import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import seaborn as sns

import pandas

from keras.models import Sequential

from keras.layers import Dense

from keras.wrappers.scikit\_learn import KerasClassifier

from keras.utils import np\_utils

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

from sklearn.model\_selection import KFold

from sklearn.preprocessing import LabelEncoder

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification\_report, confusion\_matrix, roc\_curve, roc\_auc\_score,auc, accuracy\_score

df = pd.read\_csv("C:\\Users\\unic\\Downloads\\abalone.csv")

df.head(10).T

df.columns

#from IPython.core.display import HTML # permet d'afficher du code html dans jupyter

#display(HTML(df.head(10).to\_html()))

df.shape

df.describe()

df.info()

df['Sex'] = df['Sex'].map({"M":0, "F":1, "I":2})

tabcorr = df.corr()

tabcorr

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

sns.heatmap(abs(tabcorr), cmap="coolwarm")

sns.clustermap(abs(tabcorr), cmap="coolwarm")

correlations = tabcorr.Sex

print(correlations)

correlations = correlations.drop(['Sex'],axis=0)

print(abs(correlations).sort\_values(ascending=False))

from tensorflow.keras.utils import to\_categorical

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

y = df['Sex']

X = df.drop(['Sex'], axis=1).values.astype(float)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

y\_test\_ = y\_test

y\_train = to\_categorical(y\_train, 3)

y\_test = to\_categorical(y\_test, 3)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

model = Sequential()

model.add(Dense(16,input\_dim=8, activation="tanh"))

model.add(Dense(3, activation="softmax"))

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

train = model.fit(X\_train , y\_train , validation\_data=(X\_test,y\_test), epochs=100, verbose=1)

predict\_x=model.predict(X\_test)

y\_ann=np.argmax(predict\_x,axis=1)

accuracy\_score(y\_test\_, y\_ann)

confusion\_matrix(y\_test\_, y\_ann)

def plot\_scores(train) :

accuracy = train.history['accuracy']

val\_accuracy = train.history['val\_accuracy']

epochs = range(len(accuracy))

plt.plot(epochs, accuracy, 'b', label='Score apprentissage')

plt.plot(epochs, val\_accuracy, 'r', label='Score validation')

plt.title('Scores')

plt.legend()

plt.show()

plot\_scores(train)