.

[One Hot encoder](#_ny1nwgb2t2cx)

[Máximo número de columnas al imprimir un dataframe](#_exgthtaka29s)

[Unir dos dataframes (con el mismo nº de observaciones)](#_bc091mx4hrd9)

[Eliminar características](#_twi4r9u3oqee)

[Eliminar características basadas en condición](#_krofrdwta3hk)

[Filtrar filas por condición](#_gpbu3emcreg4)

[Eliminar la primera fila](#_341jiqocvtpn)

[Averiguar el número de nan en cada columna de un Dataframe](#_yzzya0hfca80)

[Obtener las filas con valor Nan en una columna](#_x3tou4wk4i1)

[Dividir dataframe en train y test](#_fqvzxco6lxb)

[Barajar (Shuffle) un dataframe](#_crrv9guc11wd)

[Matriz de Confusión](#_5jahbgjgnda6)

[Matriz de correlación](#_a17gwsfrx6b6)

[Liberar recursos en Colab al terminar un ejercicio:](#_2a3fd8dfotof)

[Semilla para random en numpy:](#_kms8iknp1epx)

[Semilla para random en Tensorflow:](#_gkwny0agoov9)

[Subir ficheros en Colab:](#_evq0vpbxrlp7)

[Conversión de categoría a números:](#_ve6kn4lpuvyv)

[Número de Na en una columna:](#_qu757z53arm8)

[Accuracy](#_8ck4by9k816n)

[Models](#_s69sge4kg0vt)

[Linear regression](#_8fdzmdk5meot)

[Stochastic Gradient Descent (Linear Regression)](#_fiuep5b4yhcn)

[Con minibatch](#_h4wkmrx5uu9k)

[Logistic regression](#_tjx0t7ncaiu3)

[KNN](#_w5l67d9qusmf)

[Cross Validation](#_otupch82aa0)

[Z-score](#_fklf5ieqtswu)

[Seaborn pair plot](#_80f7fan9sf2m)

[Escalado / normalización](#_ot7qadjscqw0)

[Métrica en modelos de clasificación](#_acmg206b874w)

[Dibujar gráfico de un árbol de decisión](#_g1c71kuxya8)

[To use autocompletion in Kaggle](#_p14ysjuypouw)

[Para saber si una variable sigue una distribución normal gausiana](#_t505ywurxhio)

[¿Cuántos valores tenemos de cada clase?](#_c0o0bsxpembl)

[Plotear los vectores soportes en SVM (SVC):](#_qqwwzhx8bbv7)

[Change dataframe to numeric type](#_7abb3nqv6qot)

[Concatenar dos datasets](#_9k5q1gii7dko)

[Montar google drive](#_9c6x3rpdctb1)

[Subir fichero](#_f9bv7lem23l8)

[Descargar fichero](#_ktwkiqprmojy)

[Datasets interactivos en Colab](#_hist53s9v7au)

[Semilla para números aleatorios (random numbers)](#_3hcb12n1f19e)

[PCA](#_thfkuty87zro)

[TSNE](#_7jevr7jo316r)

[Guardar (salvar) un modelo en tensorflow](#_nv9h5yk6sxsd)

[Cargar un modelo tensorflow](#_tnrd3xmf9i08)

## One Hot encoder

(artículo: <https://towardsdatascience.com/what-is-one-hot-encoding-and-how-to-use-pandas-get-dummies-function-922eb9bd4970>)

from sklearn.preprocessing import OneHotEncoder

onehot\_enc = OneHotEncoder()

y\_onehot = onehot\_enc.fit\_transform(y).toarray()

Otra forma más sencilla:

fuel\_type = pd.get\_dummies(df.FUELTYPE, prefix='FUELTYPE')

Luego:

df = pd.**concat**(**[df, fuel\_type]**, axis=1)

**Con etiquetas:**

Otra forma (asigna número a cada categoría):

def encode\_label(df):

return df.astype('category').cat.codes

Luego:

df['education'] = encode\_label(df['education'])

