Aprendizaje y clasificación de documentos

- Reunir un conjunto de 15 documentos de 5 clases. Cada documento de al menos 1 pagina. Seleccionar 10 documentos de cada clase aleatoriamente para realizar el entrenamiento bajo el esquema de bolsa de palabras.
- Emplear los 5 documentos restantes de cada clase para clasificar empleando el teorema de Bayes.
- Resumir los resultados empleando una matriz de confusión.
- Calcular métricas TP, TN, FP, FN por clase, Accuracy, Precisión, Recall, F1-score por clase.
- Discutir los resultados obtenidos.

Para la realización de este trabajo se utiliza python como lenguaje de programación, ademas de hacer uso de librerías como numpy, pandas y scikitlearn, esta ultima debido a que tiene funciones especificas que simplifican el desarrollo del programa, una de ellas es *CountVectorizer* que sirvió para crear la bolsa de palabras con los textos de entrenamiento, y crear vectores para cada texto con las diferentes palabras que tienen, otra es *MultinomialNB* la cual se utilizo para entrenar el modelo con la bolsa de valores generada, y posteriormente evaluar los nuevos textos de test, por ultimo se utilizo la función *metrics* para obtener estadísticas como la matriz de confusión y el Accuracy del modelo.

Los textos utilizados son quince libros en cinco categorías diferentes, las cuales son: Artificial Intelligence, Business, Education, Fantasy y Health; todos los libros están en idioma ingles y se convirtieron a formato .txt, ademas de que todos los textos están en minúsculas para evitar la posibilidad de que se tomaran como diferentes palabras si una estaba en mayúsculas y en minúsculas en diferentes partes de los textos. Ademas en el código para realizar la clasificación se les coloco un numero a cada clase: Artificial Intelligence=0, Business=1, Education=2, Fantasy=3 y Health=4.

Código en python:

```
import operator
import numpy as np
import pandas as pd
import sklearn
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

#-------------LIBROS----------------
#0=AI 1=BUSINESS 2=EDUCATION 3=FANTASY 4=HEALTH
libros = ["Artificial Intelligence A Modern Approach.txt",
```

