

# Proyecto Final

DOCENTE	CARRERA	CURSO
Vicente Machaca Arceda	Maestría en Ciencia de la Computación	Algoritmos y Estructura de Datos

PRÁCTICA	TEMA	DURACIÓN
Final	Algoritmos de ordenamiento	3 horas

## 1. Datos de los estudiantes

Grupo: N° 8

■ Integrantes:

- Esai Josue Huaman Meza
- Alan Jerry Reyes Robles
- Jorge Luis Zegarra Guardamino
- Nestor Giraldo Calcinas Huaranga

## 2. Introducción

Este proyecto final trata de la Clasificación de correos SPAM.

Para esto se usará una parte del código utilizado en la práctica 04, para luego implementar un descriptor que para este caso será una bolsa de palabras.

Para todo ello, el lenguaje de programación a usar será Java Script y Python.

El proyecto se encuentra en el siguiente enlace: Clasificador de Correos SPAM.

El repositorio Github se encuentra en el siguiente enlace: Clasificador SPAM.

El video explicativo se encuentra en el siguiente enlace: Clasificador de Correos SPAM - Grupo 8.

El Descriptor hecho en Google Colab se encuentra en el siguiente enlace: Descriptor.py.

## 3. SPAM

### 1. ¿Qué es el SPAM?

El spam es cualquier forma de comunicación no solicitada que se envía de forma masiva (correo electrónico masivo no solicitado, o UBE). Su forma más frecuente es un correo electrónico de publicidad enviado a un gran número de direcciones (correo electrónico de publicidad no solicitado, o UCE), pero el "spamming" también existe a través de mensajes instantáneos, de texto (SMS), redes sociales o incluso mensajes de voz. Enviar spam es ilegal en la mayoría de jurisdicciones.

Una de las vías más comunes para propagar contenido no solicitado es a través de botnets, grandes redes de dispositivos "zombieinfectados". A veces los correos en cadena y fraudulentos también se consideran spam, aunque difieren en que en general se reenvían por gente con buenas intenciones.



Figura 1: Correos SPAM

## 2. Tipos de SPAM

- SPAM por correos electrónicos
- SPAMming en buscadores
- SPAM telefónico
- SPAM en redes sociales
- SPAM a través de mensajería instantánea

## 3. Origen de los Datos

- App store de Android
- Troyanos en Smartphones
- Redes sociales
- Fugas de datos

## 4. Impacto del SPAM

El SPAM está causando graves daños en el sistema mundial de comunicaciones. Los datos adicionales resultan en un esfuerzo de procesamiento considerablemente mayor. Además, la lectura y clasificación del spam es costosa, hay que comprar y mantener filtros de spam y se generan costes por cada byte transferido de spam, ya que los proveedores de servicios de Internet suelen cobrar por sus servicios en función del volumen de datos transferidos.

# 4. Marco Teórico

## 1. Medidas para evitar el correo basura (SPAM)

A pesar de que no existen técnicas infalibles para protegerse del correo basura, los expertos en seguridad informática recomiendan una serie de acciones para reducir la cantidad de correo electrónico no deseado:

- Usar una imagen para la dirección de correo electrónico.
- En vez de poner el enlace a tu cuenta, usa una redirección (puede ser temporal o por un número de usos), y bórrala cuando recibas excesivos mensajes no deseados.
- Modificar la dirección para evitar el rastreo automático.

En los grupos de noticias y listas de correo:

- No poner el remitente verdadero en las publicaciones enviados.
- Si el archivo de mensajes a la lista es visible desde web, cambiar las direcciones de remite por una imagen, ocultarlas, o escribirlas de forma que sea difícil reconocerla como tal para un programa.

## 2. Proyectos y servicios contra el correo basura (SPAM)

- SPF: tecnología creada para verificar que los remitentes de los mensajes de correo son quienes dicen ser.
- DomainKeys: otra tecnología que sirve para lo mismo que SPF y que además asegura que los mensajes de correo electrónico no han sido modificados.
- SenderID: tecnología de Microsoft que pretende competir con SPF, incluso utilizando esas mismas siglas para una tecnología que en la práctica es distinta. En realidad, SenderID utiliza consultas DNS parecidas a SPF sólo como primer paso de su proceso, que involucra también filtros antispam basados en contenido. SenderID ha sido adoptado por Hotmail. En la práctica esto obliga a adoptar esta tecnología o verse excluido de esas direcciones, que suponen unos 260 millones de usuarios en todo el mundo. No obstante, los resultados de su tecnología, y/o otras medidas paralelas adoptadas, están causando serios problemas en dominios enteros en todo el mundo.
- Proyectos como el proyecto Harvester y el emailharvest recopilan las IP de generadores de basura para bloquearlas mediante una trampa. Ponen direcciones de correo que indican la dirección del remitente de correo masivo y cuando él envía un mensaje a esa dirección se sabe desde qué dirección fue capturada, con lo que puede filtrar al remitente.
- Redirecciones temporales.