Otra forma (si necesitamos recodificar después): NOTA: LabelEncoder sólo se usa para target (y). Para valores de X usar OrdinalEncoder()

from sklearn import preprocessing

encoder = preprocessing.LabelEncoder()

encoded = encoder.fit\_transform(df['workclass'])

df['workclass'] = encoded.astype('int')

(Podemos decodificar con: encoder.inverse\_transform(df.workclass))

**OrdinalEncoder**

from sklearn.preprocessing import OrdinalEncoder

oe = OrdinalEncoder(handle\_unknown='ignore')

X\_train = oe.fit\_transform(X\_train)

X\_test = oe.transform(X\_test)

Cuando usar one-hot o LabelEncoder:

<https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizor>

## Máximo número de columnas al imprimir un dataframe

pd.options.display.max\_columns = 100

Ancho máximo de columna:

pd.options.display.max\_colwidth = 100

## Unir dos dataframes (con el mismo nº de observaciones)

df = df.join(otherDF)

## Eliminar características

df.drop(['MAKE', 'MODEL'], axis='columns', inplace=True)

## Eliminar características basadas en condición

df.drop(df[df['myFeature'] == ' '].index, inplace = True)

## Filtrar filas por condición

df = df[df['hours-per-week']>=40]

## Eliminar la primera fila

df = df.iloc[1:]

## Averiguar el número de nan en cada columna de un Dataframe

df.isnull().sum()

o

df.isna().sum()

## Obtener las filas con valor Nan en una columna

df[df['column name'].isna()]

O

df[df['column name'].isnull()]

**En cualquier columna:**

df[df.isna().any(axis=1)]

O

df[df.isnull().any(axis=1)]

## Eliminar filas con NaN

df = df.dropna()

## Dividir dataframe en train y test

# Después de dividir el dataframe en X e y

msk = np.random.rand(len(df)) < 0.8

X\_train = X[msk]

X\_test = X[~msk]

y\_train = y[msk]

y\_test = y[~msk]

Otra forma:

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size=0.3, random\_state=42)

Otra forma:

train=df.sample(frac=0.8,random\_state=200) #random state is a seed

test=df.drop(train.index)

# Después dividir train y test en X e y

## Dividir dataframe en train, validation y test

!pip install fast\_ml

from fast\_ml.model\_development import train\_valid\_test\_split

X\_train, y\_train, X\_valid, y\_valid, X\_test, y\_test =

train\_valid\_test\_split(df, target = 'SalePrice',

train\_size=0.8, valid\_size=0.1, test\_size=0.1)

## Barajar (Shuffle) un dataframe

df.sample(frac=1)

Para resetear los índices:

df = df.sample(frac=1).reset\_index(drop=True)

Otra forma de resetear los índices:

new\_df = new\_df.reset\_index().drop(['index'], axis=1)

## Matriz de Confusión

from sklearn.metrics import confusion\_matrix

confusion = confusion\_matrix(y\_pred,y\_test)

# Con labels

# pd\_conf = pd.DataFrame(confusion, columns = 'pred '+y\_test.unique(), index = 'true '+y\_test.unique())

# pd\_conf

Plot:

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(drugTree, X\_test, y\_test)

Otra forma de Plot:

import seaborn as sns

ax = sns.heatmap(confusion, annot=True, cmap='YlGnBu')

ax.set\_title('Seaborn Confusion Matrix with labels!!')

ax.set\_xlabel('Predicted class')

ax.set\_ylabel('Real Class')

ax.xaxis.set\_ticklabels(['apples', 'oranges'])

ax.yaxis.set\_ticklabels(['apples', 'oranges'])

