```

```
# prom = 95.9779873
# c2 v3
# archivo = tesis-ann-28k-v3.ipynb

annv3=[0.93083445, 0.98303571, 0.92321429, 0.92723214 ,0.9875 ,    0.99375, 0.99553571, 0.938
```

```
#stats.ttest_rel(annv2, ann)
```

```
#stats.ttest_rel(ann, ann77)
```

```
stats.ttest_rel(annv2, annv3)
# resulta mas significativo annv3
# Ttest_relResult(statistic=-6.538442668121652, pvalue=0.0001065875379783153)
# 0.010
# c1 vs c2 de tabla,  multiplicar por 100 para estar %
```

```
Ttest_relResult(statistic=-6.538442668121652, pvalue=0.0001065875379783153)

stats.ttest_rel(annv3, ann77)
# la mas significante es la ann77
# Ttest_relResult(statistic=-2.4026432030499154, pvalue=0.03972535093203618)
# 3.97
# c2 vs c3 de tabla,  multiplicar por 100 para estar %

Ttest_relResult(statistic=-3.8207162473968976, pvalue=0.0040854280856479856)


stats.ttest_rel(svm, ann)



Ttest_relResult(statistic=-3.8274191226309147, pvalue=0.004043678125891304)


# El valor p devuelto (0.0040436) es menor que el umbral 'a' (0.05). Por lo tanto, se rechaza
# que el rendimiento de 'ann' es significativamente diferente y mejorado en comparación con e


stats.ttest_rel(rf, ann)

Ttest_relResult(statistic=1.6911668673557134, pvalue=0.12506186291082466)


stats.ttest_rel(ann,rf)

Ttest_relResult(statistic=-1.6911668673557134, pvalue=0.12506186291082466)


# El valor p devuelto (0.093248655) es mayor que el umbral 'a' (0.05). Por lo tanto, se acept
# que el rendimiento de 'rf' es  NO ES significativamente diferente y mejorado en comparación


# a H0 es las medias de las dos muestras son iguales

# Si el valor p es menor que un cierto nivel de significancia α fijado por el usuario a prior
#estadısticamente significativo y H0 se debe rechazar.

#Que un resultado sea estadısticamente significativo quiere
#decir que es improbable que suceda por azar.

#Un valor p menor a 0,05 indica una fuerte evidencia en
#contra de H0 porque habr´ıa una probabilidad menor al
# 5 % que H0 sea correcta y nos lleva a aceptar H1.

#Un valor p superior a 0,05 indica una fuerte evidencia a
#favor de H0 y al rechazo de H1.




stats.ttest_rel(ann, rf)
```

```
stats.ttest_rel(ann, rf)
```

```
Ttest_relResult(statistic=-1.6911668673557134, pvalue=0.12506186291082466)
```

```
# Resultado: el estadístico de la prueba t de muestras emparejadas es -1.8770201400069235 y e
# El valor p es mayor que el nivel de significancia = 0.05.
# La prueba t de muestras pareadas no rechaza la hipótesis nula: la media de "ann" y la media
```

```
import numpy as np
#np.mean(rf)
```

```
stats.ttest_rel(ann, svm)
```

```
Ttest_relResult(statistic=3.8274191226309147, pvalue=0.004043678125891304)
```

```
stats.ttest_rel(svm, ann77)
#Ttest_relResult(statistic=-5.714908029509426, pvalue=0.00028880820708700487)
# 0.02
# ANN VS ....SVM
# TABLA FINAL
```

```
Ttest_relResult(statistic=-7.50074489121567, pvalue=3.689856448366902e-05)
```

```
stats.ttest_rel(rf, ann77)
#Ttest_relResult(statistic=0.23564252830958335, pvalue=0.8189847522277838)
# 81.89
# ANN VS ....RF
# TABLA FINAL
```

```
Ttest_relResult(statistic=-0.8072524347610931, pvalue=0.4403328220987782)
```

```
# no hay significancia pero el rendimiento es mejor en Ann
stats.ttest_rel(ann77, rf)
```

```
Ttest_relResult(statistic=0.8072524347610931, pvalue=0.4403328220987782)
```

```
# stats.ttest_rel(ann78, rf)
```

```
# https://machinelearningmastery.com/how-to-code-the-students-t-test-from-scratch-in-python/
# t-test for dependent samples
from math import sqrt
from numpy.random import seed
from numpy.random import randn
from numpy import mean
from scipy.stats import t