## 5. Clasificador de Correos Electrónicos SPAM

### 1. Utilizando KNN

El algoritmo k-nearest neighbor (KNN) es un método no paramétrico utilizado para clasificación y regresión. En lo que concierne a la clasificación, la idea básica es que, dado un objeto del cual no conocemos su etiqueta de clase, a este se le asignará la etiqueta más común entre sus vecinos más cercanos de acuerdo a una métrica de distancia.

Para este proyecto usaremos el KNN implementado en una práctica anterior.

### 2. Utilizando una Bolsa de Palabras

Es un método que se utiliza en el procesamiento del lenguaje para representar documentos ignorando el orden de las palabras. En este modelo, cada documento parece una bolsa que contiene algunas palabras. Por lo tanto, este método permite un modelado de las palabras basado en diccionarios, donde cada bolsa contiene unas cuantas palabras del diccionario. En el campo de reconocimiento de objetos, se utiliza una idea similar para las representaciones de imágenes, es decir, una imagen puede ser tratada como un documento y las características extraídas de ciertos puntos de la imagen son consideradas palabras visuales. Las principales ventajas de utilizar este modelo es su facilidad de uso y su eficiencia computacional.

### 3. Utilizando Descriptor

Los descriptores de Python se crean para gestionar los atributos de diferentes clases que utilizan el objeto como referencia. En descriptores se utilizaron tres métodos diferentes que son getters, setters y delete. Si alguno de esos métodos está definido para un objeto, puede denominarse

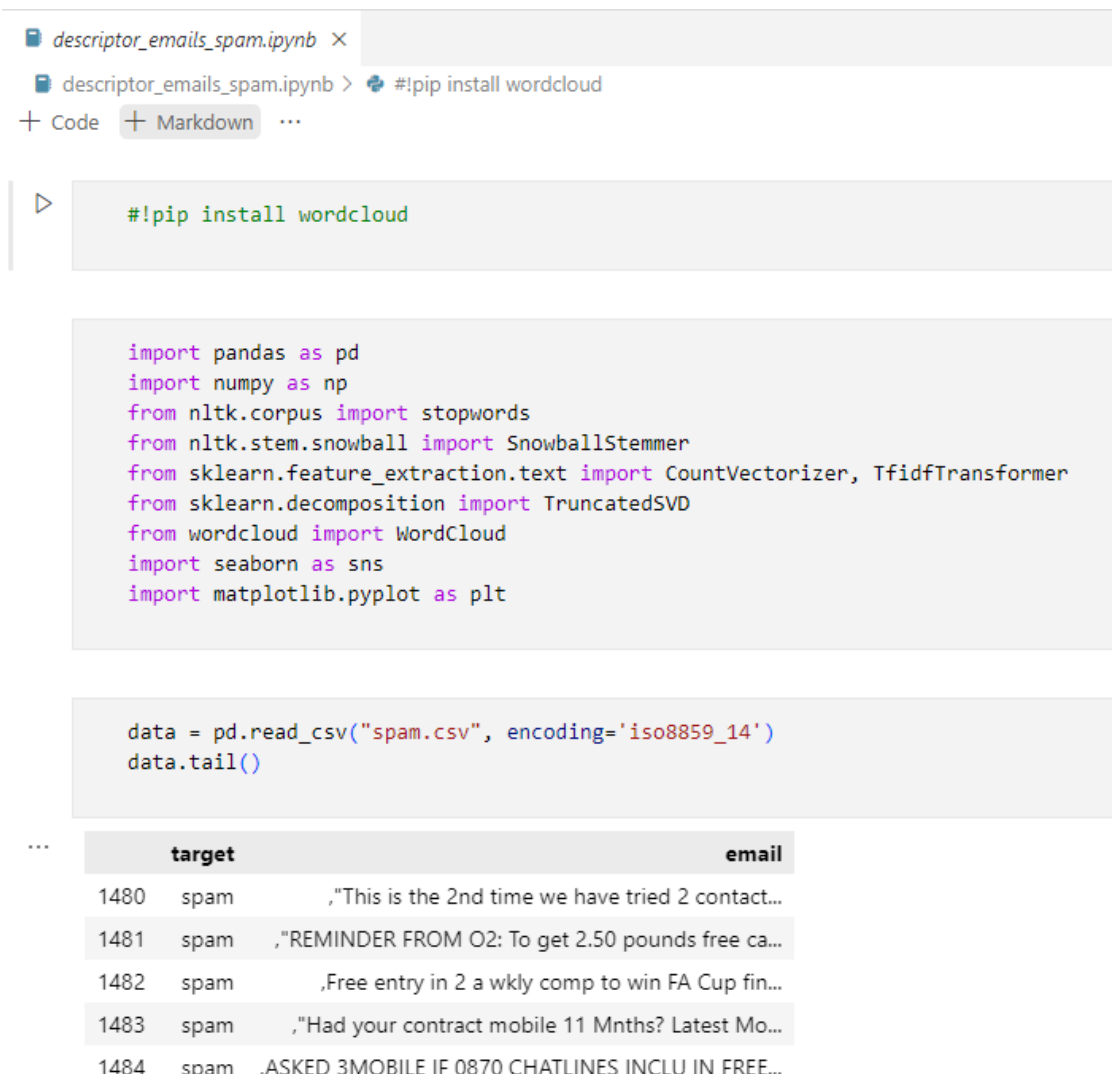
descriptor. Normalmente, Python usa métodos como getters y setters para ajustar los valores de los atributos sin ningún procesamiento especial. Es solo un sistema de almacenamiento básico. A veces, es posible que deba validar los valores que se asignan a un valor. Un descriptor es un mecanismo detrás de propiedades, métodos, métodos estáticos, métodos de clase y super.

