Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Análisis del comportamiento de diferentes algoritmos de aprendizaje automático

Modalidad: Proyecto Tecnológico

Trimestre 2018 Invierno

Stephany Guadalupe Anaya García
2112043911
stephany.anaya@hotmail.com

Asesor:
José Alejandro Reyes Ortiz
Doctor en Ciencias de la Computación
Profesor Asociado
Departamento de Sistemas
jaro@correo.azc.uam.mx

Coasesor:
Angeles Belém Priego Sánchez
Doctora en Ciencias del Lenguaje
Profesor Titular
Departamento de Sistemas
abps@correo.azc.uam.mx

24 de abril del 2018

Declaratoria

Yo, José Alejandro Reyes Ortiz, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Angeles Belém Priego Sánchez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Yo, Stephany Guadalupe Anaya García, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Resumen

Actualmente nos encontramos en la era de la información en donde se produce, se difunde y se emplea gran cantidad de ésta diariamente, la cual puede ser estudiada y analizada con el fin de obtener nueva información. Aunado a este hecho se encuentran las redes sociales en las cuales se puede expresar una idea, concepto, opinión o evento. El análisis de las ideas, comentarios de Twitter, por ejemplo, escritas por la comunidad ha despertado el interés de muchos investigadores.

Para la tarea del análisis de datos se utilizan algoritmos de aprendizaje automático, donde se emplean métodos que contribuyen a que las computadoras se instruyan con los datos proporcionados. La implementación de los algoritmos de aprendizaje automático es basta, teniendo como ejemplo, el procesamiento del lenguaje natural, diagnosis, clasificación de sucesiones de ADN, etc.

El enfoque de este proyecto es el análisis e investigación del comportamiento de los algoritmos de clasificación *K-Means*, *K-Nearest Neighbors* y *Naïve Bayes*, con el objetivo de catalogar delitos en la zona metropolitana, comparando su exactitud y precisión. Tras la ejecución de cada uno de los algoritmos,los resultados obtenidos evidencian que si se tienen los parámetros correctos se puede clasificar y obtener nueva información a partir de una ya existente.

Los algoritmos son proyectados en el entorno Rstudio, el cual es un IDE de software libre para computación y gráficos estadísticos basado en R. Se observo que, en el caso de la clasificación de delitos ocurridos en el área Metropolitana de la Ciudad de México de acuerdo a las clases objetivo proporcionadas, *Naïve Bayes* presentó resultados destacados, equiparados con los resultados de los algoritmos *K-Nearest Neighbors* y *K-Means*.

Tabla de contenido

Índice de imágenes

Índice de tablas

Introducción

Indudablemente, nos encontramos en la era de la información, en donde se produce, difunde y emplea información en abundantes cantidades diariamente. El estímulo que originó esta era se dio con el auge de las tecnologías de la información y comunicación.

Un campo encargado del estudio de esta información es el Procesamiento del Lenguaje Natural (denotado de aquí en adelante por PLN), gracias a que proporciona un análisis de los textos convenientes para detectar, recuperar y computar información subjetiva de forma metódica. Dentro del PLN encontramos el área de la Minería de Textos, la cual, actualmente ha despuntado debido a la creciente demanda en el uso de las redes sociales. Oportunamente, las empresas y otras instituciones se han beneficiado de ello, estudiando y examinando las colosales cantidades de datos que se producen y difunden por los usuarios a través de Internet, esto diferenciando la composición del sentimiento en un texto. La detección de las opiniones expresadas por los usuarios de las redes sociales, brinda la facultad de responder interrogantes en un gran número de dominios de aplicación.

Así mismo, dentro de las redes sociales emergen los denominados *influencer*, los cuales son individuos con características y aptitudes sociales favorables, que contribuyen en la toma de decisiones de un gran número de personas, guiados por las opiniones de estos mismos. Cuando se origina y difunde una opinión pública, en la cual puede estar en actuación diferentes circunstancias, desde el comportamiento en la bolsa de valores de una empresa o bien la toma de decisiones en la política de un país, se analiza el comportamiento de estas opiniones dentro de las redes sociales que brinda la capacidad de informar acerca de sucesos o acontecimientos venideros. Sin embargo, el ordenamiento y estructuración de los textos capturados, resultaría irrealizable para el ser humano, de tal suerte que se cuenta con algoritmos, el propósito de disponer de estos algoritmos es conjeturar la condición objetivo, mediante el análisis del conjunto de datos capturados, es decir etiquetar y organizar con una valoración la información difundida en el texto, el cuál es el trabajo que se llevará a cabo en este proyecto terminal.

Los algoritmos con los cuales se disponen se encuentran los de clasificación y agrupamiento (*clúster*). Los algoritmos de clasificación persiguen predecir la clase objetivo por medio del análisis del conjunto de datos de entrenamiento, por otro lado, los algoritmos de *clúster* aspiran a agrupar los elementos haciendo uso de diferentes medidas de afinidad o similitud.

Es por ello que el aprendizaje automático se apoya de estos algoritmos, que se destinan para evolucionar los métodos de procesamiento y posibilitan a las computadoras a aprender a predecir acontecimientos, en los cuales se involucran distintos métodos matemáticos. Al situar estos métodos a la práctica se instruyen de los modelos de entrenamiento para realizar su predicción.

En este proyecto terminal se realizará una comparativa de tres algoritmos de aprendizaje automático, nos apoyaremos en los algoritmos de clasificación y *clúster*. Se seleccionará, analizará y comparará determinados modelos algorítmicos, con la finalidad de identificar la precisión y exhaustividad de estos algoritmos dentro de un mismo caso de estudio. El caso de estudio en el que se implementarán los algoritmos, será enfocado en clasificar seis tipos de delitos en el área metropolitana de México que de acuerdo con el Instituto Nacional de Estadística y Geografía [1] son los incidentes que más ocurren en México. Estos son reportados por diversas fuentes de información oficiales vía *Twitter*. Los *Tweets* que contienen la información de los delitos fueron etiquetados previamente de acuerdo a

la categoría a la cual pertenecen, las categorías son: Homicidio, Suicidio, Asalto, Violación, Explotación y Secuestro.

Con la elaboración de este proyecto terminal se señalará y demostrará el algoritmo con mayor precisión y exhaustividad dentro del análisis de textos orientado a los *Tweets* que contengan esta información.

Antecedentes

Proyectos terminales

- Algoritmos de aprendizaje automático para el análisis de opiniones a partir de textos en español [3].

  En el proyecto terminal se aplicaron técnicas de PLN e implementaron cuatro diferentes procesos de Minería de Textos. La principal afinidad que se tiene con este proyecto terminal y con el que se propone en esta propuesta, es el uso de diferentes algoritmos como máquinas de soporte vectorial (SVM por sus siglas en inglés), *Naïve Bayes*, J48 y K-vecinos más cercanos para realizar la minería de texto y se compararán los resultados generados, no obstante, la diferencia radica en el uso del software Weka para implementar dichos algoritmos, mientras que en el caso de esta propuesta se implementarán en el software R.

- Comparativa de la clasificación de tumores obtenida por medio de los algoritmos *k-Means*, PAM, AGNES [4].

  En el proyecto terminal se aplicaron tres algoritmos clúster a los genes activos e inactivos, asociados con tumores pediátricos de 60 pacientes para poder determinar con ello que tipo de tumores se presentaba en cada paciente. La similitud con el presente proyecto terminal se tiene en la aplicación de algoritmos de agrupamiento y el uso del software R, sin embargo, los objetivos planteados, resultados y los datos a emplear es donde difiere.

- Sistema para la clasificación de artículos científicos mediante el algoritmo *K-means* utilizando características semánticas [5].

  En este proyecto terminal se diseñó un sistema para la clasificación de artículos científicos en idioma inglés, utilizando el formato PDF y se presentaron los resultados por medio de la tecnología Web Java Servlets. La similitud con este proyecto terminal es la clasificación de un conjunto de datos, utilizando el algoritmo *K-means*, mientras que la diferencia radica en los datos a emplear, así mismo como las herramientas a usar.

Artículos

- *Sentiment Analysis: An Approach to Opinion Mining from Twitter Data Using R*. [6].

  En el artículo se describe un trabajo de estudio sobre la eficiencia del lenguaje R en la minería de opiniones, la similitud de este articulo con el proyecto terminal radica en el uso del lenguaje R así mismo como el uso de una base de datos extraída de *Twitter*, no obstante, difiere en la aplicación de algoritmos de aprendizaje automático.

- *Machine Learning Algorithms for Opinion Mining and Sentiment Classification*. [7].
  En el artículo se aborda el tema de minería de opiniones y análisis de sentimientos la cual es una tarea de procesamiento de lenguaje natural e información que identifica las opiniones de los usuarios explicadas en forma de comentarios positivos, negativos o neutrales y citas subyacentes al texto. La similitud radica al utilizar distintos algoritmos supervisados o basados en datos no obstante en este caso se realiza para el análisis de sentimientos y también considera la precisión de la clasificación del sentimiento.

- *Algorithms for Opinion Mining and Sentiment Analysis: An Overview*. [8].

  El artículo proporciona una pesquisa sobre los desafíos y la visión general de algunos algoritmos de clasificación y agrupación utilizados para el análisis sentimental y de la minería de opiniones. La similitud radica en la intensión de los algoritmos, sin embargo, difiere en la implementación y las herramientas de software a utilizar.

Justificación

En este proyecto nos encontramos interesados en analizar el comportamiento de los algoritmos de clasificación y *clúster*, orientados en la clasificación automática de los seis principales delitos en el área metropolitana de México, que nos permitan determinar el mejor algoritmo en cuanto a precisión y exhaustividad. Esto debido a que en la actualidad el análisis de los textos difundidos a través de las redes sociales, contribuye en la predicción de sucesos o bien permite modelar situaciones.

En su artículo prospectivo *Cognitive Storage for Big Data*[2] (Almacenamiento cognitivo para Big Data), Giovanni Cherubini, Jens Jelitto y Vinodh Venkatesan proponen la noción de un sistema de "almacenamiento de datos cognitivos", en el cual se habitúa a criterios como la estimación y obsolescencia de los datos, con el objetivo de desarrollar un algoritmo de aprendizaje que contribuya a la clasificación de los datos, en varias clases de relevancia y trabajar en conjunto con una arquitectura de almacenamiento multinivel para personalizar la ubicación de los datos. El almacenamiento de los datos de forma cognitiva considera y reúne los datos, los cuales contribuyen al procedimiento de análisis de estos mismos, convenientemente el proyecto terminal contribuirá a señalar el algoritmo optimo a emplear en la situación donde se cuente con datos afines a los del caso de estudio manifestado.

Así mismo, estamos interesados en contribuir en el análisis predictivo de los seis principales delitos del área metropolitana de México, lo cual, además, permitirá modelar la situación delictiva en esta región del país.

Objetivo

Evaluar el comportamiento de tres algoritmos de aprendizaje automático para la minería de textos con la finalidad de clasificar delitos de la zona metropolitana de México.

Objetivos Específicos

- Analizar tres algoritmos de aprendizaje automático enfocados a la minería de textos, específicamente, para la clasificación de seis tipos de delitos.

- Evaluar los tres algoritmos en el conjunto de datos sobre delitos de la zona metropolitana de México.
- Analizar los resultados de los tres algoritmos para el caso de estudio, con la finalidad de determinar el algoritmo que mejores resultados proporciona en términos de precisión y exhaustividad.

Marco teórico

Un algoritmo se entiende como un proceso de sucesiones lógicas necesarias, con el propósito de efectuar una actividad determinada, como la solución de problemas en diversos ámbitos. De esta manera se atribuye a los algoritmos la característica de independencia, ya que no se encuentran ligados a resolver casos específicos y en ambientes particulares. Es decir, en el entorno de la computación, los algoritmos se pueden programar y ejecutar en una amplia gama de lenguajes de programación y arquitecturas computacionales. Las particularidades que diferencian los métodos de los algoritmos es la precisión, debido a que un algoritmo debe ser preciso en cada uno de los pasos que desarrolla, la consistencia es una característica esencial, ejecutando el mismo algoritmo con los mismos parámetros se debe llegar a una solución análoga en ambos casos. La capacidad de término de los algoritmos, o también llamado finito, es la capacidad que tiene el algoritmo para terminar siempre su ejecución, evitando así que este continúe con su ejecución de forma perpetua. Por esta razón la definición del algoritmo invariablemente debe contener una entrada, un proceso (pasos consecutivos a realizar) y una salida.

En este contexto los pasos de un algoritmo obligatoriamente deben ser sencillos y libres de indeterminaciones o confusiones. Por ello en los lenguajes de programación se señala de forma precisa, la manera en la cual se vincula la comunicación entre los programadores y las computadoras, existen lenguajes que realizan la comunicación entre ellos y las computadoras.

Existen tres clasificaciones de lenguajes de programación utilizados, lenguaje máquina, lenguaje de bajo nivel y lenguajes de alto nivel. En donde el lenguaje máquina son las ordenes específicas que van a la computadora sin ningún intermediario en el sistema binario. El lenguaje de bajo nivel son instrucciones mnemotécnicas, que contribuyen a facilitar la programación en comparación al lenguaje máquina. Por último, el lenguaje de alto nivel es el más utilizado debido a que es independiente de la computadora y su arquitectura.

Es en este último lenguaje de programación intervienen los traductores de lenguajes, estos traducen el código fuente a código máquina. La clasificación de los traductores se distribuye en compiladores e intérpretes, este último, actúa como un traductor del código fuente en forma secuencial, es decir, traduce y ejecuta el código sucesivamente instrucción por instrucción, eludiendo almacenar la traducción de la instrucción anterior. Por otra parte, los compiladores traducen el código fuente escritos en lenguajes de alto nivel a lenguaje máquina de manera completa para después consumar la ejecución del programa traducido.

R es un entorno de software libre para computación y gráficos estadísticos, basado en un traductor de lenguaje de tipo interprete. Se interpreta y ejecuta en una amplia variedad de plataformas UNIX, Windows y MacOS [9]. RStudio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R. Incorpora una consola, editor de sintaxis que asiste en la ejecución del código,

herramientas para realizar gráficas, la depuración y la administración del espacio de trabajo [10]. R es capaz de soportar y manejar una gran cantidad de datos de forma simultánea.

Clasificación supervisada y no supervisada.
Los algoritmos de aprendizaje automático, disponen de la capacidad para clasificar elementos teniendo en cuenta métodos y reglas. La clasificación permite asignar una misma clase a uno o más elementos que poseen atributos semejantes.

Dentro de los tipos de clasificación se encuentra la clasificación supervisada, está clasificación posee la característica de contar con un conocimiento de la clasificación de un conjunto de elementos previamente catalogados. A estos elementos de conocimiento previo se le denomina conjunto de entrenamiento, el conjunto de entrenamiento es la herramienta que permite instruir al algoritmo, con la finalidad de clasificar los elementos o el conjunto de elementos que no han sido categorizados.