## Matriz de correlación

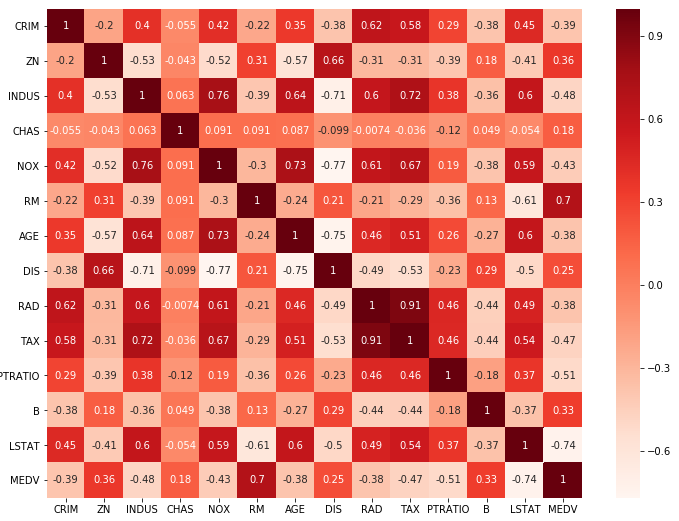
import seaborn as sns

plt.figure(figsize=(12,10))

cor = df.corr()

sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)

plt.show()



## Liberar recursos en Colab al terminar un ejercicio:

import os, signal

os.kill(os.getpid(), signal.SIGKILL)

## Semilla para random en numpy:

np.random.seed(42)

## Semilla para random en Tensorflow:

tf.random.set\_seed(42)

## Subir ficheros en Colab:

from google.colab import files

uploaded = files.upload()

Ejemplo:

from keras.preprocessing import image

for fn in uploaded.keys():

# predicting images

path = '/content/' + fn

img = image.load\_img(path, target\_size=(300, 300))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

images = np.vstack([x])

classes = model.predict(images, batch\_size=10)

print(classes[0])

if classes[0]>0.5:

print(fn + " is a human")

else:

print(fn + " is a horse")

## Conversión de categoría a números:

df["PoolQC"].unique() ## Pare ver los valores que toma

poolQCValues = ["Excellent": 3, "Good": 2, "Bad": 1, np.nan: 0]

df[‘PoolQC’] = df["PoolQC"].map(poolQCValues)

**Otra forma:**

df['sex'].replace({'Male': 1, 'Female':0}, inplace=True)

**Otra forma:**

df["PoolQC"] = df["PoolQC"].astype(‘categpry’).cat.codes

**Otra forma:**

from sklearn import preprocessing

le\_sex = preprocessing.LabelEncoder()

le\_sex.fit(['F','M'])

X[:,1] = le\_sex.transform(X[:,1])

## Número de Na en una columna:

print(“Número de Na en la columna”, df[“PoolQC”].isna().sum()

## Accuracy

from sklearn import metrics

metrics.accuracy\_score(y\_test, y\_pred)

## Models

### Linear regression

**from** **sklearn.linear\_model** **import** LinearRegression

model = LinearRegression()

### Stochastic Gradient Descent (Linear Regression)

from sklearn.linear\_model import SGDRegressor

lin\_reg\_SGD = SGDRegressor()

#### Con minibatch

from sklearn.linear\_model import SGDRegressor

lin\_reg\_SGD = SGDRegressor()

#lin\_reg = LinearRegression()

for i in range(1, 1000):

XX, YY = \_get\_batch(X\_poly, y, batch\_size = 10)

lin\_reg\_SGD.partial\_fit(XX, YY.ravel())

### Logistic regression

**from** **sklearn.linear\_model** **import** LogisticRegression

logreg = LogisticRegression()

### KNN

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

knnclassifier = KNeighborsClassifier(n\_neighbors=4)

## Cross Validation

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

knnclassifier = KNeighborsClassifier(n\_neighbors=4)

print(cross\_val\_score(knnclassifier, x, y, cv=10, scoring ='accuracy').mean()) ## cv is the number of k-folds

**Example to compare 2 models:**

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

knnclassifier = KNeighborsClassifier(n\_neighbors=4)

print(cross\_val\_score(knnclassifier, x, y, cv=10, scoring ='accuracy').mean())

0.7893574108818011

**from** **sklearn.linear\_model** **import** LogisticRegression

logreg = LogisticRegression()

print (cross\_val\_score(logreg, x, y, cv=10, scoring = 'accuracy').mean())