## 6. Implementación

Este proyecto se desarrolla utilizando los lenguajes Python y Java Script, los cuales se pueden encontrar en el siguiente repositorio Github Clasificador SPAM., y se obtienen las siguientes imágenes.

## 7. Resultados

### 1. Descriptor



```
descriptor_emails_spam.ipynb ×
descriptor_emails_spam.ipynb > #!pip install wordcloud
+ Code + Markdown ...

#!pip install wordcloud


import pandas as pd
import numpy as np
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.decomposition import TruncatedSVD
from wordcloud import WordCloud
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv("spam.csv", encoding='iso8859_14')
data.tail()
```

	target	email
1480	spam	, "This is the 2nd time we have tried 2 contact...
1481	spam	, "REMINDER FROM O2: To get 2.50 pounds free ca...
1482	spam	, Free entry in 2 a wkly comp to win FA Cup fin...
1483	spam	, "Had your contract mobile 11 Mnths? Latest Mo...
1484	spam	, ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE...

Figura 2: Descriptor

## 2. Data de correos SPAM

 data2vector.csv X data2vector.csv

```
1 ;Var1;Var2;target;;;;;;;;;
2 0;117;52;true;,{Var1:117,Var2:52,spam:true};;;458;0;var1;0;500
3 1;31;21;true;,{Var1:31,Var2:21,spam:true};;;439;-436;var2;-500;500
4 2;107;-116;true;,{Var1:107,Var2:-116,spam:true};;;;;;;;;;
5 3;277;-223;true;,{Var1:277,Var2:-223,spam:true};;;;;;;;;;
6 4;248;199;true;,{Var1:248,Var2:199,spam:true};;;;;;;;;;
7 5;120;-20;true;,{Var1:120,Var2:-20,spam:true};;;;;;;;;;
8 6;152;104;true;,{Var1:152,Var2:104,spam:true};;;;;;;;;;
9 7;260;1;true;,{Var1:260,Var2:1,spam:true};;;;;;;;;;
10 8;105;-114;true;,{Var1:105,Var2:-114,spam:true};;;;;;;;;;
11 9;73;85;true;,{Var1:73,Var2:85,spam:true};;;;;;;;;;
12 10;43;47;true;,{Var1:43,Var2:47,spam:true};;;;;;;;;;
13 11;156;94;true;,{Var1:156,Var2:94,spam:true};;;;;;;;;;
14 12;132;14;true;,{Var1:132,Var2:14,spam:true};;;;;;;;;;
15 13;359;-347;true;,{Var1:359,Var2:-347,spam:true};;;;;;;;;;
16 14;60;62;true;,{Var1:60,Var2:62,spam:true};;;;;;;;;;
17 15;253;-214;true;,{Var1:253,Var2:-214,spam:true};;;;;;;;;;
18 16;150;86;true;,{Var1:150,Var2:86,spam:true};;;;;;;;;;
19 17;15;17;true;,{Var1:15,Var2:17,spam:true};;;;;;;;;;
20 18;155;33;true;,{Var1:155,Var2:33,spam:true};;;;;;;;;;
21 19;286;193;true;,{Var1:286,Var2:193,spam:true};;;;;;;;;;
22 20;21;6;true;,{Var1:21,Var2:6,spam:true};;;;;;;;;;
23 21;28;28;true;,{Var1:28,Var2:28,spam:true};;;;;;;;;;
24 22;277;-78;true;,{Var1:277,Var2:-78,spam:true};;;;;;;;;;
25 23;99;-22;true;,{Var1:99,Var2:-22,spam:true};;;;;;;;;;
26 24;107;15;true;,{Var1:107,Var2:15,spam:true};;;;;;;;;;
27 25;196;148;true;,{Var1:196,Var2:148,spam:true};;;;;;;;;;
28 26;41;19;true;,{Var1:41,Var2:19,spam:true};;;;;;;;;;
29 27;94;57;true;,{Var1:94,Var2:57,spam:true};;;;;;;;;;
30 28;122;-59;true;,{Var1:122,Var2:-59,spam:true};;;;;;;;;;