En contra parte del aprendizaje supervisado, el aprendizaje no supervisado no cuenta con un conjunto de entrenamiento, el aprendizaje no supervisado habitualmente recibe el nombre de *clúster*. La función del aprendizaje no supervisado es encontrar y revelar un conjunto de características semejantes entre el conjunto de elementos, para después agrupar los elementos con características afines en *clústers.*

***K-Means*: definición y función.**
El algoritmo *K-means* es un algoritmo de clasificación, que actúa o aprende de manera no supervisada, dado esto relaciona los elementos en un conjunto de acuerdo a sus particularidades, este conjunto es un *clúster*. El conjunto de elementos que se forma, se lleva a cabo disminuyendo la suma de distancias entre cada elemento y el centroide de su *clúster*. Es importante destacar que mayormente se usa la distancia cuadrática.

Para desarrollar el agrupamiento de elementos con características semejantes, el algoritmo *K-means* requiere inicializar el número de grupos, denotados por *k*, este es el número de clúster en los cuales se agrupan los elementos que se clasificaran. Posteriormente se asignan los centroides y cada elemento es asignado a su centroide más cercano, una vez que se tiene la asignación del elemento se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los elementos pertenecientes al grupo. El algoritmo repite la asignación de elementos a su centroide más cercano y la actualización del centroide, esto hasta que los centroides no se muevan.

Los elementos que se requieren agrupar se simbolizan como vectores, de dimensión d $(x_1, x_2, \ldots, x_n)$, esto para que *K-means* construya k grupos donde se requiere que el factor a disminuir sea la suma de distancias de los elementos dentro de cada grupo o clúster $S= \{S_1, S_2,\ldots,S_k\}$, a su centroide.

***K-Nearest Neighbors*: definición y función.**
K-Nearest Neighbors es un algoritmo de aprendizaje automático, que corresponde a la clasificación supervisada. El algoritmo conoce un conjunto de datos previamente clasificados, también llamados datos de entrenamiento, que clasifican las coordenadas en grupos identificados por un atributo.

El algoritmo *K-Nearest Neighbors* se fundamenta en clasificar el conjunto elementos, estableciendo cada elemento a la clase que le corresponde, de acuerdo a las características afines por cada elemento.

Las características de cada elemento son aglutinadas, para el conjunto de entrenamiento y el conjunto de prueba.

Un elemento se clasifica de acuerdo a una estimación censada de sus vecinos, es decir, los elementos ya clasificados, otorgando la clasificación más frecuente de los vecinos *k* más cercanos. Para determinar que vecinos se encuentran más cercanos del elemento a clasificar se consideran tres fórmulas para conocer su distancia, distancia euclidiana, distancia de Manhattan y distancia de Minkowsky. Cabe destacar que estas distancias se pueden calcular solo para variables continuas, es decir, aquellas variables que son numéricas y pueden contener un número infinito de valores. Por otro lado, para aquellas variables denominadas como variables categóricas, se utiliza la distancia de Hamming.

Para medir la distancia euclidiana se emplea el teorema de Pitágoras en el espacio euclídeo, esto es, un espacio vectorial normado donde la norma corresponde a la longitud (distancia) del vector. En contra parte, la distancia de Manhattan es la suma de las diferencias absolutas de sus coordenadas. La generalización de la distancia euclidiana y la distancia de Manhattan, es la mencionada distancia de Minkowsky, debido a que es una métrica en el espacio vectorial normado y considera la suma de las diferencias absolutas de las coordenadas a considerar. Para la distancia de Hamming se calcula contando el número de bits opuestos entre dos palabras que se encuentran en código binario, esto es, si un par de palabras no coinciden *d* bits (distancia d), se requieren *d* transformaciones para igualar las palabras. Es importante aclarar que cuando se encuentran variables categóricas y variables continuas en un conjunto de elementos, las variables numéricas se estandarizan entre 0 y 1.

***Naïve Bayes*: definición y función.**

El algoritmo *Naïve Bayes* es un algoritmo de clasificación, comúnmente utilizado debido a su sencillez y velocidad, este algoritmo realiza la clasificación de manera supervisada y se fundamenta en el teorema de Bayes. Este teorema calcula la probabilidad condicional de un hecho que posee aleatoriedad que está sujeto a otro hecho, es decir, la probabilidad de un evento *A* dado un evento *B*.

El algoritmo computa las probabilidades condicionales, dado un conjunto de atributos y la clase en la cual se clasifican los elementos de dichos atributos, el conjunto de entrenamiento fija la llamada probabilidad independiente, esta probabilidad conjetura la clase objetivo de cada elemento del conjunto de elementos a clasificar.

Este algoritmo localiza y revela del conjunto de elementos la existencia o carencia de ciertos atributos singulares que se definen para la clase objetivo con el conjunto de entrenamiento. La utilidad de este algoritmo se sustenta debido a que solicita un conjunto de entrenamiento reducido para conjeturar la clase objetivo del conjunto de elementos a clasificar, ya que únicamente se requiere determinar las varianzas de las variables de cada una de las clases.

Desarrollo del proyecto

Conjunto de datos

Los *Tweets* que se emplearon para someter a prueba los algoritmos de aprendizaje automático, fueron recabados desde el mes de noviembre de 2017 hasta febrero de 2018, delimitando su obtención a cuentas de *Twitter* oficiales dedicadas a la difusión de noticias relevantes en México y el área de incidencia de los delitos reportados, el área de demarcación para los delitos reportados, es la zona metropolitana de la Ciudad de México.

El acervo de *Tweets* recabados tiene una longitud de tres mil quinientos registros, cada *Tweet* está conformado por un identificador único de dieciocho dígitos numéricos, la fecha de emisión que se encuentra en CST (*Central Standard Time*) es decir cuenta con la hora, minutos y segundos de la emisión del *Tweet* con 28 caracteres alfanuméricos incluyendo espacios*,* la cuenta de *Twitter* el cual es un identificador único que se otorga a cada usuario que pertenece a esta red social con una longitud de caracteres variable por cada cuenta*,* el *Tweet* emitido o bien el mensaje que es difundido sobre este medio de comunicación con una longitud de 1 a 280 caracteres y su clasificación de este mismo (Violación, Asalto, Homicidio, Suicidio, Explotación y Secuestro), cabe mencionar que la clasificación se realizó de forma manual para cada *Tweet*. El identificador es único e irrepetible a lo largo del registro de estos *Tweets* no obstante el dato es sobrante, no se requiere al someter los *Tweets* ante los algoritmos de clasificación y agrupamiento.

El formato del archivo con el cual se almacenaron los registros fue *.ext. txt.*, es decir un archivo de texto plano debido a que no cuenta con una tipografía*,* no obstante, para realizar el procesamiento de los mismos dentro del entorno Rstudio se requiere en formato .CSV, por lo que fue preciso cambiar este formato leyendo el archivo en una hoja de cálculo, para después ser guardado como .CSV con una codificación necesaria de UTF-8 (*8-bit Unicode Transformation Format*), con la finalidad de otorgar la facilidad para representar todos los caracteres requeridos por el acervo de *Tweets*. En la Figura.0.0.1 se exhibe una muestra del archivo que corresponde a *tweets*.csv, la primera columna corresponde al identificador de cada *Tweet*, seguida de la columna de la fecha, la tercera columna corresponde a el nombre de la cuenta de usuario de la red social *Twitter,* el mensaje o *Tweet* emitido por la cuenta de usuario y, por último, la columna de la clasificación que se le otorgo al *Tweet*.

	A	B	C	D	E
1	9.3174E+17	Fri Nov 17 22:38:04 CST 2017	NTelevisa_com	#EnPunto Peligra atención m	Secuestro
2	9.3175E+17	Fri Nov 17 23:07:04 CST 2017	NTelevisa_com	Se metieron a robar un cajer	Asalto
3	9.3177E+17	Sat Nov 18 00:27:23 CST 2017	NTelevisa_com	Con camionetas y cuerdas in	Asalto
4	9.3195E+17	Sat Nov 18 12:01:01 CST 2017	SSP_CDMX	Detuvimos a tres sujetos acu	Asalto
5	9.3198E+17	Sat Nov 18 14:07:34 CST 2017	elsolde_mexico	Comando ejecuta a dos pers	Homicidio
6	9.3202E+17	Sat Nov 18 17:00:01 CST 2017	lajornadaonline	#Morrissey cuestiona ola de	Violación
7	9.3204E+17	Sat Nov 18 18:16:00 CST 2017	Milenio	El #Vaticano investigará abu	Violación
8	9.3205E+17	Sat Nov 18 18:40:00 CST 2017	Pajaropolitico	La violencia aumentó en el p	Homicidio
9	9.3206E+17	Sat Nov 18 19:36:00 CST 2017	Milenio	'Naturista' envenena a 14 y r	Homicidio
10	9.3207E+17	Sat Nov 18 20:10:00 CST 2017	El_Universal_Mx	Los homicidios ocurrieron en	Homicidio
11	9.3207E+17	Sat Nov 18 20:34:01 CST 2017	Pajaropolitico	A un año de que @EPN term	Homicidio
12	9.3213E+17	Sun Nov 19 00:17:00 CST 2017	Pajaropolitico	Los niños indígenas en Méxi	Homicidio
13	9.3219E+17	Sun Nov 19 04:30:00 CST 2017	NoticiasMVS	.@Uber encara demanda col	Violación
14	9.3231E+17	Sun Nov 19 11:55:25 CST 2017	Notimex	15 personas murieron y 5 res	Homicidio
15	9.3233E+17	Sun Nov 19 13:25:00 CST 2017	El_Universal_Mx	La PGJ cumplimentó las órde	Asalto
16	9.3233E+17	Sun Nov 19 13:45:00 CST 2017	El_Universal_Mx	El asesinato fue perpetrado	Asalto
17	9.3237E+17	Sun Nov 19 15:55:02 CST 2017	GoogleNewsMX	Silencioso aumento: precio (	Suicidio
18	9.3238E+17	Sun Nov 19 16:49:08 CST 2017	Reforma	El director de Izzi, Adolfo Lag	Homicidio
19	9.324E+17	Sun Nov 19 18:26:38 CST 2017	NTelevisa_com	Los homicidios contra mujer	Homicidio
20	9.3242E+17	Sun Nov 19 19:12:30 CST 2017	Milenio	#PGR colaborará en caso de a	Homicidio
21	9.3242E+17	Sun Nov 19 19:30:01 CST 2017	elsolde_mexico	#Almomento | @EPN conde	Homicidio
22	9.3243E+17	Sun Nov 19 19:51:32 CST 2017	Siete24Noticias	La @PGR_mx participará en l	Homicidio

FIGURA.0.0.1 MUESTRA DEL ARCHIVO TWEETS.CSV.

La lectura de los datos en el entorno de desarrollo Rstudio se realizó como se muestra en la Figura: 0.0.2

```
#**********************Read Tweets**********************
tweets = read.csv("tweets.csv", header = F, stringsAsFactors = FALSE,
                  fileEncoding = "latin1")
#To remove any missing value that might be present in the data, type this:
typeEmpty = which(tweets$V5 == "")
tweets = tweets[-c(typeEmpty), ]
id   = tweets$V1
date = tweets$V2
user = tweets$V3
text = tweets$V4
type = tweets$V5
```

FIGURA.0.0.2 LECTURA DE LOS TWEETS PROVENIENTES DE UN ARCHIVO .CSV.

Como puede observarse, en la *Figura.0.0.2*, se creo un variable llamada *tweets* la cual almacenó los datos provenientes del archivo *tweets*.csv, especificando que el documento no contiene cabeceras, no contiene cadenas como elementos y precisando que el archivo se encuentra con una codificación latina. Es sustancial que se eliminen los *Tweets* que se encuentran sin una clasificaciónclasificación previa, por lo que se recabaron para despues ser eliminados del registro.

La manipulación de los datos se considera mas asequible cuando se encuentran identificados y catalogados de forma especifica, por lo que se crearon variables que contengan a cada elemento que conforma elregistro de cada *Tweet*, id contiene el identificador por cada *Tweet* emitido, date la fecha, user la cuenta de usuario de *Twitter,* text el *Tweet* y type la clasificación que se le dio al *Tweet*.

Para poder contener las variables mencionadas en un solo registro, se utilizó un *data frame*, el cual es una estructura de forma rectangular la cual almacena datos de diferente tipo de caracteres, por lo que se utilizó esta estructura para contener y manipular los datos que contiene cada *Tweet*, este procedimiento se puede observar en la figura: 0.0.0.0

```
#**************Create DataFrame*****************************
df = data.frame(id, date, user, text, type)
```

FIGURA.0.0.3 CREACIÓN DE UN DATA FRAME CON LOS DATOS DE CADA TWEET.

Para el procesamiento de los datos no se requirió la hora que contiene el campo *date*, por lo que se eliminó este dato y se cambió el formato de la fecha, dado que el formato anterior excedia el número de caracteres y la representacion grafica del dato *date* resultaba exceder los margenes de la imagen.

```
#**********************Transform Date**********************
#Extract current configuration
my_date = strptime(df$date, tz = 'CST', format = "%a %b %d %H:%M:%S CST %Y")
day     = day(my_date)
year    = year(mi_date)
mounth  = month(mi_date)
newDate = paste(mounth,year)
df$date = newDate
```

FIGURA.0.0.4 MODIFICACIÓN DE FORMATO DEL CAMPO DATE.

En la Figura.0.0.4 se muestra el procedimiento llevado a cabo para la modificacion del formato del campo *date*, se realizó la organización de los elemntos que integran la fecha en *day*, *mounth* y *year*, con la intención de agrupar y representar la fecha exclusivamente en un formato de mes y año.

La graficación de elementos cuantitativos es una ayuda visual significativa, por lo cual en este caso, es un elemento que nos proporciona la capacidad de visualizar y comprender los elementos que integran el conjunto de datos que se manejó. En la Figura.0.0.5 se muestran las cuentas o usuarios de *Twitter* oficiales que se utilizaron para recabar los *Tweets* graficando el número de *Tweets* emitidos en el periodo de tiempo establecido, se observa que la cuenta con mayor número de *Tweets* emitidos fue NTelevisa_com. En contra parte, las cuentas con menor número de *Tweets* fueron cibercrimen, MexDenuncia y noticimexico.