"1._100_ways_to_motivate_yourself.txt",
"A Student's Introduction to English Grammar (PDFDrive).txt",
"4_harry_potter_and_the_goblet_of_fire.txt",
"36 Signature Training Workout Programs (PDFDrive).txt",
"Abstract Computing Machines A Lambda Calculus Perspective (PDFDrive).txt",
"13 Things Mentally Strong People Don't Do Take Back Your Power, Embrace Change, Face Your Fears, and Train Your Brain for Happiness and Success (PDFDrive).txt",
"An Introduction to English Grammar, Longman Grammar, Syntax and Phonology, Second Edition (PDFDrive).txt",
"A Feast for Crows (PDFDrive).txt",
"101 High Intensity Workouts for Fast Results (101 Workouts) (PDFDrive).txt",
"Emergency Medicine (PDFDrive).txt",
"A Storm of Swords (PDFDrive).txt",
"An introduction to English sentence structure (PDFDrive).txt",
"business law (PDFDrive).txt",
"Compilers - Principles, Techniques, and Tools (2006).txt",
"The Flavor Bible The Essential Guide to Culinary Creativity, Based on the Wisdom of America's Most Imaginative Chefs (PDFDrive).txt",
"The Lord of the Rings The Two Towers (PDFDrive).txt",
"Spoken English Flourish Your Language (PDFDrive).txt",
"Richard_Templar-The_Rules_of_Work-EN.txt",
"Programming__Principles_and_Practice_Using C++ (Cpp Cplusplus).txt",
"ProgrammingInPython3.txt",
"Rich Dad Poor Dad (PDFDrive).txt",
"Learning English as a Foreign Language for Dummies (PDFDrive).txt",
"The Last Battle (Narnia) (PDFDrive).txt",
"The End of Diabetes The Eat to Live Plan to Prevent and Reverse Diabetes (PDFDrive).txt",
"Essentials of Anatomy and Physiology (PDFDrive).txt",
"A-Game-Of-Thrones-by-George-R.R.-Martin.txt",
"Analysing sentences an introduction to English syntax (PDFDrive).txt",
"Business Law- An Introduction (PDFDrive).txt",
"Introduction to algorithms.txt",
"Handbook of Medicinal Herbs (PDFDrive).txt",
"(Book 3) Harry Potter And The Prisoner Of Azkaban_001.txt",
"bese.txt",
"How To Win Friends and Influence People (PDFDrive).txt",
"ips6e.ex.st.txt",
"Probability and Computing Randomization and Probabilistic Techniques in Algorithms and Data Analysis.txt",
"Never Split the Difference Negotiating As If Your Life Depended On It (PDFDrive).txt",
"Grammar of the English Verb Phrase, Volume 1 The Grammar of the English Tense System A Comprehensive Analysis (PDFDrive).txt",

```
          "The Horse and His Boy (The Chronicles of Narnia, Book 3) ( PDFDrive ).txt",
          "The Complete Home Guide to Herbs, Natural Healing, and Nutrition ( PDFDrive ).txt",
          "Joel Fuhrman - Eat To Live ( PDFDrive ).txt",
          "The Chronicles of Narnia 1 - The Magicians Nephew ( PDFDrive ).txt",
          "English Grammar Reference Book_ - Jacqueline Melvin.txt",
          "International Law and International Relations ( PDFDrive ).txt",
          "Java Java Programming For Beginners - A Simple Start to Java Programming ( PDFDrive
          ).txt",
          "Lord of the Rings - The Fellowship of the ring ( PDFDrive ).txt",
          "Law of Success (21st Century Edition) ( PDFDrive ).txt",
          "English_Sentence_Analysis.txt",
          "Salt, Fat, Acid, Heat Mastering the Elements of Good Cooking ( PDFDrive ).txt",
          "Practical Common Lisp [Seibel 2005-04-11].txt"]

classlibros=[0,1,2,3,4,0,1,2,3,4,4,3,2,1,0,4,3,2,1,0,0,1,2,3,4,4,3,2,1,0,4,3,2,1,0,0,1,2,3,4,4,3,2
,1,0,3,1,2,4,0]
tl=len(classlibros)

testlib= ["Remote Sensing Digital Image Analysis.txt",
          "Time Management Proven Techniques for Making Every Minute Count ( PDFDrive ).txt",
          "Webster's Word Power Better English Grammar. Improve Your Written and Spoken
          English ( PDFDrive ).txt",
          "The Silver Chair (Narnia) ( PDFDrive ).txt",
          "Your Body is Your Gym Use Your Bodyweight to Build Muscle and Lose Fat With the
          Ultimate Guide to Bodyweight Training ( PDFDrive ).txt",
          "clash of kings.txt",
          "Stock investing for Dummies.pdf ( PDFDrive ).txt",
          "The Grammar of English Grammars ( PDFDrive ).txt",
          "The Return of the King ( PDFDrive ).txt",
          "PROLOG.txt",
          "First Aid for the Emergency Medicine Board ( PDFDrive ).txt",
          "Book2-Harry-Potter-and-the-Chamber-of-Secrets.txt",
          "Basic English Grammar For English Language Learners (Basic English Grammar for
          English Language Learners) ( PDFDrive ).txt",
          "How Successful People Think Change Your Thinking, Change Your Life ( PDFDrive ).txt",
          "Introduction to the Design and Analysis of Algorithms.txt",
          "How to Cook Everything, Completely Revised 10th Anniversary Edition 2,000 Simple
          Recipes for Great Food ( PDFDrive ).txt",
          "how to write a business plan.txt",
          "Complete English Grammar Rules Examples, Exceptions, Exercises, and Everything
          You Need to Master Proper Grammar ( PDFDrive ).txt",
          "The Science of Cooking Understanding the Biology and Chemistry Behind Food and
          Cooking ( PDFDrive ).txt",
          "Java for Absolute Beginners Learn to Program the Fundamentals the Java 9+ Way
          ( PDFDrive ).txt",
          "Pattern Recognition and Machine Learning .txt",
          "International Political Economy ( PDFDrive ).txt",
          "EnglishGrammarMasterin30Days.txt",
