Proyecto Final

|  |  |
| --- | --- |
| Algoritmos de aprendizaje automático para tareas de Clasificación y Análisis Inteligente de Datos. | Fecha: 30/11/2022 |
| Alumno: Victor Alberto Lopez Cardona | Exp: 747175 |

## Rúbrica.

Entregar  en un archivo zip o rar en la actividad en CANVAS:

1. código fuente

2. reporte de proyecto

Rúbrica:

40 puntos menos -  Si no anexan el código fuente

40 puntos menos -  Si no anexan reporte de proyecto

20 puntos menos -  Si esta incompleto su reporte, recuerden poner referencias y conclusiones.

40 puntos menos -  Si  la simulación  es incorrecta

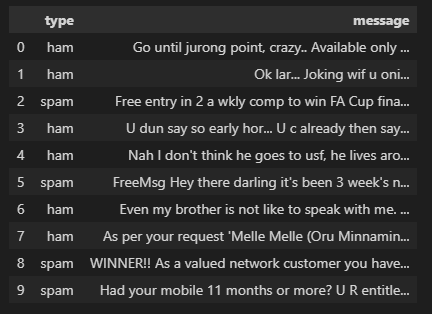
## Objetivo general.

Programar alguno de los siguientes algoritmos.

* COLOREADO DE GRAFOS EMPLEANDO EL METODO DE ABSORCIÓN
* COLOREADO DE GRAFOS EMPLEANDO ALGORITMOS GENETICOS
* ALGORITMO DE HUFFMAN (Aplicando Árboles binarios alfabéticos óptimos)
* **CLASIFICADOR DE TEXTOS USANDO REDES BAYESIANAS INGENUAS**
* ARBOLES DE DECISIÓN MEDIANTE ID3, REPTREE Ó J48

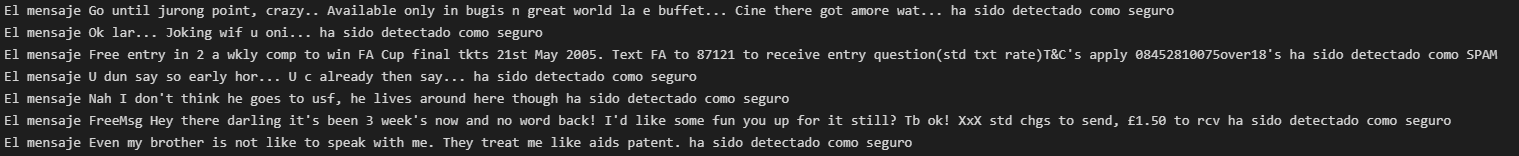
## NAIVE BAYES CLASIFICADOR DE TEXTOS :

Entrada : Texto a clasificar. (Spam\_data)



Salida : Modelo Clasificador.

* Clasificador de clase, aquella con la probabilidad posterior más alta como resultado de la predicción.





Algoritmo:

* Tabla de frecuencias del conjunto de Datos
* Tabla de probabilidad calculando las correspondientes a que ocurran los diversos eventos.
* Aplicación de la ecuación Naive Bayes para calcular la probabilidad posterior de cada clase.

## Codigo.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import string

from nltk.corpus import stopwords

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfTransformer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.metrics import accuracy\_score

import nltk

try:

    nltk.corpus.stopwords.words('english')

except:

    nltk.download('stopwords')

#remover letras que apocan poca info

def clean\_data(mnsj):

    msg = [x for x in mnsj if x not in string.punctuation]

    msg = ''.join(msg)

    clean\_msg = [x for x in msg.split() if x.lower not in nltk.corpus.stopwords.words('english')]

    return clean\_msg

#leer informacion

all\_data = pd.read\_csv('Spam\_data',sep='\t',names=["type","message"])

all\_data.head(10)

all\_data.groupby('type').describe()

#determinar atributos que nos puedan indicar si es SPAM o no

all\_data['length']=all\_data['message'].apply(len)

all\_data.length.describe()

#transformar la informacion a vectores

vector = CountVectorizer(analyzer=clean\_data).fit(all\_data['message'])

df\_vector = vector.transform(all\_data['message'])

frecuencias=TfidfTransformer().fit(df\_vector)

#obtener una tabla de frecuencias

frecuencias\_data\_frame=frecuencias.transform(df\_vector)

print(frecuencias\_data\_frame.shape)

#entrenar el modelo

modelo = MultinomialNB().fit(frecuencias\_data\_frame,all\_data['type'])

#predecir todos los datos con el modelo

resultados = modelo.predict(frecuencias\_data\_frame)

print(resultados)

#imprimir los primeros mensajes y obtener si es seguro o SPAM

for i in range(0,50):

    if resultados[i] == "ham":

        print(f"El mensaje {all\_data['message'][i]} ha sido detectado como seguro")

    elif resultados[i] == "spam":

        print(f"El mensaje {all\_data['message'][i]} ha sido detectado como SPAM")

#calcular la precision

puntaje = accuracy\_score(all\_data['type'], resultados)

print(f'Precision: {puntaje\*100}')

## Conclusiones.

El objetivo del proyecto es hacer uso de redes bayesianas ingenuas para clasificar un mensaje como SPAM o como mensaje seguro, para hacer esto podemos hacer uso de 3 distintos algoritmos, que son el Multinomial, Gaussiano, y de Bernoulli, para este proyecto se opto por utilizar el algoritmo multinomial ya que para nuestro set de información que es un texto, este algoritmo funciona mejor.

El proyecto logra clasificar los mensajes como SPAM o mensaje seguro con un 97% de precisión aproximadamente.

Como áreas de mejora o trabajo futuro, podemos normalizar la información que tenemos, hay algunos mensajes que contienen una palabra mal escrita y podemos detectar esta palabra y corregirla para tener una tabla de frecuencias correcta en nuestro proyecto, esto nos llevara a un modelo de predicción mas preciso.

En el código podemos observar que tenemos una función clean\_data(), en esta función estamos eliminando palabras que no nos aportan información valiosa al mensaje, por lo que las eliminamos por cada uno de los mensajes que tenemos, esto mejoro la precisión de nuestro modelo en un 30%, ya que la mayoría de las palabras que se repetían en los mensajes eran “the”, “a”, is“, y el modelo identificaba erróneamente los mensajes que contenían SPAM.

## Bibliografía.

<https://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/MGP_RepProy_Abr_29.pdf>

<https://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note07-2up.pdf>

<https://www.geeksforgeeks.org/naive-bayes-classifiers/>