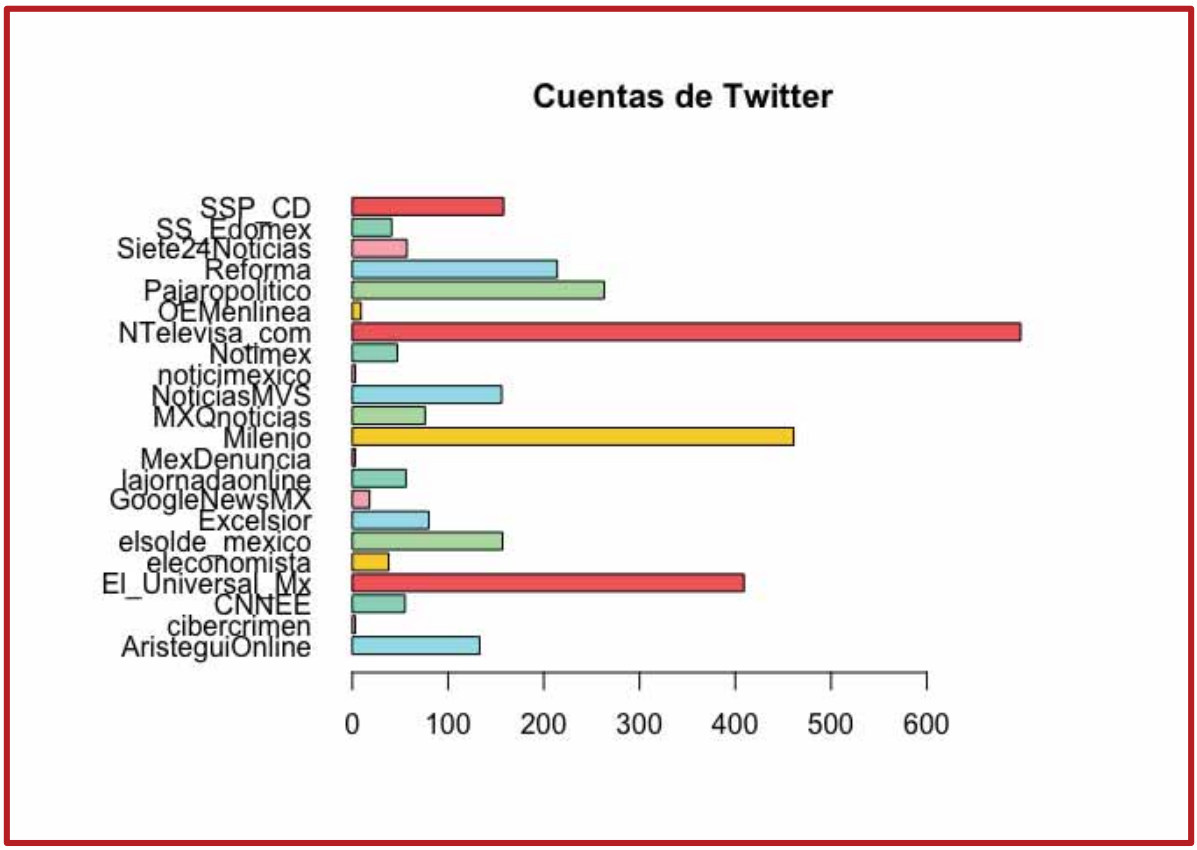

FIGURA.0.0.5 GRAFICA DE BARRAS DE LAS CUENTAS DE TWITTER Y EL NÚMERO DE TWEETS EMITIDOS.

La Tabla.0.1 muestra las cuentas de *Twitter* en conjunto con el número de *Tweets* emitidos en el periodo de tiempo que se establecio.

TABLA.0.1 TABLA DE LAS CUENTAS DE TWITTER Y EL NÚMERO DE TWEETS EMITIDOS.

Cuenta de usuario de *Twiiter*	Número de *Tweets* emitidos
AristeguiOnline	133
cibercrimen	3
CNNEE	55
El_Universal_Mx	409
eleconomista	38
elsolde_mexico	157
Excelsior	80
GoogleNewsMX	18
lajornadaonline	56
MexDenuncia	3
Milenio	461
MXQnoticias	76
NoticiasMVS	156
noticimexico	3

Notimex	47
NTelevisa_com	698
OEMenlinea	9
Pajaropolitico	263
Reforma	214
Siete24Noticias	57
SS_Edomex	41
SSP_CDMX	158

La clasificación de los delitos reportados es el dato fundamental a tener en cuenta, por este motivo se realizó su representacion gráfica, la cual se puede observar en la Figura.0.0.6, en dicha figura las barras representan la clasificaciónclasificación Asalto, Explotación, Homicidio, Secuestro, Suicidio y7 Violación, respectivamente de izquierda a derecha y de acuerdo a la incidencia reportada.

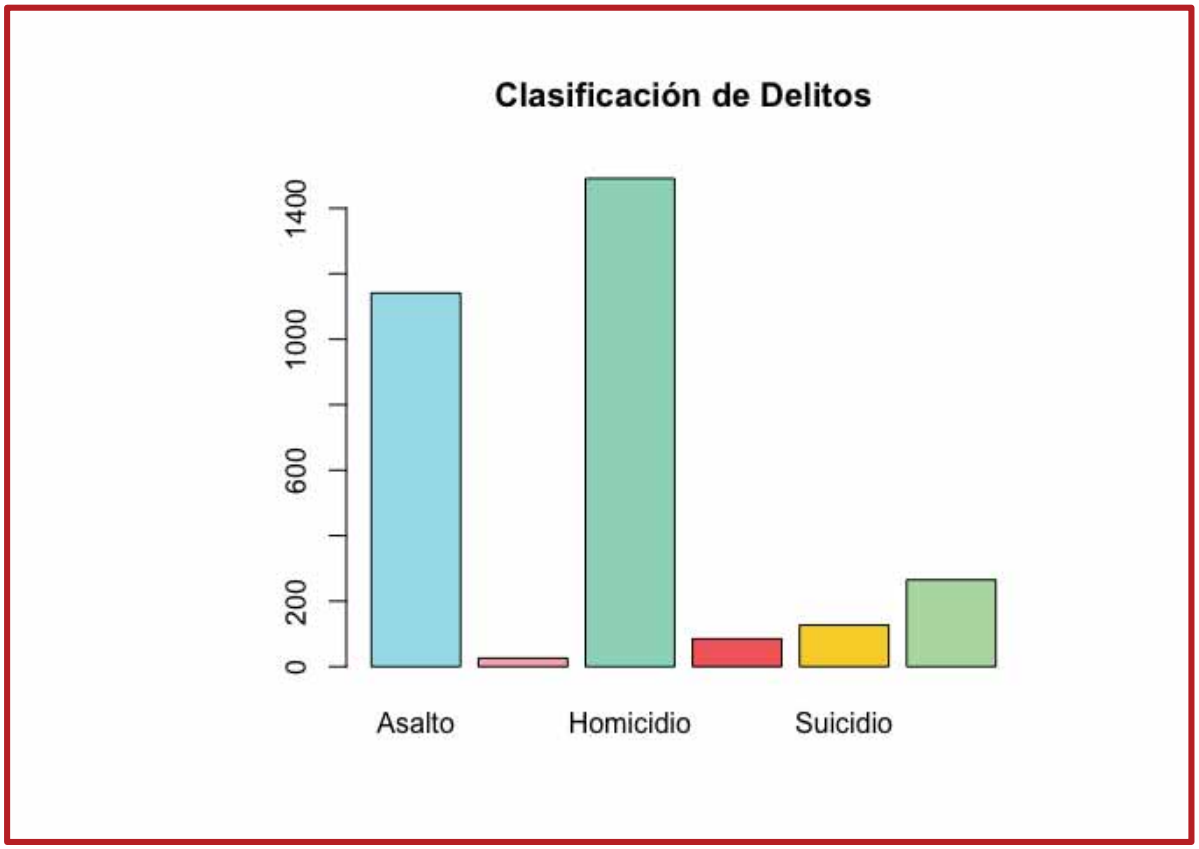

FIGURA.0.0.6 GRAFICA DE BARRAS DE LA CLASIFICACIÓN DE TWEETS Y EL NÚMERO DE INCIDENCIA

Se destaca el reporte de Homicidios ya que es el delito con mayor número de *Tweets* clasificados en esta categoria, por otro lado la categoria Explotación resulto ser la clasificaciónclasificación con menor número de *Tweets*.

En la Tabla.0.2 se exhiben las cuentas de *Twitter* y el número de *Tweets* emitidos de acuerdo a su clasificación de cada uno de ellos. Pudiendo observar que la clase Homicidio resulto ser a la que se clasificaciónle otorgó el mayor número de *Tweets*, éstos emitidos por la cuenta de usuario

NTelevisa_com, con un total de trescientos treinta y dos *Tweets*. Teniendo como contra parte, un solo *Tweet* clasificado, la categoria de Explotación. Por otro lado, las cuentas de usuario cibercrimen, MexDenuncia y noticimexico resultaron ser las que tienen un menor número de *Tweets* emitidos, sin embargo, coincidieron en el número mayor de *Tweets* clasificados en la categoria de Asalto.

TABLA.0.2 TABLA DE LAS CUENTAS DE TWITTER Y EL NÚMERO DE TWEETS EMITIDOS POR CLASIFICACIÓN.

Cuentas de Usuario de Twitter	Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación
AristeguiOnline	22	2	85	6	5	13
cibercrimen	2	1	0	0	0	0
CNNEE	2	3	29	0	2	19
El_Universal_Mx	146	3	177	6	20	57
eleconomista	17	0	16	1	1	3
elsolde_mexico	44	1	78	5	10	19
Excelsior	32	0	35	3	6	4
GoogleNewsMX	5	2	8	2	1	0
lajornadaonline	9	0	38	1	0	8
MexDenuncia	2	0	1	0	0	0
Milenio	136	3	234	12	36	40
MXQnoticias	41	0	28	1	6	0
NoticiasMVS	32	4	89	4	6	21
noticimexico	3	0	0	0	0	0
Notimex	17	3	23	3	0	1
NTelevisa_com	267	1	332	32	22	44
OEMenlinea	0	0	6	1	1	1
Pajaropolitico	81	3	151	5	4	19
Reforma	66	0	127	3	6	12
Siete24Noticias	27	0	25	0	1	4
SS_Edomex	40	0	1	0	0	0
SSP_CDMX	150	0	8	0	0	0

Para poder tener una representacion visual de los elementos descritos en la *Tabla.0.2* se graficaron las cuentas de usuario de *Twitter* por la clasificaciónclasificación del número de *Tweets* emitidos con respecto a cada una de éstas. Dicha respresentacion, gráfica, se muestra en la Figura.0.0.7, las líneas puntuadas señalan la ausencia de *Tweets* clasificados en la categoria, mientras que los bloques representan la cantidad en volumen de los *Tweets* clasificados en la categoria por cada cuenta de usuraio.

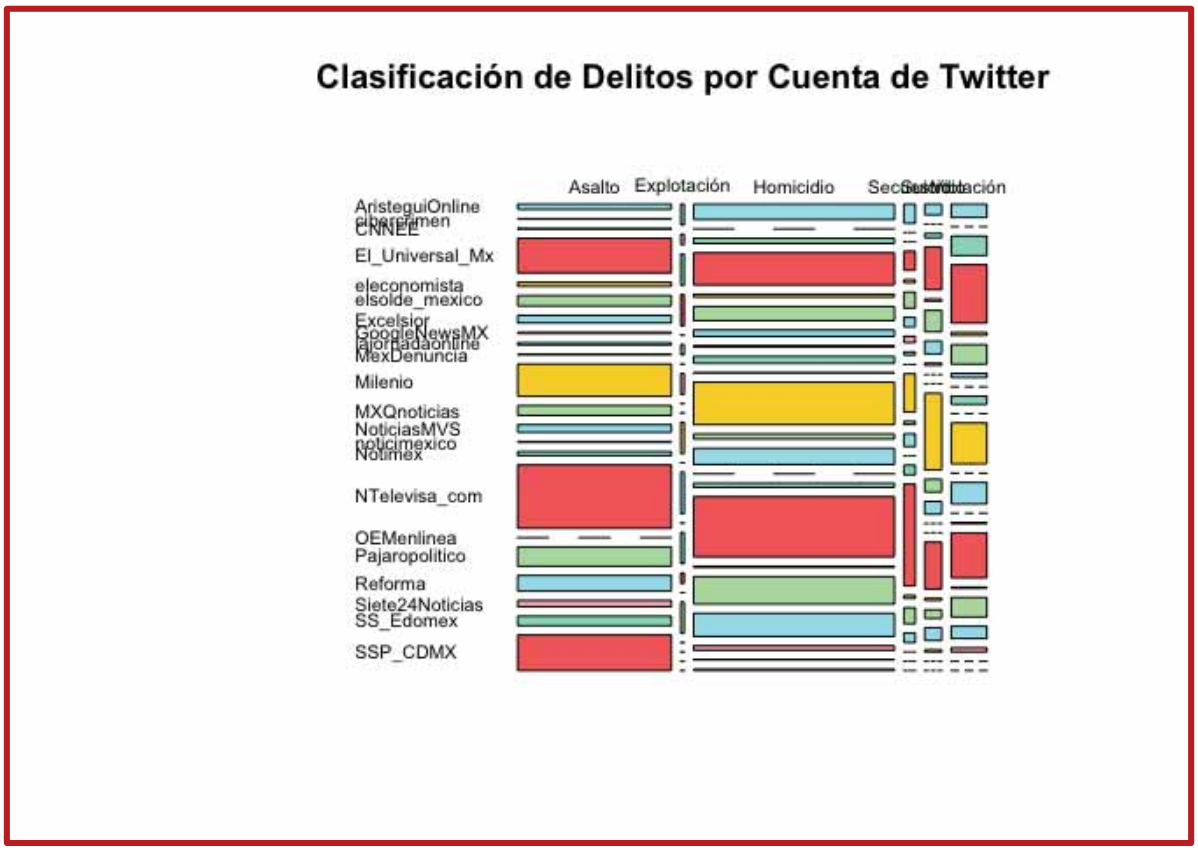

FIGURA.0.0.7 GRÁFICA DE LAS CUENTAS DE TWITTER Y EL NÚMERO DE TWEETS EMITIDOS POR CLASIFICACIÓN.

Es valioso conocer el número de *Tweets* emitidos en determinado mes con respecto a su clasificación proporcionada, por lo cual en la Tabla.0.3 se modela esta información. En dicha tabla se puede observar que la categoría de Homicidio resulto ser la clasificación con mayor número de incidencias en todos los meses que se recabaron estos *Tweets*, entre tanto Explotación resulto ser la categoría con menor número de *Tweets* clasificados en esta categoría en todos los meses. Cabe destacar que el mes de enero de 2018 se emitieron la mayor cantidad de *Tweets,* con un total de 1,377 emisiones, de las cuales se puede observar que Asalto resulto ser la categoría con mayor incremento, mientras que Homicidio bajo el número de *Tweets* clasificados en esta categoría.

TABLA.0.3 TABLA DE LA CLASIFICACIÓN DE TWEETS Y EL NÚMERO DE TWEETS EMITIDOS POR FECHA.

Fecha (mes / año)	Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación
11 / 2017	78	2	117	5	21	24
12 / 2017	462	15	658	48	53	86
1 / 2018	523	8	621	28	51	146
2 / 2018	78	1	95	4	2	9

La frecuencia en la que se emitieron los *Tweets* por cada cuenta de usuario de *Twitter* es un factor a tener en cuenta. Por lo cual, en la Tabla.0.4 se muestra esta relación, además se puede observar que el mes de mayor actividad para la cuenta NTelevisa_com es diciembre de 2017, como se ha mencionado es la cuenta con mayor número de *Tweets* emitidos, en comparación con la cuenta noticimexico, que únicamente emitió tres *Tweets,* los cuales fueron publicados en el mes de enero de 2018.

TABLA.0.4 TABLA DE LA CUENTAS DE USUARIO Y EL NÚMERO DE TWEETS EMITIDOS POR FECHA.