31 29;164;60;true;,{Var1:164,Var2:60,spam:true};;;;;;;;;;
32 30;168;-66;true;,{Var1:168,Var2:-66,spam:true};;;;;;;;;;
```

Figura 3: Data Generado por el Descriptor (1476 datos)

### 3. KNN

```
JS script.js ×
JS script.js > KNN_c > constructor
30
31     for (let k = 1; k <= 5; k++) {
32         const minDistance = Math.min.apply(Math, distances)
33         const index = distances.indexOf(minDistance)
34         chart.data.datasets[0].pointBackgroundColor[index] == 'red' ? redNeighbors++ : blueNeighbors++
35         distances[index] = +Infinity
36     }
37     if (redNeighbors > blueNeighbors) {
38         document.querySelector('#output').innerHTML = 'Correo Spam'
39         chart.data.datasets[0].pointBackgroundColor[chart.data.datasets[0].data.length - 1] = 'red'
40     } else {
41         document.querySelector('#output').innerHTML = 'Correo No Spam'
42         chart.data.datasets[0].pointBackgroundColor[chart.data.datasets[0].data.length - 1] = 'blue'
43     }
44     chart.update()
45 }
46
47 class KNN_c {
48
49     constructor(k = 1, data, labels) {
50         this.k = k;
51         this.data = data;
52         this.labels = labels;
53     }
54
55     generateDistanceMap(point) {
56
57         const map = [];
58         let maxDistanceInMap;
59
60         for (let index = 0, len = this.data.length; index < len; index++) {
61
```

Figura 4: KNN utilizado

### 4. Index.html

```
<> index.html ×
<> index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3    <head background=#000000;>
4      <link rel="stylesheet" href="style.css">
5      <script src="https://cdn.jsdelivr.net/npm/chart.js/2.4.0/Chart.min.js"></script>
6      <script src="data.js"></script>
7      <script src="config.js"></script>
8
9      <meta charset="UTF-8"/>
10     <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
11
12     <link
13       | rel="stylesheet"
14       | href="https://use.fontawesome.com/releases/v5.15.1/css/all.css"
15       | integrity="sha384-vp86vTRFVJgpjF9jiIGPEEqYqlDwgyBgEF109VFjmqGmIY/Y4HV4d3Gp2irVfcrp"
16     />
17     <link rel="stylesheet" href="./styles/styles.css"/>
18     <title>KNN con JS</title>
19
20   </head>
21   <body>
22     <!-- Navbar Section -->
23     <nav class="navbar">
24       <a class="navbar__logo">KNN con JS - <i>Grupo 8</i></a>
25       <div class="navbar__bars"><i class="fas fa-bars"></i></div>
26       <div class="navbar__menu">
27         <a class="navbar__menu--links">Algoritmos y Estructura de Datos (AyED) - UNSA</a>
28       </div>
29     </nav>
30     <p style="padding-top: 30px; padding-right: 90px; padding-bottom: 0px; padding-left: 110px">El algoritmo de
31       más cercanas o K-nearest neighbors (knn) es un algoritmo de Machine Learning que pertenece a los algor
32       aprendizaje supervisado simples y fáciles de aplicar que pueden ser utilizados para resolver problemas
```