0.6499937460913072

## Z-score

**From scipy import stats**

**data = np.array([6, 7, 7, 12, 13, 13, 15, 16, 19, 22])**

**stats.zscore(data)**

**[-1.394, -1.195, -1.195, -0.199, 0, 0, 0.398, 0.598, 1.195, 1.793]**

In statistics, a **z-score** tells us how many standard deviations away a value is from [the mean](https://www.statology.org/measures-central-tendency/). We use the following formula to calculate a z-score:

**z** = (X – μ) / σ

## Seaborn pair plot

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns

sns.pairplot(df, height=2.5)

plt.show()

## Escalado / normalización

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler() ## MinMaxScaler para valores entre 0 y 1

X\_std = scaler.fit\_transform(X\_train)

scaler.fit() calcula media y varianza (para después aplicar los mismos media/varianza a los conjuntos de entrenamiento y de validación). transform() aplica la normalización con esa media y varianza y fit\_transform() hace las dos cosas a la vez.

**IMPORTANTE**: Hacer scale.fit\_transform() sobre el train set. Y transform() sobre el test set, de lo contrario el modelo “aprende” cómo debe ser la distribución del test set.

## Métrica en modelos de clasificación

from sklearn.metrics import classification\_report

print(classification\_report(y\_test, y\_pred,

target\_names=labels))

Ejemplo:

precision recall f1-score support

Ideal 0.58 0.39 0.46 57

Premium 0.49 0.60 0.54 45

Good 0.68 0.73 0.70 74

Fair 0.61 0.64 0.62 74

accuracy 0.60 250

macro avg 0.59 0.59 0.58 250

weighted avg 0.60 0.60 0.59 250

## Dibujar gráfico de un árbol de decisión

from sklearn.externals.six import StringIO

import pydotplus

import matplotlib.image as mpimg

from sklearn import tree

%matplotlib inline

dot\_data = StringIO()

filename = "drugtree.png"

featureNames = my\_data.columns[0:5]

targetNames = my\_data["Drug"].unique().tolist()

out=tree.export\_graphviz(drugTree,feature\_names=featureNames,

out\_file=dot\_data,

class\_names= np.unique(y\_trainset),

filled=True,

special\_characters=True,

rotate=False)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png(filename)

img = mpimg.imread(filename)

plt.figure(figsize=(100, 200))

plt.imshow(img,interpolation='nearest')

## To use autocompletion in Kaggle

%config Completer.use\_jedi = False

## Para saber si una variable sigue una distribución normal gausiana

from scipy import stats

print('Kurtosis:', stats.kurtosis(df['DTI']))

print('Skewness:', stats.skew(df['DTI']))

Un valor de curtosis y/o coeficiente de asimetría entre -1 y 1, es generalmente considerada una ligera desviación de la normalidad (Bulmer, 1979), (Brown, n.d.). Entre -2 y 2 hay una evidente desviación de la normal pero no extrema.

## ¿Cuántos valores tenemos de cada clase?

(Contar el número de valores diferentes)

df['Ticket'].value\_counts()

## Plotear los vectores soportes en SVM (SVC):

ax = plt.gca()

plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=Y, s=50, cmap='autumn')

xlim = ax.get\_xlim()

ylim = ax.get\_ylim()

xx = np.linspace(xlim[0], xlim[1], 30)

yy = np.linspace(ylim[0], ylim[1], 30)

YY, XX = np.meshgrid(yy, xx)

xy = np.vstack([XX.ravel(), YY.ravel()]).T

Z = clf.decision\_function(xy).reshape(XX.shape)

ax.contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,

linestyles=['--', '-', '--'])

ax.scatter(clf.support\_vectors\_[:, 0], clf.support\_vectors\_[:, 1], s=100,

linewidth=1, facecolors='none', edgecolors='k')

plt.show()

## Change dataframe to numeric type

#changing column type to numeric

df = df.apply(pd.to\_numeric)