Cuenta de usuario / Fecha	11-2017	12-2017	1-2018	2-2018
AristeguiOnline	6	67	57	3
cibercrimen	0	0	3	0
CNNEE	6	22	27	0
El_Universal_Mx	35	162	192	20
eleconomista	3	14	16	5
elsolde_mexico	17	58	78	4
Excelsior	2	32	38	8
GoogleNewsMX	1	12	4	1
lajornadaonline	4	24	26	2
MexDenuncia	0	2	1	0
Milenio	30	218	194	19
MXQnoticias	4	22	44	6
NoticiasMVS	12	51	81	12
noticimexico	0	0	3	0
Notimex	2	17	28	0
NTelevisa_com	72	313	255	58
OEMenlinea	1	3	5	0
Pajaropolitico	23	113	115	12
Reforma	10	97	81	26
Siete24Noticias	2	18	34	3
SS_Edomex	4	23	14	0
SSP_CDMX	13	54	81	10

Dado que el factor de interés fue clasificar el texto de los *Tweets* emitidos, se creó el corpus únicamente con esta variable de interés, el corpus es la colección de documentos que contienen texto en lenguaje natural, el corpus puede contener dos tipos de metadatos, en este caso el corpus que se creó no contiene metadatos. En la Figura.0.0.8 se muestra cómo se estableció el corpus a base de los textos de cada *Tweet*.

```
#**************Building  Corpus*****************************
myCorpus = Corpus(VectorSource(df$text))
```

FIGURA.0.0.8 CREACIÓN DEL CORPUS A BASE DE LOS TEXTOS DE CADA TWEET.

La funcion VectorSource es la herramienta que nos proporcionó la capacidad de poder convertir cada *Tweet* del vector df$text como un documento único.

Para poder realizar el procesamiento de los textos por cada algoritmo y ejecutar de forma gráfica su representación, es conveniente realizar una limpieza del texto, es por esto que el corpus creado se sometio a dicha limpieza, en la cual se convirtió cada letra mayuscula a minuscula, se eliminaron signos de puntuación, números, palabras vacias las cuales son aquellas palabras que carecen de significado presentadas de forma individual y los *links* (enlace a un documento electrónico) que acompañan a algunos textos, este proceso en conjunto es denominado preprocesado del corpus. En la Figura.0.0.9 se ejemplifica el procedimiento llevado a cabo para realizar el preprocesado de forma adecuada.

```
#***********Convert to lowercase*******************
myCorpus = tm_map(myCorpus, tolower)

#***********Remove Puntuacions*********************
myCorpus = tm_map(myCorpus, removePunctuation)

#***********Remove numbers*************************
myCorpus = tm_map(myCorpus, removeNumbers)

#***********Remove URLs****************************
removeURL = function(x)
  gsub("http[^[:space:]]*", "", x)
myCorpus = tm_map(myCorpus, content_transformer(removeURL))

#***********Remove Stopwords***********************
myStopWords = c(stopwords('spanish'), "available", "via")
myCorpus    = tm_map(myCorpus, removeWords, myStopWords)
myCorpus
inspect(myCorpus)
summary(myCorpus)
```

FIGURA.0.0.9 PROCEDIMIENTO DEL PROCESAMIENTO DEL CORPUS.

La matriz denominada *TermDocumentMatriz*, o bien matriz de documento de término, es la matriz que permite describir la frecuencia de las palabras contenidas en una colección de documentos, las filas corresponden a los términos en la colección y las columnas corresponden a los documentos. Existen varios métodos para determinar el valor que debe tener cada entrada en la matriz. En el caso, de representar la frecuencia de las palabras contenidas en cada documento, o texto de cada *Tweet*, se proporcionó a la función TermDocumentMatriz el corpus creado y como segundo parámetro de control la frecuencia que admite la entrada a la matriz; esta frecuencia es igual a uno, es decir, toda palabra que aparezca en los documentos es ingresada a la matriz con su respectiva frecuencia por cada documento.

Se requirio conocer las primeras sesenta y cinco palabras más frecuentes dentro del corpus y la frecuencia por cada una de ellas. Con una frecuancia no menor a cincuenta, se realizó la

representación gráfica de estas palabras, tal que ejemplifica de forma visual que la palabra con mayor frecuencia es robo, mientras que la de menor frecuncia dentro de las más frecuentes es denuncia y conductor, esta gráfica se puede observar en la Figura.0.1.0.

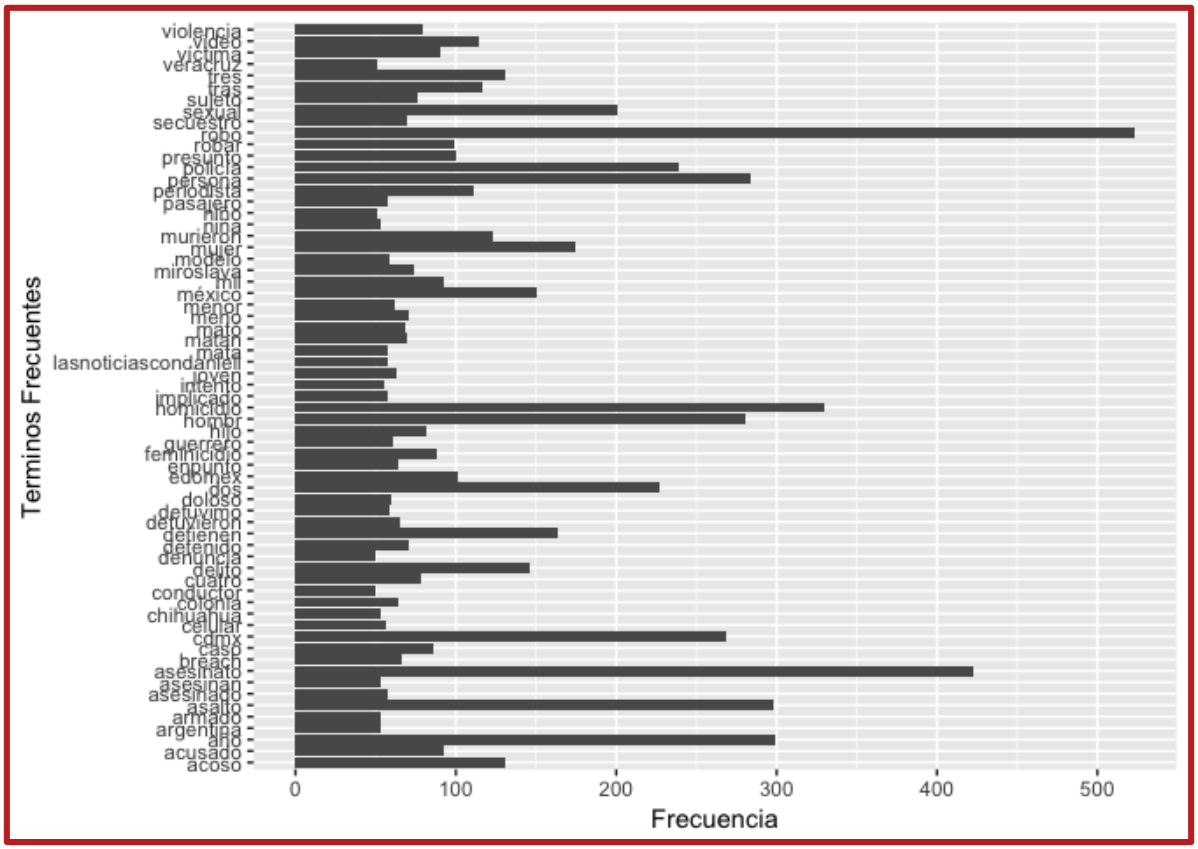

FIGURA.0.1.0. TÉRMINOS MÁS FRECUENTES NO MENORES A CINCUENTA REPETICIONES.

Una representación gráfica útil y conveniente para ejemplificar de forma adecuada la frecuencia de las palabras en cuestión es *Word Cloud* o nube de palabras, este recurso nos permite colocar las palabras que se encuentran dentro de un texto, conforme a la frecuencia en la que estas aparecen dentro del texto, resaltando aquellas con mayor frecuencia. Por lo que en la Figura.0.1.1 se exhibe la nube de palabras, o *Word Cloud,* con una frecuencia mínima de 50, lo que establece que las palabras que aparecen dentro de esta representación tienen una frecuencia mayor a 50.

FIGURA.0.1.1 WORD CLOUD CON UNA FRECUENCIA MAYOR A CINCUENTA REPETICIONES.

El procedimiento que se elaboró para poder representar la frecuencia de las palabras en una nube se muestra en la Figura.0.1.2

```
pal3 = brewer.pal(8,"RdGy")
wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=50,max.words=Inf, rot.per=.15, colors=pal3)
```

FIGURA.0.1.2. PROCEDIMIENTO PARA REPRESENTAR UNA NUBE DE PALABRAS CON UNA FRECUENCIA MÍNIMA A CINCUENTA REPETICIONES.

Es importante señalar que las representaciones de nube de palabras se crean de forma aleatoria, es decir, la organización y la aparicion de las palanbras se hacen de forma incierta, las palabras se situan en un espacio que este vacante y cumpla con las dimensiones adecuadas teniendo en cuenta que no se sobreponga una de otra.

La Figura.0.1.3. se muestra la nube de palabras con una frecuencia mayor a 40 repeticiones, en esta nueva nube de palabras, se puede recalcar la aleatoriedad que se tiene para representar la frecuencia de palabras en este elemento visual, ya que la pabra "*murieron*"que aparece en la nube de palabras de la Figura.0.1.2. ya no aparece en la nueva nube, teniendo en cuenta que tiene frecuencia mayor a 40 repeticiones.

FIGURA.0.1.3. WORD CLOUD CON UNA FRECUENCIA MAYOR A CUARENTA REPETICIONES.

La palabra “asalto” no apareció en las nubes de palabras mostradas anteriormente, no obstante aparece en la nube de palabras de la Figura.0.1.4. de forma destacada.

FIGURA.0.1.4. WORD CLOUD CON UNA FRECUENCIA MAYOR A TREINTA REPETICIONES.

En la nube de palabras de la Figura.0.1.5. podemos precisar las palabras relacionadas con la categoria de “Homicidio”, ya que aparecen palabras como “asesinan”, “mato”, “matan”, “muerto”, entre otras. Estas palabras refieren que el mayor número de *Tweets* se encuentran clasificados en la categoria de “Homicidio”.

FIGURA.0.1.5. WORD CLOUD CON UNA FRECUENCIA MAYOR A VEINTE REPETICIONES.

En la Figura.0.1.6. se muestra que palabras como “murieron” y “mató” toman un mayor peso frente a otras palabras que aparecen en las nubes de palabras anteriores.

FIGURA.0.1.6. WORD CLOUD CON UNA FRECUENCIA MAYOR A DIEZ REPETICIONES.

En la representación de la frecuencia de palabras mayor a 6 repeticiones, que se muestra en la Figura.0.1.7, se observa que la aparición de la palabra “policía” se representa de forma sobresaliente en comparación con las palabras que aparecen en la misma nube de palabras.

FIGURA.0.1.7. WORD CLOUD CON UNA FRECUENCIA MAYOR A SEIS REPETICIONES.

Algoritmo *K-means*

El algoritmo de clasificación *K-means* dirigido al caso de estudio de clasificar seis tipos de delitos en el área metropolitana de México, se efectuó estableciendo una semilla o *seed*, con la principal necesidad de obtener resultados reproducibles, esto es posible debido a la aleatoriedad, que se desarrolla a lo largo de la ejecución del algoritmo y que tiene la habilidad de reproducir esta misma aleatoriedad.

La dispersión es un umbral de la frecuencia relativa del corpus para un término, sobre del cual se eliminará el término. La frecuencia relativa del corpus en este caso se refiere a una proporción, dado esto la dispersión es menor a medida que se aproxima a 1. En la Figura.0.2.0 se puede observar el procedimiento para eliminar los *Sparse Terms* (términos escasos), en donde se sitúa *sparse* = 0.95 lo que establece que se eliminó solo los términos que sean más escasos que 0.95.

```
#***************remove sparse terms*********************************
inspect(myCorpus[1:3])
removeSparseTDM = removeSparseTerms(tdm, sparse = 0.95)
inspect(removeSparseTDM)
matrixRemoveSparseTDM = as.matrix(removeSparseTDM)
write.csv(matrixRemoveSparseTDM, file = "matrixRemoveSparseTDM.csv")
```

FIGURA.0.2.0. FUNCIÓN REMOVESPARSETERMS CON SPARSE IGUAL A CERO PUNTO NOVENTA Y CINCO.

Se requirió realizar un análisis de *clúster* jerárquico sobre el conjunto de datos *Sparse Terms*, dado esto se realizó la gráfica de este resultado como se muestra en la Figura.0.2.1. en la que se establece que "asesinato" es la palabra con una reiteración superior en el conjunto de datos *Sparse Terms*, seguido de la palabra "robo", se observa que estas palabras en conjunto con "homicidio", "asalto" y "persona" se encuentran clasificadas en un solo *clúster*. Mientras que las palabras "sexual", "año", "hombre", "cdmx", "policía", "dos", "mujer" y "detienen" se encuentran en un *clúster* de forma conjunta, esto se debe a que se especificó un número de *clúster* igual a seis.

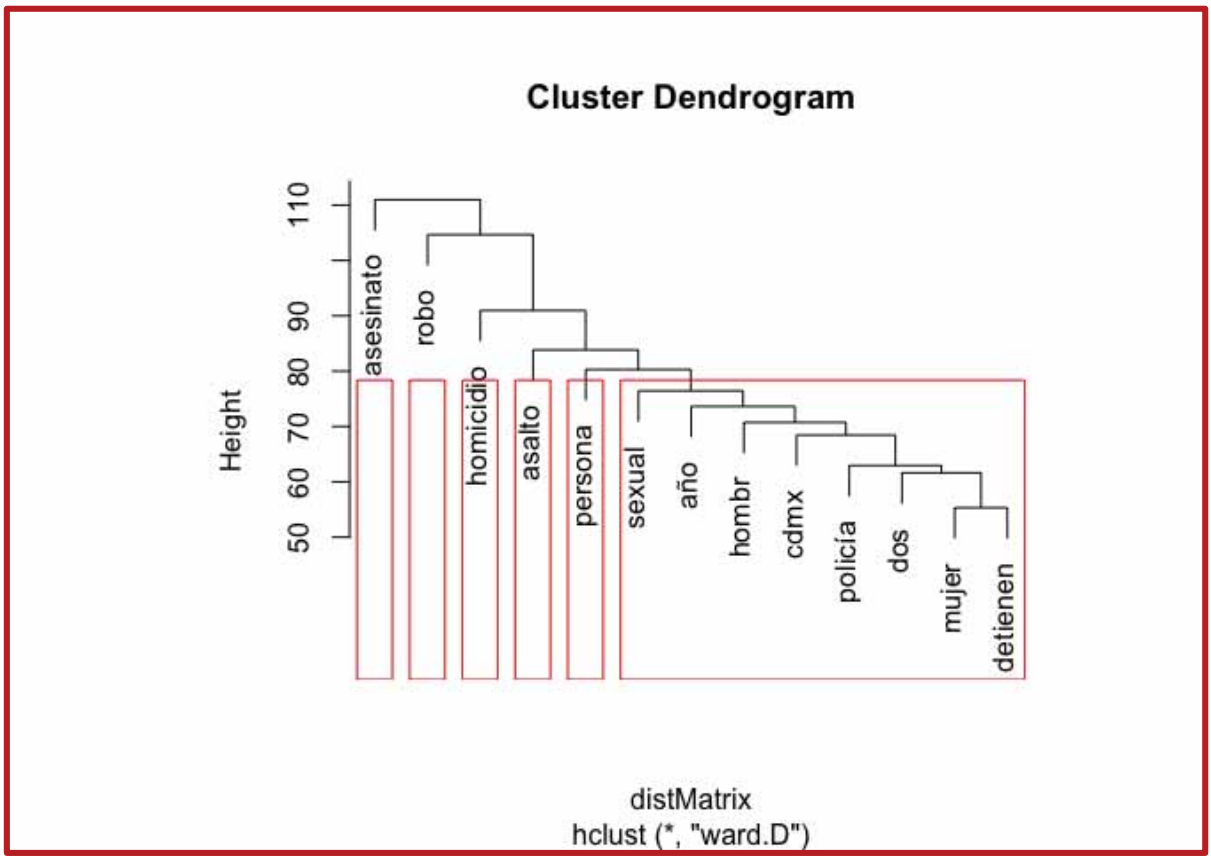

FIGURA.0.2.1 GRÁFICA DE BARRAS DE LA CLASIFICACIÓN DE *TWEETS* CON RESPECTO AL NÚMERO DE INCIDENCIA.