# function for calculating the t-test for two dependent samples
def dependent_ttest(data1, data2, alpha):
```

```
def dependent_ttest(data1, data2, alpha):
  # calculate means
  mean1, mean2 = mean(data1), mean(data2)
  # number of paired samples
  n = len(data1)
  # sum squared difference between observations
  d1 = sum([(data1[i]-data2[i])**2 for i in range(n)])
  # sum difference between observations
  d2 = sum([data1[i]-data2[i] for i in range(n)])
  # standard deviation of the difference between means
  sd = sqrt((d1 - (d2**2 / n)) / (n - 1))
  # standard error of the difference between the means
  sed = sd / sqrt(n)
  # calculate the t statistic
  t_stat = (mean1 - mean2) / sed
  # degrees of freedom
  df = n - 1
  # calculate the critical value
  cv = t.ppf(1.0 - alpha, df)
  # calculate the p-value
  p = (1.0 - t.cdf(abs(t_stat), df)) * 2.0
  # return everything
  return t_stat, df, cv, p

# seed the random number generator
seed(1)
# generate two independent samples (pretend they are dependent)
data1 = 5 * randn(100) + 50
data2 = 5 * randn(100) + 51
# calculate the t test
alpha = 0.05
#t_stat, df, cv, p = dependent_ttest(data1, data2, alpha)
t_stat, df, cv, p = dependent_ttest(ann77, rf, alpha)
print('t=%.3f, df=%d, cv=%.3f, p=%.3f' % (t_stat, df, cv, p))
# interpret via critical value
if abs(t_stat) <= cv:
  print('Accept null hypothesis that the means are equal.')
else:
  print('Reject the null hypothesis that the means are equal.')
# interpret via p-value
if p > alpha:
  print('Accept null hypothesis that the means are equal.')
else:
  print('Reject the null hypothesis that the means are equal.')
```

```
t=0.807, df=9, cv=1.833, p=0.440
Accept null hypothesis that the means are equal.
Accept null hypothesis that the means are equal.
```

```
# t-test for dependent samples
from math import sqrt
from numpy.random import seed
```

```
from numpy.random import randn
from numpy import mean
from scipy.stats import t

# function for calculating the t-test for two dependent samples
def dependent_ttest(data1, data2, alpha):
  # calculate means
  mean1, mean2 = mean(data1), mean(data2)
  # number of paired samples
  n = len(data1)
  # sum squared difference between observations
  d1 = sum([(data1[i]-data2[i])**2 for i in range(n)])
  # sum difference between observations
  d2 = sum([data1[i]-data2[i] for i in range(n)])
  # standard deviation of the difference between means
  sd = sqrt((d1 - (d2**2 / n)) / (n - 1))
  # standard error of the difference between the means
  sed = sd / sqrt(n)
  # calculate the t statistic
  t_stat = (mean1 - mean2) / sed
  # degrees of freedom
  df = n - 1
  # calculate the critical value
  cv = t.ppf(1.0 - alpha, df)
  # calculate the p-value
  p = (1.0 - t.cdf(abs(t_stat), df)) * 2.0
  # return everything
  return t_stat, df, cv, p

# seed the random number generator
seed(1)
# generate two independent samples (pretend they are dependent)
data1 = 5 * randn(100) + 50
data2 = 5 * randn(100) + 51
# calculate the t test
alpha = 0.05
alpha = 0.01
#t_stat, df, cv, p = dependent_ttest(data1, data2, alpha)
#t_stat, df, cv, p = dependent_ttest(ann, rf, alpha)
t_stat, df, cv, p = dependent_ttest(rf, rf2, alpha)
print('t=%.3f, df=%d, cv=%.3f, p=%.3f' % (t_stat, df, cv, p))
# interpret via critical value
if abs(t_stat) <= cv:
  print('Accept null hypothesis that the means are equal.')
else:
  print('Reject the null hypothesis that the means are equal.')
# interpret via p-value
if p > alpha:
  print('Accept null hypothesis that the means are equal.')
else:
  print('Reject the null hypothesis that the means are equal.')
```

```
t=1.118, df=9, cv=2.821, p=0.293
Accept null hypothesis that the means are equal.
Accept null hypothesis that the means are equal.
```