Df.dtypes

## Concatenar dos datasets

df = pd.concat([df1, df2], axis=1)

## Descargar fichero de Google Drive (sencillo):

!pip install --upgrade --no-cache-dir gdown

## Pare evitar un error de permiso

!gdown --id yourFileIdHere

## Otra forma de descargar fichero de Google Drive autenticando usuario (más complicado):

# Importamos librerías para interactuar con Google Drive

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive

from google.colab import auth

from oauth2client.client import GoogleCredentials

# Nos autenticamos con nuestra cuenta de Google Drive para acceder a los datos

auth.authenticate\_user()

gauth = GoogleAuth()

gauth.credentials = GoogleCredentials.get\_application\_default()

drive = GoogleDrive(gauth)

# Nos descargamos los datos

download = drive.CreateFile({'id': 1CLcO5G9EmFgquSRI56XKxre1bQF-74QI'})

download.GetContentFile('english-spanish.pkl')

## Montar google drive

from google.colab import drive

drive.mount('/content/gdrive')

root\_path = 'gdrive/My Drive/your\_project\_folder/'

pd.read\_csv(f'{root\_path}file.csv')

## Subir fichero

from google.colab import files

files.upload()

## Descargar fichero

from google.colab import files

files.download('path/to/your/file')

## Datasets interactivos en Colab

from google.colab import data\_table

data\_table.enable\_dataframe\_formatter()

df

## Semilla para números aleatorios (random numbers)

seed = 42

np.random.seed(seed)

tf.random.set\_seed(seed)

## PCA

from sklearn.decomposition import PCA

pca = PCA(n\_components = 2)

X\_pca = pca.fit\_transform(X\_std)

## TSNE

from sklearn.manifold import TSNE

# Project the data: this step will take several seconds

tsne = TSNE(n\_components=2, init='random', random\_state=42)

X\_tsne = tsne.fit\_transform(X\_std)

## Guardar (salvar) un modelo en tensorflow

<https://www.tensorflow.org/guide/keras/save_and_serialize>

Hay 2 formatos (tensorflow y h5). Tensorflow genera un directorio con ficheros, pero guarda más información.

1. En formato h5

model.save(‘name.h5’)

1. En formato tesnorflow

model.save(‘name\_directory’)

Luego hacer:

!zip -r name.zip name\_directory

## Cargar un modelo tensorflow

tensorflow.keras.models.load\_model(‘name.h5’)

O en formato tensorflow:

!unzip name.zip

tensorflow.keras.models.load\_model(‘name\_directory’)

## Guardar / Cargar Modelo (Mío):

# Guardar modelo

import pandas as pd

def save\_model(model, filename):

model.save(filename)

model.save(filename+'.h5')

!zip -r {filename}.zip {filename}

history\_df = pd.DataFrame(model.history.history)

history\_df.to\_csv(filename+'.history.csv')

# Cargar el modelo

import pandas as pd

def load\_model(filename):

modelName = os.path.basename(filename)

if modelName.endswith('.zip'):

modelName = modelName[:-4]

else:

if modelName.endswith('.h5'):

modelName = modelName[:-3]

if filename.endswith('.h5'):

model = tf.keras.models.load\_model(filename)

else:

if filename.endswith('.zip'):

!unzip {filename}

else:

!unzip {filename}.zip

model = tf.keras.models.load\_model(modelName)

try:

history = pd.read\_csv(modelName+'.history.csv')

except:

history = None

return model, history

## Early Stopping:

from tensorflow.keras.callbacks import EarlyStopping

es = EarlyStopping(monitor='val\_accuracy', mode='max', patience=5, restore\_best\_weights=True)

results = model.fit(X\_train,y\_train,batch\_size=16,epochs=100, callbacks=[es])

## Resetear la memoria (interesante si se generan varios modelos)

tf.keras.backend.clear\_session()

## Añadir una dimensión:

X\_train = np.expand\_dims(X\_train, axis=-1)

# Ej: (60000, 28, 28) -> (60000, 28, 28, 1)