Es requerido para la ejecución del algoritmo *K-means* transponer la *matrizRemoveSparseTDM* que se puede ver en la Figura.0.0.0.0 a los documentos del clúster (*Tweets*). Así bien, se estableció un número de *clúster* igual a tres para la ejecución del algoritmo, el procedimiento se puede observar en la Figura.0.2.2, al algoritmo se le proporcionó la matriz transpuesta y el número de *clúster* requeridos.

```
#**************k-means clustering of tweets k = 3*****************
kmeansResult3 = kmeans(m3, 3)
kmeansResult3
tablekmeansResult3 = table(kmeansResult3$cluster)
write.csv(tablekmeansResult3, file = "tablekmeansResult3.csv")
```

FIGURA.0.2.2 EJECUCIÓN DEL ALGORITMO *K-MEANS* CON K IGUAL A TRES.

En la Tabla.0.5. se observa de forma enumerada los *Tweets* asiganados a cada *clúster,* en el primer *clúster* se encuentran clasificados la mayoría de los *Tweets* con un número total de dos mil trescientos dieciocho, por otra parte el *clúster* con menor número de *Tweets* clasificados en él, es el número tres; en el cual se encuentran tan solo doscientos sesenta y ocho.

TABLA.0.5. TABLA DE LOS TWEETS ASIGNADOS A UN NÚMERO DE CLÚSTER IGUAL A TRES.

Clúster	*Tweets*
1	2318
2	549
3	268

Las presentaciones gráficas de *clústers* es una herramienta para realizar un analisis visual de los elementos que integran a cada *clúster* y como estos interactuan con los demás *clústers*, frecuentemente los gráficos se muestran en estructuras de árbol, no obstante pueden representarse en términos de correlación de elemento por clúster. Como se puede observarse en la Figura.0.2.3. se distinguen los tres *clúster* y los elementos dentro de ellos. Además, se puede destacar la condición de los *clúster* al tener elementos (*Tweets*) clasificados en los tres *clúster*, esta particularidad se produjó cuando el algoritmo calcula los centroides y determina la distancia a la cual deben estar los *Tweets* para ser asignados a cada *clúster,* teniendo centroides adyacentes cercanos, lo cual origina el translape entre los *clúster*.

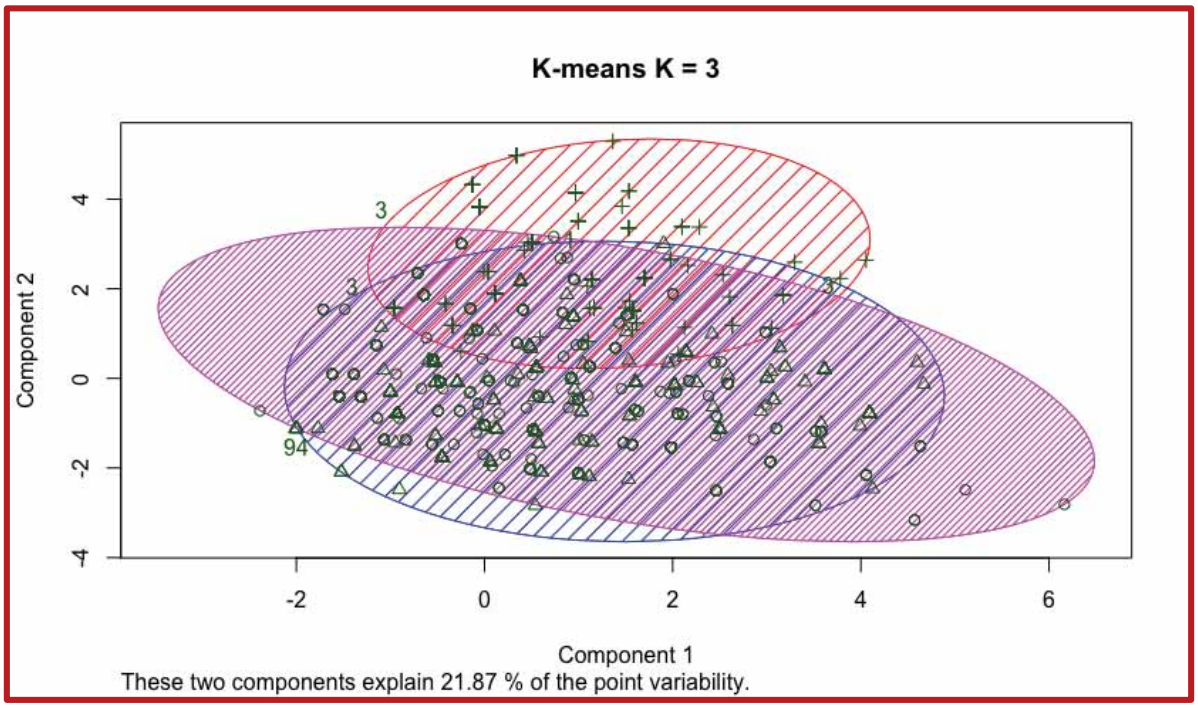

FIGURA.0.2.3. GRÁFICA DE TRES CLÚSTERS CON EL ALGORITMO *K-MEANS* .

Para considerar un análisis del algoritmo *K-mean,* íntegro y fehaciente, se requirió efectuar múltiples ejecuciones del algoritmo, modificando la variable de interés. Es decir, el número de *clúster* en los que se clasifican los *Tweets*, por lo que en la Figura.0.2.4. se observa el cumplimiento de este requerimiento, modificando la variable que determina el número de *clústers* que se emplearon, en esta segunda ejecución del algoritmo se dispuso de seis *clústers*.

```
#***************k-means clustering of tweets k = 6***************
kmeansResult6 = kmeans(m3, 6)
kmeansResult6
tablekmeansResult6 = table(kmeansResult6$cluster)
write.csv(tablekmeansResult6, file = "tablekmeansResult6.csv")
```

FIGURA.0.2.4 EJECUCIÓN DEL ALGORITMO *K-MEANS* CON K IGUAL A SEIS.

En la Tabla.0.6. se observa de forma enumerada los *Tweets* asiganados a cada *clúster*, en la que se destaca la clasificaciónclasificación de un mayor número de *Tweets* a el *clúster* número uno, mientras que el *clúster* número cuatro contiene el menor número de *Tweets* clasificados, con solo ciento veinticuatro *Tweets*.

TABLA.0.6. TABLA DE LOS TWEETS ASIGNADOS A UN NÚMERO DE CLÚSTER IGUAL A TRES.

Clúster	*Tweets*
1	1967
2	412
3	242
4	124
5	186
6	204

El análisis de los elementos asignados a los diferentes *clúster,* también se realiza mediante la representación gráfica, diferenciando entre *clúster* por el tipo de color de cada uno de ellos. En la Figura.0.2.5. se muestra la representación gráfica de el algoritmo *K-means* con un número de seis *clústers*, en el cual se observa el empalme entre cada uno de los *clúster*, esto advierte que al seguir aumentando el número de *clústers* en donde son clasificados los *Tweets* se incrementa el empalme de éstos.

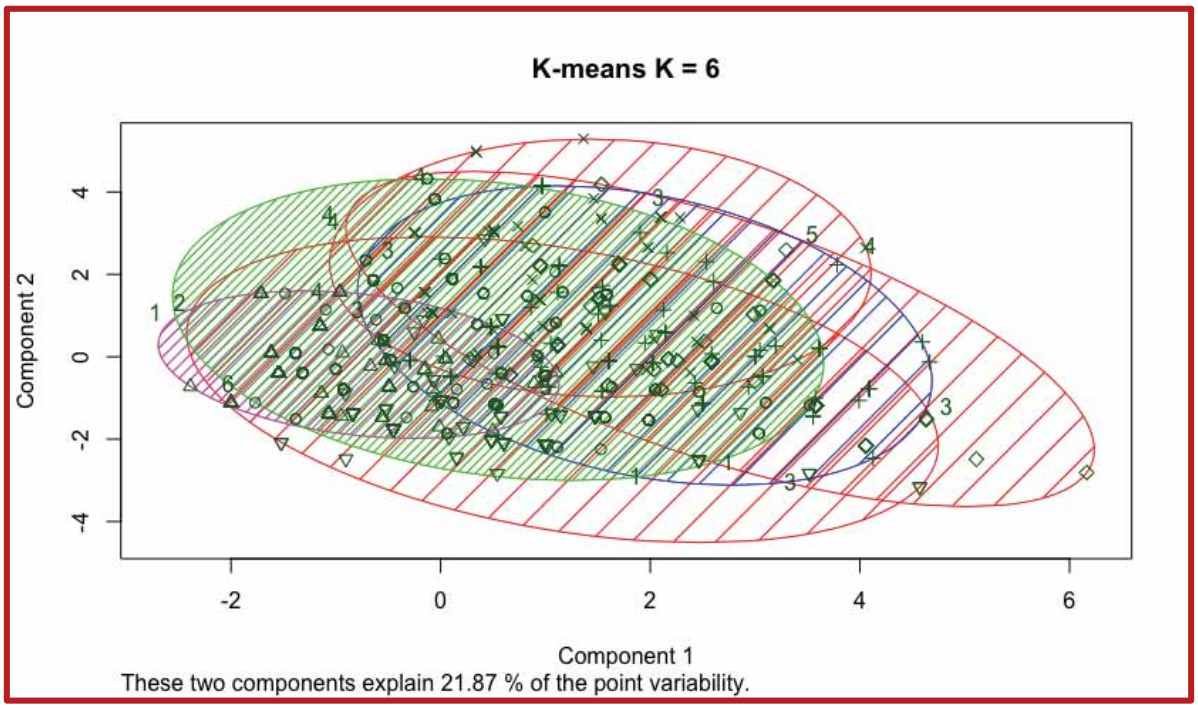

FIGURA.0.2.5 GRAFICA DE SEIS CLÚSTERS CON EL ALGORITMO *K-MEANS*.

Oportunamente se determinó reducir el número de *clústers,* con la finalidad de examinar la estructura en la cual se clasificaron los *Tweets*, en la Figura.0.2.6. se señala el procedimiento que se realizó al ejecutar el algoritmo con un número de dos *clúster*.

```
#***************k-means clustering of tweets k = 2***************
kmeansResult2 = kmeans(m3, 2)
kmeansResult2
tablekmeansResult2 = table(kmeansResult2$cluster)
write.csv(tablekmeansResult2, file = "tablekmeansResult2.csv")
```

FIGURA.0.2.6. EJECUCIÓN DEL ALGORITMO *K-MEANS* CON K IGUAL A DOS.

Para observar la asignacion de los *Tweets* a cada *clúster*, se realizó una tabulacion, la cual ejemplificó de forma correcta esta asignacion y se contempla que la mayoria de los *Tweets* fueron clasificados al primer *clúster*. Esto nos dice que la distancia del limite del *clúster* al centroide es más amplia en comparacion con la distancia del segundo *clúster*, la clasificaciónclasificación de los *Tweets* puede contemplarse en la Tabla.0.7.

TABLA.0.7. TABLA DE LOS TWEETS ASIGNADOS A UN NÚMERO DE CLÚSTER IGUAL A DOS.

Clúster	*Tweets*
1	2837
2	298

En las visualizaciones gráficas anteriores, Figura.0.2.3. y la Figura.0.2.5., de la clasificación de los *Tweets* en cada *clúster*, se ha presentado un empalme de estos, advirtiendo que los *Tweets* han sido clasificados en dos o más *clústers*, originado a que el límite de un *clúster* sobre pasa el límite de un segundo *clúster*, teniendo como limite la cercanía a un centroide del segundo *clúster*. Para constatar la premisa anterior se requirió contemplar de forma gráfica la clasificación de los *Tweets* en los dos *clúster* mencionados. En la Figura.0.2.7. se visualiza la representación gráfica de la clasificación de los Tweets en dos clúster, como se previno se origina el empalme de estos clúster ocasionando que existan *Tweets* clasificados en los dos *clúster*.

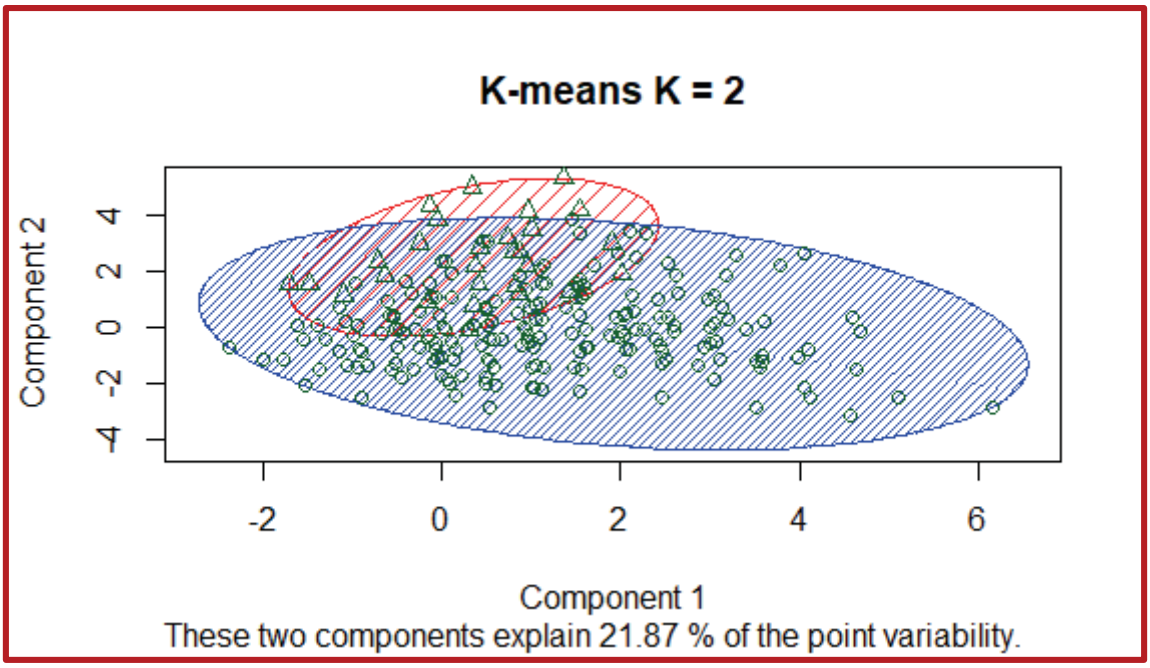

FIGURA.0.2.7. GRÁFICA DE DOS CLÚSTERS CON EL ALGORITMO *K-MEANS*.