Figura 5: Index.html

## 5. Visualización HTML

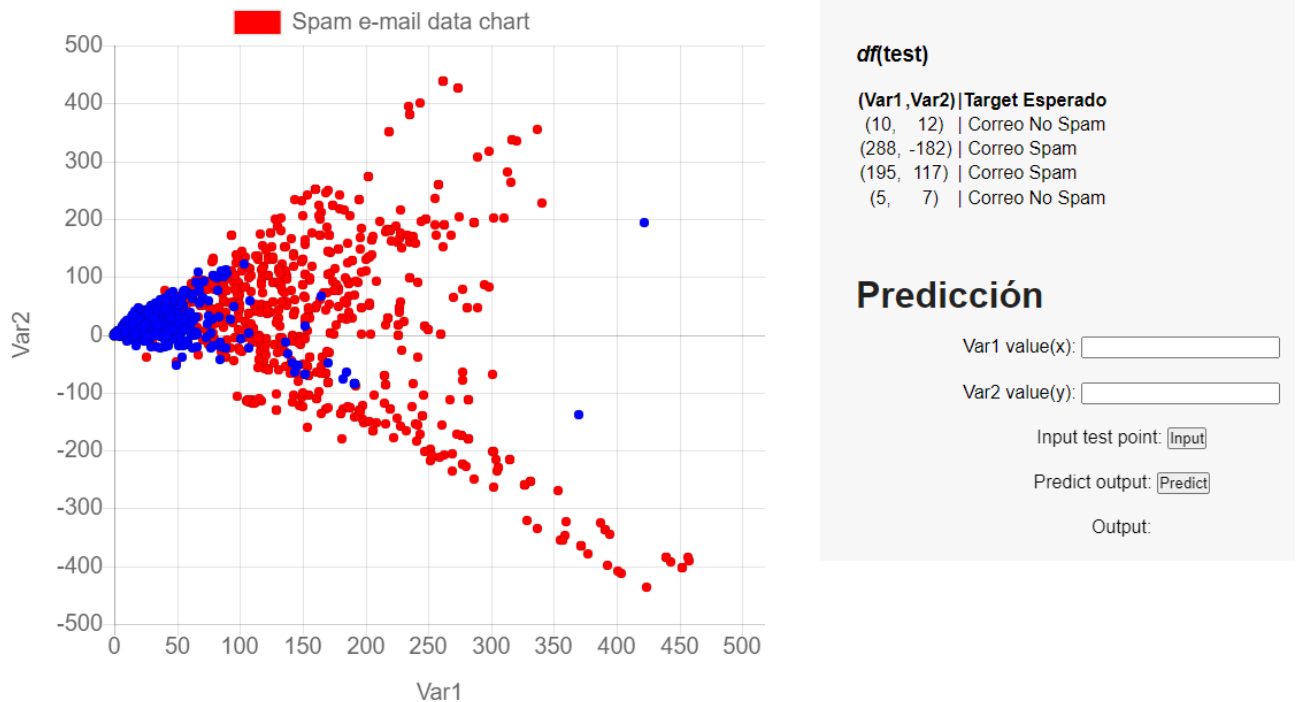


Figura 6: Visualización de pantalla

## 8. Conclusiones

- El filtrado de correo no deseado es un tema importante en seguridad de red y actualmente las técnicas de aprendizaje automático brindan una alternativa muy interesante y eficiente para resolver este tipo de problemas.
- El Knn es un algoritmo de aprendizaje supervisado, a partir de un juego de datos inicial, su objetivo será el de clasificar correctamente todas las instancias nuevas. El juego de datos típico de este tipo de algoritmos está formado por varios atributos descriptivos y un solo atributo objetivo.
- El Knn es sensible a la variable k, que con valores distintos de k podemos obtener resultados también muy distintos sobretodo en los bordes de la clasificación. La métrica de similitud utilizada (el descriptor a usar), puesto que esta influirá, fuertemente, en las relaciones de cercanía que se irán estableciendo en el proceso de construcción del algoritmo.