          "harry_potter_annd_the_sorcerers_stone.txt",
          "On Food and Cooking The Science and Lore of the Kitchen ( PDFDrive ).txt"]

classtest=[0,1,2,3,4,3,1,2,3,0,4,3,2,1,0,4,1,2,4,0,0,1,2,3,4]
tt=len(classtest)

#--------CREANDO TRAINSET----------
trainset=np.array(())
for i in range(tl):
          libro=libros[i]
          lib=open(libro)
          aux=np.array((lib.read()))
          trainset=np.append(trainset,aux)
          lib.close()

print(np.shape(classlibros))
print(np.shape(classtest))

vect=CountVectorizer() #inicio de conteo
vect.fit(trainset) #entrenamiento
vect.get_feature_names() #se obtiene el vocabulario
matrixtrain=vect.transform(trainset) #crea la matriz
datatrain=pd.DataFrame(matrixtrain.toarray(),columns=vect.get_feature_names())
#transforma a dataframe
#print(datatrain)

#----------CREANDO TESTSET-----------
testset=np.array(())
for i in range(tt):
          libro=libros[i]
          lib=open(libro)
          aux=np.array((lib.read()))
          testset=np.append(testset,aux)
          lib.close()

matrixtest=vect.transform(testset)
datatest=pd.DataFrame(matrixtest.toarray(), columns=vect.get_feature_names())
#print(datatest)
```

```
#------EVALUANDO--------
nb=MultinomialNB()
nb.fit(matrixtrain,classlibros)

classpred=nb.predict(matrixtest)
classpredprob=nb.predict_proba(matrixtest)


print("La matriz de confusion es:")
confmatrix=metrics.confusion_matrix(classtest,classpred)
print(confmatrix)

print("El accuracy es:")
porcentaje=metrics.accuracy_score(classtest,classpred)
print(porcentaje)

recall0=confmatrix[0][0]/(confmatrix[0][0]+confmatrix[0][1]+confmatrix[0][2]+confmatrix[0]
[3]+confmatrix[0][4])
recall1=confmatrix[1][1]/(confmatrix[1][0]+confmatrix[1][1]+confmatrix[1][2]+confmatrix[1]
[3]+confmatrix[1][4])
recall2=confmatrix[2][2]/(confmatrix[2][0]+confmatrix[2][1]+confmatrix[2][2]+confmatrix[2]
[3]+confmatrix[2][4])
recall3=confmatrix[3][3]/(confmatrix[3][0]+confmatrix[3][1]+confmatrix[3][2]+confmatrix[3]
[3]+confmatrix[3][4])
recall4=confmatrix[4][4]/(confmatrix[4][0]+confmatrix[4][1]+confmatrix[4][2]+confmatrix[4]
[3]+confmatrix[4][4])
mar=(recall0+recall1+recall2+recall3+recall4)/5

print("Recalls:")
print(recall0)
print(recall1)
print(recall2)
print(recall3)
print(recall4)
print("El Macro Average Recall es:")
print(mar)

precision0=confmatrix[0][0]/(confmatrix[0][0]+confmatrix[1][0]+confmatrix[2]
[0]+confmatrix[3][0]+confmatrix[4][0])
precision1=confmatrix[1][1]/(confmatrix[0][1]+confmatrix[1][1]+confmatrix[2]
[1]+confmatrix[3][1]+confmatrix[4][1])
```

```
precision2=confmatrix[2][2]/(confmatrix[0][2]+confmatrix[1][2]+confmatrix[2]
[2]+confmatrix[3][2]+confmatrix[4][2])
precision3=confmatrix[3][3]/(confmatrix[0][3]+confmatrix[1][3]+confmatrix[2]
[3]+confmatrix[3][3]+confmatrix[4][3])
precision4=confmatrix[4][4]/(confmatrix[0][4]+confmatrix[1][4]+confmatrix[2]
[4]+confmatrix[3][4]+confmatrix[4][4])
maap=(precision0+precision1+precision2+precision3+precision4)/5

print("Precisiones:")
print(precision0)
print(precision1)
print(precision2)
print(precision3)
print(precision4)
print("El Macro Average Precision es:")
print(maap)

f10=2*(precision0*recall0)/(precision0+recall0)
f11=2*(precision1*recall1)/(precision1+recall1)
f12=2*(precision2*recall2)/(precision2+recall2)
f13=2*(precision3*recall3)/(precision3+recall3)
f14=2*(precision4*recall4)/(precision4+recall4)

print("F1-scores:")
print(f10)
print(f11)
print(f12)
print(f13)
print(f14)

m0=np.array([
[confmatrix[0][0], (confmatrix[1][0]+confmatrix[2][0]+confmatrix[3][0]+confmatrix[4][0])],
[(confmatrix[0][1]+confmatrix[0][2]+confmatrix[0][3]+confmatrix[0][4]), (tt-confmatrix[0][0]-
(confmatrix[0][1]+confmatrix[0][2]+confmatrix[0][3]+confmatrix[0][4])-(confmatrix[1]
[0]+confmatrix[2][0]+confmatrix[3][0]+confmatrix[4][0]))]
])

m1=np.array([
[confmatrix[1][1], (confmatrix[0][1]+confmatrix[2][1]+confmatrix[3][1]+confmatrix[4][1])],
[(confmatrix[1][0]+confmatrix[1][2]+confmatrix[1][3]+confmatrix[1][4]), (tt-confmatrix[1][1]-
(confmatrix[0][1]+confmatrix[2][1]+confmatrix[3][1]+confmatrix[4][1])-(confmatrix[1]
[0]+confmatrix[1][2]+confmatrix[1][3]+confmatrix[1][4]))]
])
```

```
m2=np.array([
[confmatrix[2][2], (confmatrix[0][2]+confmatrix[1][2]+confmatrix[3][2]+confmatrix[4][2])],
[(confmatrix[2][0]+confmatrix[2][1]+confmatrix[2][3]+confmatrix[2][4]), (tt-confmatrix[2][2]-
(confmatrix[0][2]+confmatrix[1][2]+confmatrix[3][2]+confmatrix[4][2])-(confmatrix[2]
[0]+confmatrix[2][1]+confmatrix[2][3]+confmatrix[2][4]))]
])

m3=np.array([
[confmatrix[3][3], (confmatrix[0][3]+confmatrix[1][3]+confmatrix[2][3]+confmatrix[4][3])],
[(confmatrix[3][0]+confmatrix[3][1]+confmatrix[3][2]+confmatrix[3][4]), (tt-confmatrix[3][3]-
(confmatrix[0][3]+confmatrix[1][3]+confmatrix[2][3]+confmatrix[4][3])-(confmatrix[3]
[0]+confmatrix[3][1]+confmatrix[3][2]+confmatrix[3][4]))]
])

m4=np.array([
[confmatrix[4][4], (confmatrix[0][4]+confmatrix[1][4]+confmatrix[2][4]+confmatrix[3][4])],
[(confmatrix[4][0]+confmatrix[4][1]+confmatrix[4][2]+confmatrix[4][3]), (tt-confmatrix[4][4]-
(confmatrix[0][4]+confmatrix[1][4]+confmatrix[2][4]+confmatrix[3][4])-(confmatrix[4]
[0]+confmatrix[4][1]+confmatrix[4][2]+confmatrix[4][3]))]
])

print("Matrice de confusion de cada clase:")
print(m0)
print(m1)
print(m2)
print(m3)
print(m4)
```