Algoritmo *K-Nearest Neighbors*

El algoritmo *k-nearest neighbors* (k vecinos más cercanos) o *k-nn*, como se mencionó, es un algoritmo supervisado, que busca clasificar objetos en este caso específico los *Tweets*, para la identificación de modelos afines. Los *Tweets* se clasifican basados en un conjunto de entrenamiento, suministrando *Tweets* clasificados previamente.

Para comenzar la preparación de los *Tweets* se requirió contenerlos en una matriz llamada documento-término, que establece que la frecuencia de los términos que ocurren en una colección de documentos, este procedimiento puede observarse en la Figura.0.3.0.

```
#***********Read myCorpus, convert to DocumentTermMatrix*************
docs = myCorpus
dtm  = DocumentTermMatrix(docs)
```

FIGURA.0.3.0 PROCEDIMIENTO DE TRANSFORMACIÓN DE CORPUS A MATRIZ DOCUMENTO-TÉRMINO.

Para la manipulación de la matriz documento-término fue necesario contenerla en una data *frame,* lo que permitió asignarle la clasificación a cada Tweet que fue removido en la sección de conjunto de datos. Por convención se modificó el nombre de la columna que contiene la clasificación de cada *Tweet* por *category*.

Es necesario dividir nuestro conjunto de datos en dos categorías, cómo se mencionó anteriormente, conjunto de entrenamiento y conjunto de prueba. También por convención se establece que el conjunto de entrenamiento sea del setenta por ciento del total del conjunto de datos existentes, mientras que el conjunto de prueba sea el restante treinta por ciento. Es trascendental para los resultados que se generen que el conjunto de datos tanto de entrenamiento como de prueba se tomen de forma aleatoria con la finalidad de eludir redundancia en la clasificación de los *Tweets* en el conjunto de entrenamiento que provocaría una errónea clasificación de los *Tweets* en el conjunto de prueba. En la Figura.0.3.1. se ejemplifica el procedimiento de partición del conjunto de datos en las dos categorías mencionadas.

```
#**********Split data by rownumber into two portions**************
train = sample(nrow(mat.df), ceiling(nrow(mat.df) * .70))
test  = (1:nrow(mat.df))[- train]
```

FIGURA.0.3.1. PROCEDIMIENTO DE SEPARACIÓN DEL CONJUNTO DE DATOS EN ENTRENAMIENTO Y PRUEBA.

En el modelo de los datos que se tienen se eliminó la categoria o clasificaciónclasificación de cada *Tweet*, para poder manipular los *Tweets,* con el algoritmo *k-nn*.

Para la ejecución del algoritmo en el entorno de Rstudio se requirio el conjunto de datos de entrenamiento, de prueba y la categoria en la cual se clasificarón los *Tweets,* asi como el número de vecinos considerados. En la Figura.0.3.2. se observa el procedimiento que se llevó acabo para la ejecucion de este algoritmo, consideran un número de vecinos igual a tres.

```
#****Create model: training set, test set, training set classifier k = 3*****
knn.pred3 = knn(modeldata[train, ], modeldata[test, ], cl[train], k = 3)
knn.pred3
```

FIGURA.0.3.2. PROCEDIMIENTO DE EJECUCIÓN DEL ALGORITMO K-NN CON EL NÚMERO DE TRES VECINOS.

La exactitud del algoritmo, al clasificar los *Tweets,* es un factor sustancial a tomar en cuenta para medir la eficiencia que presenta el algoritmo ante el conjunto de datos que se le suministra. Por ello, es importante conocer lo que se denomina como *Accuracy*, de la ejecucion del algoritmo con el conjunto de prueba y entramiento así mismo como la clasificaciónclasificación de los *Tweets* y el número de vecinos que se consideró. En la Figura.0.3.3. se muestra el *Accuracy*, exactitud, considerando tres vecinos cercanos.

```
> #************************Accuracy************************************
> (accuracy3 = sum(diag(conf.mat3))/length(test) * 100)
[1] 83.40426
```

FIGURA.0.3.3. EXACTITUD DEL ALGORITMO K-NN CON EL NÚMERO DE TRES VECINOS.

Siguiendo con un estandar de las variables solicitadas por cada algoritmo, se continuó estableciendo el número de vecinos cercanos a seis, lo cual la premisa nos dice que al aumentar el número de k vecinos mas cercanos, se aminora la alteracion al clasificar los *Tweets* que pueda provocar el ruido, es decir, que el conjunto de datos de entrenamiento no elecciona correctamente el algoritmo, provocando asi una inexacta clasificación. En la Figura.0.3.4. se señala la práctica que se llevó a cabo al ejecutar el algoritmo *k-nn* con un número de seis vecinos.

```
#****Create model: training set, test set, training set classifier k = 6*****
knn.pred6 = knn(modeldata[train, ], modeldata[test, ], cl[train], k = 6)
```

FIGURA.0.3.4. PROCEDIMIENTO DE EJECUCIÓN DEL ALGORITMO K-NN CON EL NÚMERO DE SEIS VECINOS.

Teniendo en memoria el resultado que se obtuvo al correr el algoritmo *k-nn,* se calcula la exatitud con la que el algoritmo clasificó los *Tweets.* En la Figura.0.3.5. se puede corroborar el procedimiento para calcular la exactitud y el resultado obtenido.

```
> (accuracy6 = sum(diag(conf.mat6))/length(test) * 100)
[1] 83.19149
```

FIGURA.0.3.5. EXACTITUD DEL ALGORITMO K-NN CON EL NÚMERO DE SEIS VECINOS.

Refutando la premisa que se consideraba, al estimar que a mayor número de vecinos cercanos, mayor seria la exactitud con la cual se clasificarian los *Tweets*. Esto se debió, principalmente, a que existe la presencia de singularidades no necesarias para considerarse en el conjunto de datos de entrenamiento, originando en la clasificación una exactitud menor, que al considerarse tres vecinos cercanos.

Dado lo anterior, se determinó disminuir el número de *k* vecinos mas cercanos, teniendo así, 2 vecinos. Se visualizó que al aumentar el número de vecinos de tres a seis, la exactitud disminuyo, por lo que con dos vecinos, se estimó que la exactitud del algoritmo aumentará. En la Figura.0.3.6. se contempla el procedimiento que se llevo acabo, para ejecutar el algoritmo con dos vecinos mas cercanos.

```
#****Create model: training set, test set, training set classifier k = 2*****
knn.pred2 = knn(modeldata[train, ], modeldata[test, ], cl[train], k = 2)
```

FIGURA.0.3.6. PROCEDIMIENTO DE EJECUCIÓN DEL ALGORITMO K-NN CON EL NÚMERO DE DOS VECINOS.

Con el resultado almacenado en memoria en la variable llamada *conf.mat2,* se calculó la exactitud del algoritmo al clasificar los *Tweets* teniendo en cuenta dos vecinos, esta exactitud se calculó obteniendo la suma de la diagonal de la matriz de confusión dividida entre el número del conjunto de prueba, multiplicado por cien. En la Figura.0.3.7. se analiza a detalle el procedimiento del calculo de la *accuracy,* en comparación con la exactitud de del algotimo *knn* con tres y seis vecinos, la exactitud

disminuyo, teniendo de esta forma un rendimiento menor al clasificar los *Tweets* con dos vecinos mas cercanos.

```
> #************************Accuracy**************************************
> (accuracy2 = sum(diag(conf.mat2))/length(test) * 100)
[1] 82.87234
```

FIGURA.0.3.7. EXACTITUD DEL ALGORITMO K-NN CON EL NÚMERO DE DOS VECINOS.

Algoritmo *Naïve Bayes*

El algoritmo *Naïve Bayes* es el tercer, y último algoritmo que se analizó, esté algoritmo es un algoritmo de clasificación. Éste realiza la clasificación de manera supervisada, teniendo en cuenta, la probabilidad de un evento A dado un evento B.

Cuando se crea un modelo de clasificación se requiere un diagnóstico integral, que cuantifique la eficiencia del algoritmo. Con este objetivo se divide el conjunto de elementos, en *train* o bien entrenamiento y *test*, prueba. El conjunto de entrenamiento determina las probabilidades condicionales de cada una de las palabras contenidas, con la finalidad principal de examinar la clasificación en el conjunto de prueba.

Usualmente se divide el conjunto de elementos en conjunto de entrenamiento y conjunto de pruebas con un porcentaje de los setenta porcientos de entrenamiento y treinta porcientos de prueba.

```
#********Split data by rownumber into two equal portions********
#train = sample(nrow(df), ceiling(nrow(df) * .70))
#test  = (1:nrow(df))[- train]

df.train = df[1:2195,]
write.csv(df.train , file = "dataFrameTrain.csv")
df.test  = df[2195:3135,]
write.csv(df.test , file  = "dataFrameTest.csv")
```

FIGURA.0.4.0 PROCEDIMIENTO DE SEPARACIÓN DEL CONJUNTO DE DATOS EN ENTRENAMIENTO Y PRUEBA DEL DATA FRAME.

El algoritmo *Naïve Bayes* requiere de la matriz de término documento, la cual tambien se necesitó dividir entre un conjunto de entrenamiento y un conjunto de prueba como se muestra en la Figura.0.4.1.

```
tdm.train = tdm[1:2195,]
dim(tdm.train)
tdm.test  = tdm[2195:3135,]
dim(tdm.test)
```

FIGURA.0.4.1. PROCEDIMIENTO DE SEPARACIÓN DE LA MATRIZ TÉRMINO DOCUMENTO EN ENTRENAMIENTO Y PRUEBA.

El corpus creado en la sección de conjunto de datos, tambien es un elemento que se requirió para el funcionamiento correcto del algoritmo *Naïve Bayes*, este corpus contiene los *Tweets* que se preoprocesaron, para eliminar elementos incensarios para el procesamiento de estos *Tweets* con el algoritmo. En la Figura.0.4.2. se muestra el procedimiento que se utilizó para separar el corpus, en un corpus de prueba y un corpus de entrenamiento.

```
myCorpus.train = myCorpus[1:2195]
myCorpus.test  = myCorpus[2195:3135]
```

FIGURA.0.4.2. PROCEDIMIENTO DE SEPARACIÓN DEL CORPUS EN ENTRENAMIENTO Y PRUEBA.

Se requrió conocer los términotérminos frecuentes de la matriz término del documento de entrenamiento, teniendo en cuenta una frecuencia baja de diez, veinte y treinta elementos. En la Figura.0.4.3. se visualiza el procedimiento que se llevo acabo para conocer los términos frecuentes de la matriz terrmino documento.

```
#**************FrecuencyTerms in train************************
terFrecuency10 = findFreqTerms(tdm.train, lowfreq = 10)
write.csv(terFrecuency10 , file = "terFrecuency10.csv")

terFrecuency20 = findFreqTerms(tdm.train, lowfreq = 20)
write.csv(terFrecuency20 , file = "terFrecuency20.csv")

terFrecuency30 = findFreqTerms(tdm.train, lowfreq = 30)
write.csv(terFrecuency30 , file = "terFrecuency30.csv")
```

FIGURA.0.4.3. TÉRMINOS FRECUENTES EN LA MATRIZ TÉRMINO DOCUMENTO DE ENTRENAMIENTO.

Se requirió analizar los elementos del copus de entrenamiento en acotación de los términos frecuentes tasados en la Figura.0.4.3. Cada una de estas matrices se almacenan de acuerdo a los términos frecuentes, es decir, diez, veinte y treinta. En la Figura.0.4.4. se ejemplifica la metodología que se realizó para la conversión del corpus de entrenamiento.

```
tdm.train.nb10 = DocumentTermMatrix(myCorpus.train,
                                    control = list(dictionary = terFrecuency10))
dim(tdm.train.nb10)

tdm.train.nb20 = DocumentTermMatrix(myCorpus.train,
                                     control = list(dictionary = terFrecuency20))
dim(tdm.train.nb20)

tdm.train.nb30 = DocumentTermMatrix(myCorpus.train,
                                    control = list(dictionary = terFrecuency30))
dim(tdm.train.nb30)
```

FIGURA.0.4.4. CONVERSIÓN DEL CORPUS EN DOCUMENTO TÉRMINO MATRIZ TOMANDO EN CUENTA LA FRECUENCIA DE TÉRMINOS.

Se llevó acabo el mismo procedimiento, para la conversión del corpus de prueba en una matriz de documento término en acotación de los términos frecuentes.

Fue ineludible la conversión de la matriz documento término en expresión numérica, esto para que el algoritmo Naïve Bayes fuese capaz de procesar de forma correcta cada uno de los elementos. En la Figura.0.4.5. se ejemplifica la función que se eleboró para realizar la conversión de la matriz tanto de entrenamiento como de prueba.

```
#**************Convert type to number*********************************
convert_count = function(x) {
  y = ifelse(x > 0, 1,0)
  y = factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}
```

FIGURA.0.4.5. FUNCIÓN PARA CONVERTIR LA MATRIZ DOCUMENTO TÉRMINO EN EXPRESIÓN NUMÉRICA.

Se adapto el modelo de los datos usando el conjunto de entrenamiento y se proporciono la variable objetivo del *data frame* de entrenamiento, de tal forma que esta variable fue la variable dependiente. Con la finalidad de realizar predicciones a base de nuestro modelo otorgado por el algoritmo *Naïve Bayes*, se utilizó la funcion *predict*, la cual contiene como argumentos el modelo de datos arrojado por *Naïve Bayes* y el conjunto de prueba.

Se analizó la exactitud del algoritmo, con la elaboración de tablas que contienen las predicciones de los datos contra poniendolos con los datos actuales. De tal forma que se pudo realizar la visualizacion grafica de los datos vaticinados. En la Figura.0.4.6. se visualiza el procedimiento que se desarrolló para pronosticar de forma correcta la clasificación de los *Tweets,* teniendo en cuenta el conjunto de datos de frecuencia baja, se considera frecuencia baja si se tiene una frecuencia igual a diez.

```
#**************NaiveBayes10*******************************************************
system.time( classifier <- naiveBayes(trainNB10, df.train$type, laplace = 1) )
# Use the NB classifier we built to make predictions on the test set.
system.time( pred10 <- predict(classifier, newdata = testNB10) )
# Create a truth table by tabulating the predicted class labels with the actual class labels
tablePredAct10 = table("Predictions"= pred10,  "Actual" = df.test$type)
write.csv(tablePredAct10 , file = "tablePredAct10.csv")
plot(pred10,main="Predictions 10",col = colors,las=1)
crossTable10 = CrossTable(pred10,df.test$type)
crossTable10 = data.frame(crossTable10)
write.csv(crossTable10 , file = "crossTable10.csv")
conf.mat10 = confusionMatrix(pred10, df.test$type)
```

FIGURA.0.4.6. PROCEDIMIENTO DEL ALGORITMO *NAÏVE BAYES* CON UNA FRECUENCIA BAJA, IGUAL A DIEZ.

La visualización gráfica de la predicción con frecuencia baja, es decir, igual a diez, se puede observar en la Figura.0.4.7.

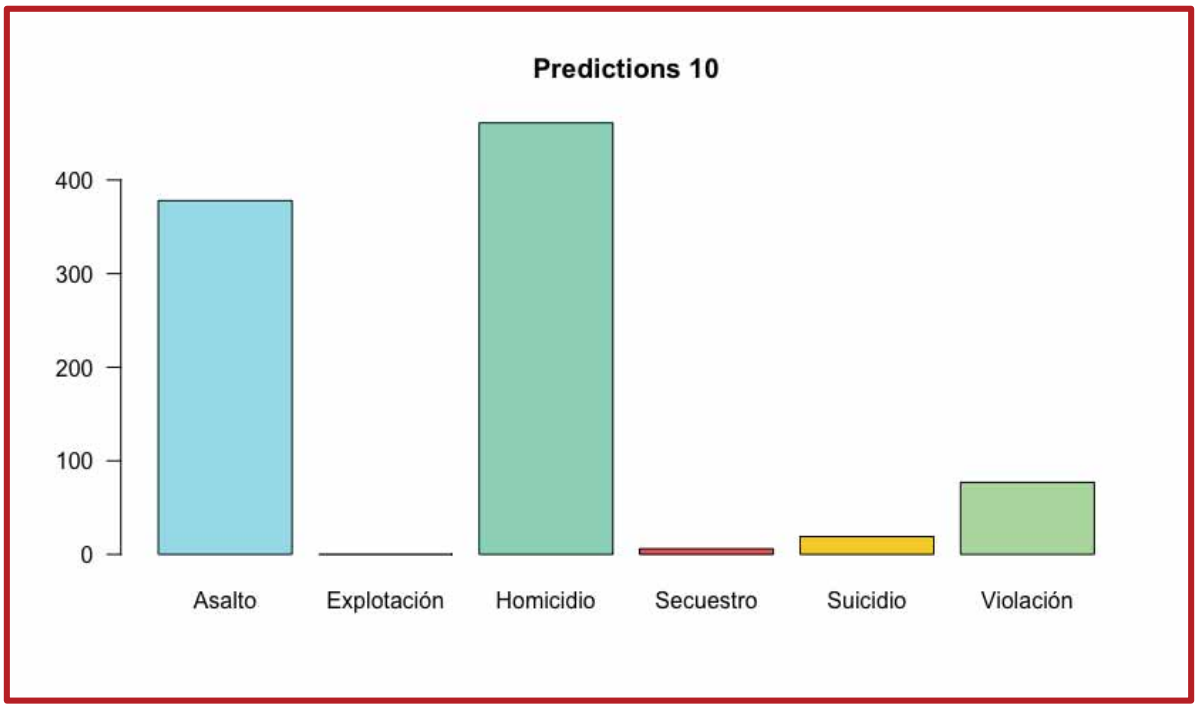

FIGURA.0.4.7. HISTOGRAMA DE LA PREDICCIÓN CON UNA FRECUENCIA BAJA, IGUAL A DIEZ.

El algoritmo se ejecutó con las tres variantes, es decir, con una frecuencia baja de diez, veinte y treinta, con la finalidad de obtener diferentes resultados de la clasificación de los *Tweets*, teniendo así un panorama integral del funcionamiento y precisión del algoritmo. En la Figura.0.4.8. se muestra a detalle la metodología que se llevo a cabo para la ejecución del algoritmo con una frecuencia baja de veinte.

```
#**************NaiveBayes20**********************************************************
system.time( classifier = naiveBayes(trainNB20, df.train$type, laplace = 1) )
# Use the NB classifier we built to make predictions on the test set.
system.time( pred20 = predict(classifier, newdata = testNB20) )
# Create a truth table by tabulating the predicted class labels with the actual class labels
tablePredAct20 = table("Predictions" = pred20,  "Actual" = df.test$type)
write.csv(tablePredAct20 , file = "tablePredAct20.csv")
plot(pred20,main = "Predictions 20",col = colors,las=1)
crossTable20 = CrossTable(pred20,df.test$type)
crossTable20 = data.frame(crossTable20)
write.csv(crossTable20 , file = "crossTable20.csv")
conf.mat20 = confusionMatrix(pred20, df.test$type)
```

FIGURA.0.4.8. PROCEDIMIENTO DEL ALGORITMO *NAÏVE BAYES* CON UNA FRECUENCIA BAJA, IGUAL A VEINTE.

El histograma proporciona la representación gráfica de la variable de clasificación en barras, estas barras otorgan la frecuencia de los valores o categorias de la clasificación de los *Tweest.* En la Figura.0.4.9. se visualiza el histograma con una frecuencia baja igual a veinte.

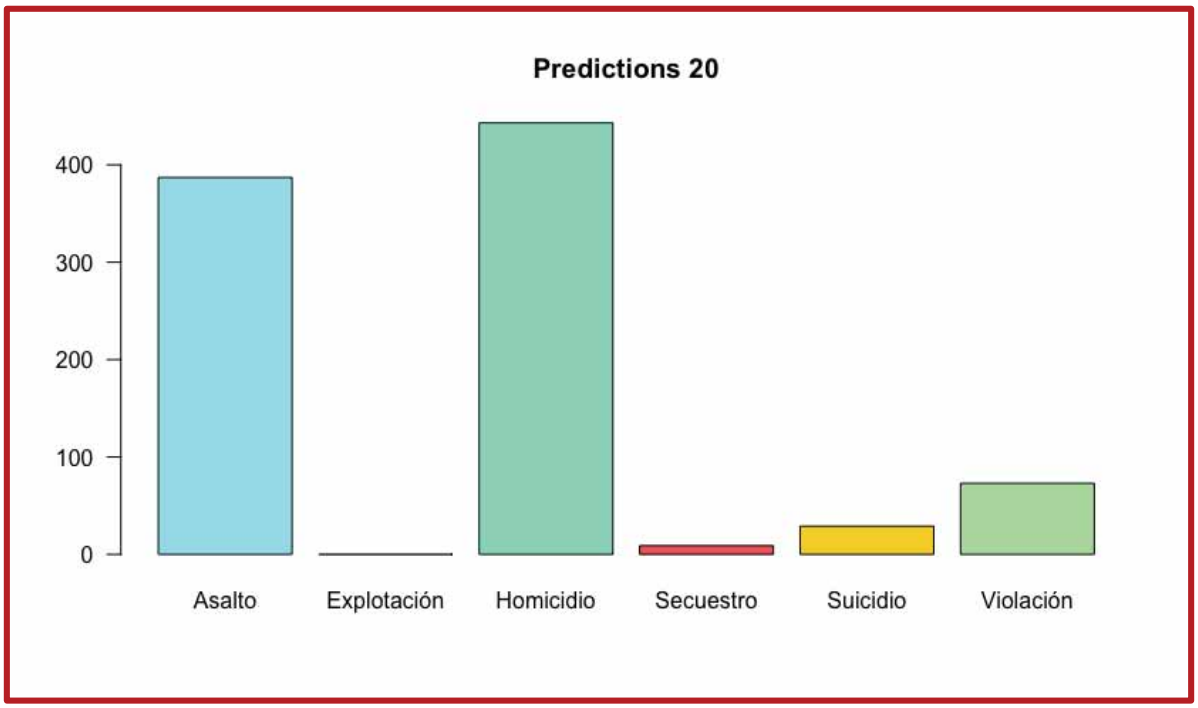

FIGURA.0.4.9. HISTOGRAMA DE LA PREDICCIÓN CON UNA FRECUENCIA BAJA, IGUAL A VEINTE.

El conjunto de datos de entrenamiento de los *Tweets* y la variable objetivo (clasificación), se interpretan en una matriz de frecuencia, con una frecuencia baja igual a treinta, estos argumentos se emplean para informar al algoritmo *Naïve Bayes* de la estructura y modelo de los datos, para que la ecuación *Bayesiana* calcule la probabilidad de clasificación para cada *Tweet,* la propababilidad mas alta sera el resultado de la predicción. En la Figura.0.5.0. se puede observar la metodología que se empleo para resolver la prediccion de la clasificación de los *Tweets.*

```
#**************NaiveBayes30*********************************************************
system.time( classifier30 = naiveBayes(trainNB30, df.train$type, laplace = 1) )
# Use the NB classifier we built to make predictions on the test set.
system.time( pred30 = predict(classifier, newdata = testNB30) )
# Create a truth table by tabulating the predicted class labels with the actual class labels
tablePredAct30 = table("Predictions"= pred30,  "Actual" = df.test$type)
write.csv(tablePredAct30 , file = "tablePredAct30.csv")
plot(pred30,main = "Predictions 30",col = colors,las=1)
crossTable30 = CrossTable(pred30,df.test$type)
crossTable30 = data.frame(crossTable30)
write.csv(crossTable30 , file = "crossTable30.csv")
conf.mat30 = confusionMatrix(pred30, df.test$type)
```

FIGURA.0.5.0. PROCEDIMIENTO DEL ALGORITMO *NAÏVE BAYES* CON UNA FRECUENCIA BAJA, IGUAL A TREINTA.

El beneficio y rendimiento del histograma tiene una relación directa con la habilidad de proporcionar de forma visual, los datos numéricos estadísticos del pronostico de la clasificación de los *Tweets.*En la Figura.0.5.1. se muestra el histograma derivado del pronostico realizado con una frecuencia baja igual a treinta.

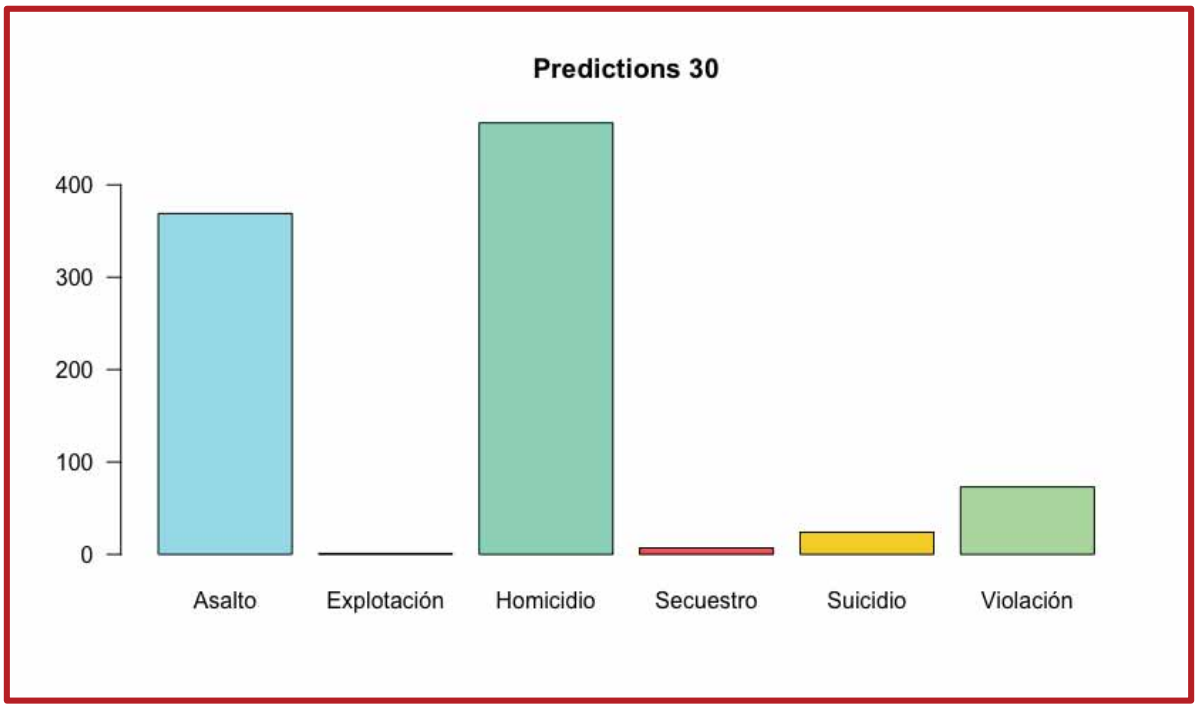

FIGURA.0.5.1. HISTOGRAMA DE LA PREDICCIÓN CON UNA FRECUENCIA BAJA, IGUAL A TREINTA.

Resultados

Algoritmo *K-means*

Como se mencionó, en este documento, el algoritmo *K-means* es un algoritmo de clasificación, ya que agrupa los elementos de un conjunto de datos en una clase o variable objetivo, para poder observar y analizar la exactitud del comportamiento del algoritmo se utilizó una matriz de confusión, en esta matriz tiene por columnas las variables objetivo que se vaticinaron y por filas las variables objetivos autenticas y probadas. Cada una de las celdas que conforman la matriz es el calculo de los elementos de una clase verdadera clasificados en las clases predichas.

La matriz de confusión tiene la capacidad de mostrar si las clases verdaderas se encuentran equivocadas al clasificar los elementos del conjunto de datos. Las clasificaciones totalmente exactas se distinguen debido a que todas las casillas de la matriz se encuentran en cero salvo la diagonal. En contra parte, una matriz confusa o no clasificada correctamente contendra valores grandes situados fuera de la diagonal. En la Figura.0.6.0 se muestra la matriz de confusión del algoritmo *K-means* con un número de *clústers* igual a tres. Se percibe que en el clúster número uno se clasificaron, ochocientos cincuenta y dos elementos(*Tweets*) de la clase *Asalto*, veinticinco *Tweets* de la clase *Explotación,* cincuenta y nueve de la clase *Secuestro,* ciento dos de la clase *Suicidio* y docientos cincuenta y uno de la clase *Violación,* siendo el clúster número uno con mayor elementos agrupados dentro de el.

Cell Contents
N
Chi-square contribution
N / Row Total
N / Col Total
N / Table Total

Total Observations in Table: 3135

kmeansResult3$cluster	df$type: Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación	Row Total
1	852	25	1029	59	102	251	2318
	0.083	1.735	4.892	0.236	0.698	15.472	
	0.368	0.011	0.444	0.025	0.044	0.108	0.739
	0.747	0.962	0.690	0.694	0.803	0.947	
	0.272	0.008	0.328	0.019	0.033	0.080	
2	119	0	406	8	6	10	549
	32.683	4.553	80.409	3.185	11.859	28.562	
	0.217	0.000	0.740	0.015	0.011	0.018	0.175
	0.104	0.000	0.272	0.094	0.047	0.038	
	0.038	0.000	0.130	0.003	0.002	0.003	
3	170	1	56	18	19	4	268
	53.829	0.673	40.064	15.855	6.108	15.360	
	0.634	0.004	0.209	0.067	0.071	0.015	0.085
	0.149	0.038	0.038	0.212	0.150	0.015	
	0.054	0.000	0.018	0.006	0.006	0.001	
Column Total	1141	26	1491	85	127	265	3135
	0.364	0.008	0.476	0.027	0.041	0.085	

FIGURA.0.6.0 MATRIZ DE CONFUSIÓN CON UN NÚMERO DE CLÚSTER IGUAL A TRES.

Dado que el número de clases objetivo son seis, se decidió instaurar seis *clústers* para clasificar los *Tweets* en estos, de esta forma se tiene la capacidad de visualizar si la clase objetivo y los *clústers* coinciden con el número de elementos clasificados. Es importante destacar que la principal finalidad es analizar el comportamiento del algoritmo, sometido al conjunto de datos específico. Este conjunto de datos evidencia la situación de seguridad que se vive en la Ciudad de México y Área Metropolitana.