Los resultados obtenidos son los siguientes:

La matriz de confusión es:

```
[[4 0 0 0 1]
 [0 4 0 1 0]
 [0 0 5 0 0]
 [1 0 0 4 0]
 [0 1 0 0 4]]
```

El Accuracy es:
0.84

```
Recalls:
R0=0.8
R1=0.8
R2=1.0
R3=0.8
R4=0.8

El Macro Average Recall es:
0.8400000000000001

Precisiones:
P0=0.8
P1=0.8
P2=1.0
P3=0.8
P4=0.8

El Macro Average Precisión es:
0.8400000000000001

F1-scores:
F0=0.8000000000000002
F1=0.8000000000000002
F2=1.0
F3=0.8000000000000002
F4=0.8000000000000002

Matrices de confusión de cada clase:

MC0=[[ 4  1]
     [ 1 19]]

MC1=[[ 4  1]
     [ 1 19]]

MC2=[[ 5  0]
     [ 0 20]]
```

```
MC3=[[ 4  1]
     [ 1 19]]

MC4=[[ 4  1]
     [ 1 19]]
```

Como se puede ver en los resultados el Accuracy obtenido es de 84%, ya que la mayoría de las clases clasificaron erróneamente un libro, excepto por la clase 2, Education, que clasifico de manera correcta sus cinco libros de test, las demás clases aunque si clasificaron en total cinco libros para cada una, uno de esos libros pertenecía a una clase diferente, es por eso que sus Recall, precisión y F1-score son de 0.8, esto es mas notorio en las matrices de confusión de cada una de las clases donde se aprecia que tienen un falso positivo y un falso negativo respectivamente, pero en general se obtiene un porcentaje aceptable de clasificación, aunque posiblemente podría ser mejorado si se le agregan mas ejemplos de entrenamiento. Por otra parte gracias a la librería de scikitlearn el programa se ejecuta de manera rápida, tardando menos de un minuto en ejecutarse y obtener los resultados esperados, por lo que es una herramienta muy útil, que al estar tan bien optimizada para python es posible realizar clasificaciones muy rápidas.