En la ejecución del algoritmo *K-means* con un número de *clústers* igual a seis, se observó que el primer clúster continuó siendo el que contiene los mayores elementos agrupados dentro de él. La inspección de esta afirmación se dio gracias a la llamada matriz de confusión, con la cual se llega a la conclusión que resulta ser una agrupación malograda, debido a que existe un empalme entre las variables objetivo y los *clústers,* es decir, no se encuentra una matriz con casillas en cero exceptuando su diagonal. En la Figura.0.6.1. se visualiza la matriz de confusión con seis *clústers.*

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3135
```

kmeansResult6$cluster	df$type: Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación	Row Total
1	797	25	775	59	83	228	1967
	9.187	4.626	27.537	0.602	0.138	22.919	
	0.405	0.013	0.394	0.030	0.042	0.116	0.627
	0.699	0.962	0.520	0.694	0.654	0.860	
	0.254	0.008	0.247	0.019	0.026	0.073	
2	6	0	406	0	0	0	412
	138.190	3.417	225.176	11.171	16.690	34.826	
	0.015	0.000	0.985	0.000	0.000	0.000	0.131
	0.005	0.000	0.272	0.000	0.000	0.000	
	0.002	0.000	0.130	0.000	0.000	0.000	
3	125	0	92	7	10	8	242
	15.478	2.007	4.634	0.029	0.004	7.585	
	0.517	0.000	0.380	0.029	0.041	0.033	0.077
	0.110	0.000	0.062	0.082	0.079	0.030	
	0.040	0.000	0.029	0.002	0.003	0.003	
4	61	1	46	9	3	4	124
	5.580	0.001	2.854	9.455	0.815	4.008	
	0.492	0.008	0.371	0.073	0.024	0.032	0.040
	0.053	0.038	0.031	0.106	0.024	0.015	
	0.019	0.000	0.015	0.003	0.001	0.001	
5	114	0	53	8	9	2	186
	31.672	1.543	14.215	1.734	0.285	11.977	
	0.613	0.000	0.285	0.043	0.048	0.011	0.059
	0.100	0.000	0.036	0.094	0.071	0.008	
	0.036	0.000	0.017	0.003	0.003	0.001	
6	38	0	119	2	22	23	204
	17.696	1.692	4.979	2.254	22.831	1.921	
	0.186	0.000	0.583	0.010	0.108	0.113	0.065
	0.033	0.000	0.080	0.024	0.173	0.087	
	0.012	0.000	0.038	0.001	0.007	0.007	
Column Total	1141	26	1491	85	127	265	3135
	0.364	0.008	0.476	0.027	0.041	0.085	

FIGURA.0.6.1. MATRIZ DE CONFUSIÓN CON UN NÚMERO DE CLÚSTER IGUAL A SEIS.

El análisis de los *clústers* destaca la agrupación de los elementos, esta relación identificó el umbral que separa los grupos. Cada uno de los elementos se agrupó en función de su distancia a todos los centros de cada *clúster*, teniendo así que el *clúster* contiguo se consideró el más afín. En la Fugura.0.6.2. se visualiza la matriz de confusión con un número de dos *clústers,* siendo que no se obtuvieron los resultados deseados con un número de *clústers* igual a seis, se siguen aglomerando los *Tweets* en el primer *clúster* originando que la clasificación resulte siendo la menos adecuada para proporcionar una clasificación certera y exacta.

```
   Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  3135
```

kmeansResult2$cluster	df$type Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación	Row Total
1	845	26	1489	85	127	265	2837
	34.063	0.260	14.470	0.849	1.268	2.646	
	0.298	0.009	0.525	0.030	0.045	0.093	0.905
	0.741	1.000	0.999	1.000	1.000	1.000	
	0.270	0.008	0.475	0.027	0.041	0.085	
2	296	0	2	0	0	0	298
	324.287	2.471	137.756	8.080	12.072	25.190	
	0.993	0.000	0.007	0.000	0.000	0.000	0.095
	0.259	0.000	0.001	0.000	0.000	0.000	
	0.094	0.000	0.001	0.000	0.000	0.000	
Column Total	1141	26	1491	85	127	265	3135
	0.364	0.008	0.476	0.027	0.041	0.085	

FIGURA.0.6.2. MATRIZ DE CONFUSIÓN CON UN NÚMERO DE CLÚSTER IGUAL A DOS.

En términos estadísticos la clasificación *k-means,* resulta ser una clasificación a base de *clústers*, que no alcanza a cumplir con los estándares de exactitud y precisión requeridos para dictaminar una clasificación exitosa, dentro del conjunto de datos proporcionado. No obstante, teniendo que elegir dentro del algoritmo *k-means* un modelo que, mínimamente, cumpla con los requerimientos mencionados se elige el algoritmo *K-means* con un número igual a seis *clústers.*

Algoritmo *K-nearest neighbors*

La exactitud dentro de la clasificación de los elementos, puede ser ilusoria e irreal, por lo cual se empleó la matriz de confusión, que determina la forma exacta en la cual se clasificaron cada uno de los elementos dentro de una clase objetivo y si este procedimiento se llevó o no a cabo de manera exitosa. Llegados a este punto, cabe destacar que esté algoritmo resulto ser el algoritmo con mayor tiempo consumido durante su ejecución, dado que la matriz que se emplea de entrenamiento y de prueba es bastante extensa. En la Figura.0.7.0. se muestra la matriz de confusión teniendo en cuenta un número de vecinos igual a tres, en donde se contempla una clasificación con ciertos elementos clasificados en variables objetivo distintas a las originales, siendo que la variable objetivo *Homicidio* cuenta con los veintiún elementos clasificados en la categoría de *Homicidios,* en contra parte tan solo diez elementos clasificados de forma correcta en la clase que le corresponde.

	Actual					
Predictions	Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación
Asalto	250	0	7	2	4	1
Explotación	0	3	0	0	0	0
Homicidio	83	3	450	8	21	18
Secuestro	0	0	0	14	0	0
Suicidio	2	0	4	0	10	0
Violación	2	0	0	0	1	57

FIGURA.0.7.0. MATRIZ DE CONFUSIÓN CON UN NÚMERO DE VECINOS IGUAL A TRES.

Se estima que a mayor número de vecinos cercanos mayor será la exactitud que el algoritmo toma para realizar su predicción de la clasificación de los *Tweets*, por lo que se consideraron seis vecinos cercanos Sin embargo, esto no resulto ser cierto, debido a que se obtuvo una exactitud menor, en comparación de tener tres vecinos cercanos, esta situación se puede corroborar en la Figura.0.7.1. donde se ejemplifica la clasificación de los elementos, en cada una de las variables objetivo, se visualiza que la variable objetivo *Suicidio* aumento el número de error, ya que fueron clasificados un mayor número de *Tweets* en una variable distinta.

	Actual					
Predictions	Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación
Asalto	253	0	4	2	2	1
Explotación	0	3	0	0	0	0
Homicidio	82	3	453	12	26	19
Secuestro	0	0	0	10	0	0
Suicidio	1	0	4	0	7	0
Violación	1	0	0	0	1	56

FIGURA.0.7.1. MATRIZ DE CONFUSIÓN CON UN NÚMERO DE VECINOS IGUAL A SEIS.

Por último, se decidió disminuir el número de vecinos cercanos a tan solo dos, con la principal finalidad de poder aumentar la exactitud del algoritmo al clasificar los *Tweets*, en la Figura.0.7.2. se muestra la matriz de confusión del algoritmo *K-nearest Neighbors,* con un número total de vecinos cercanos igual a dos. Se observa que el número de elementos clasificados de forma incorrecta disminuyo en comparación a tomar seis vecinos cercanos. Sin embargo, aumento en comparación de tomar tan solo tres vecinos cercanos.

Esto concluye que la mejor metodología para el modelo de datos que se le suministro al algoritmo, es tomar en cuenta tres vecinos más cercanos, ya que su exactitud es destacable haciendo la comparación con los modelos de dos y seis vecinos respectivamente. Como se mencionó este algoritmo tiene la gran desventaja en el tiempo de ejecución, dado que procesa cada elemento de una matriz de término documento ocasionando que los recursos que requiere para la ejecución satisfactoria del algoritmo aumenten.

	Actual					
Predictions	Asalto	Explotación	Homicidio	Secuestro	Suicidio	Violación
Asalto	248	0	12	1	3	1
Explotación	0	3	0	0	0	0
Homicidio	82	3	441	7	21	15
Secuestro	3	0	4	16	0	0
Suicidio	3	0	3	0	11	0
Violación	1	0	1	0	1	60

FIGURA.0.7.2. MATRIZ DE CONFUSIÓN CON UN NÚMERO DE VECINOS IGUAL A DOS.

Algoritmo *Naïve Bayes*

Una matriz de confusión proporciona la capacidad de visualizar una representación tabular de la ejecución del algoritmo *Naïve Bayes*, en la cual se ejemplifica el desempeño de este algoritmo. La columna de la matriz simboliza la predicción de la clase o variable objetivo, y las filas corresponden a los elementos de referencia.

Lo que es importante visualizar son las casillas donde se entrelazan los valores vaticinados contra los valores de referencia. De tal forma que la cantidad que se representa en cada casilla, pertenece a la proporción de los *Tweets* clasificados de forma correcta en la variable objetivo o clase que pertenece (Asalto, Explotación, Homicidio, Secuestro, Suicidio, Violación).

El conjunto de datos (*Tweets*) tiene una extensión de tres mil cientos treinta y cinco, de los cuales la categoría de *Asalto* consta con mil ciento cuarenta y uno, *Explotación* tan solo veintiséis, *Homicidio* mil cuatrocientos noventa y uno, *Secuestro* ochenta y cinco, *Suicidio* ciento veintisiete y *Violación* doscientos sesenta y cinco.

Teniendo estas cifras en mente se posee la matriz de confusión, donde en la matriz de confusión con frecuencia baja igual a diez tienen los elementos cruzados de las variables objetivo. Es decir, los valores de referencia y los valores de predicción. La variable objetivo *Asalto* tiene mil cientos cuarenta uno *Tweets* clasificados en esta categoría, no obstante, con el algoritmo *Naïve Bayes* tan solo cuenta con trecientos cuarenta y siete *Tweets* y treinta y siete elementos que fueron clasificados en una categoría diferente.

En la Figura.0.8.0. se visualiza la matriz de confusión con una frecuencia baja igual a diez, en la cual se advierte que el algoritmo tiene una exactitud de clasificación de los elementos de ochenta y siete punto cero cuatro por ciento (87.04%), se considera una tasa de exactitud aceptable. Sin embargo, se establece que para propicitos de sondeos rigurosos y precisos esta predicción cataloga como incierta.

```
> conf.mat10
Confusion Matrix and Statistics

            Reference
Prediction    Asalto Explotación Homicidio Secuestro Suicidio Violación
  Asalto         347           0        27         2        1         1
  Explotación      0           0         0         0        0         0
  Homicidio       15           5       383        18       15        25
  Secuestro        5           0         0         1        0         0
  Suicidio         1           0         4         0       14         0
  Violación        2           1         0         0        0        74

Overall Statistics

               Accuracy : 0.8704
                 95% CI : (0.8472, 0.8912)
    No Information Rate : 0.44
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7899
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Asalto Class: Explotación Class: Homicidio Class: Secuestro Class: Suicidio Class: Violación
Sensitivity                 0.9378           0.000000           0.9251         0.047619         0.46667          0.74000
Specificity                 0.9457           1.000000           0.8520         0.994565         0.99451          0.99643
Pos Pred Value              0.9180                NaN           0.8308         0.166667         0.73684          0.96104
Neg Pred Value              0.9591           0.993624           0.9354         0.978610         0.98265          0.96991
Prevalence                  0.3932           0.006376           0.4400         0.022317         0.03188          0.10627
Detection Rate              0.3688           0.000000           0.4070         0.001063         0.01488          0.07864
Detection Prevalence        0.4017           0.000000           0.4899         0.006376         0.02019          0.08183
Balanced Accuracy           0.9418           0.500000           0.8886         0.521092         0.73059          0.86822
```

FIGURA.0.8.0. MATRIZ DE CONFUSIÓN CON UNA FRECUENCIA BAJA IGUAL A DIEZ.

Los resultados proporcionados por el algoritmo *Naïve Bayes* con una frecuencia baja igual a veinte, proporcionan mejores resultados en comparación con el mismo algoritmo con una frecuencia baja igual a diez, no obstante no son los resultados deseados debido a que tan solo mejoran en un cero punto diez porciento (0.10%) lo cual no otorga la certeza de una clasifación correcta en términos de un analisis inequivoco. En la Figura.0.8.1. se puede observar que en la variable objetivo *Explotación* tiene clasificados tan solo cinco *Tweets* en comparación con los veintiseis elementos que cuenta originalmente la variable objetivo, esto indica que no representa una prediccion confiable.

```
> conf.mat20
Confusion Matrix and Statistics

            Reference
Prediction    Asalto Explotación Homicidio Secuestro Suicidio Violación
  Asalto         355           0        28         2        2         0
  Explotación      0           0         0         0        0         0
  Homicidio       12           5       372        14       11        29
  Secuestro        1           0         3         5        0         0
  Suicidio         1           0        11         0       17         0
  Violación        1           1         0         0        0        71

Overall Statistics

               Accuracy : 0.8714
                 95% CI : (0.8483, 0.8921)
    No Information Rate : 0.44
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7932
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Asalto Class: Explotación Class: Homicidio Class: Secuestro Class: Suicidio Class: Violación
Sensitivity                 0.9595           0.000000           0.8986         0.238095         0.56667          0.71000
Specificity                 0.9440           1.000000           0.8653         0.995652         0.98683          0.99762
Pos Pred Value              0.9173                NaN           0.8397         0.555556         0.58621          0.97260
Neg Pred Value              0.9729           0.993624           0.9157         0.982833         0.98575          0.96659
Prevalence                  0.3932           0.006376           0.4400         0.022317         0.03188          0.10627
Detection Rate              0.3773           0.000000           0.3953         0.005313         0.01807          0.07545
Detection Prevalence        0.4113           0.000000           0.4708         0.009564         0.03082          0.07758
Balanced Accuracy           0.9517           0.500000           0.8819         0.616874         0.77675          0.85381
```

FIGURA.0.8.1. MATRIZ DE CONFUSIÓN CON UNA FRECUENCIA BAJA IGUAL A VEINTE.

Teniendo en cuenta los dos resultados obtenidos con el algoritmo *Naïve Bayes,* con una frecuencia baja igual a diez y veinte respectivamente, se requirió analizar los resultados del algoritmo con una frecuencia baja igual a treinta, como se muestra en la Figura.0.8.2.

```
> conf.mat30
Confusion Matrix and Statistics

          Reference
Prediction   Asalto Explotación Homicidio Secuestro Suicidio Violación
  Asalto        344           0        19         1        4         1
  Explotación     1           0         0         0        0         0
  Homicidio      23           5       388        12       14        25
  Secuestro       0           0         0         7        0         0
  Suicidio        1           0         7         1       12         3
  Violación       1           1         0         0        0        71

Overall Statistics

               Accuracy : 0.8735
                 95% CI : (0.8506, 0.8941)
    No Information Rate : 0.44
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7954
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Asalto Class: Explotación Class: Homicidio Class: Secuestro Class: Suicidio Class: Violación
Sensitivity                 0.9297           0.000000          0.9372         0.333333         0.40000          0.71000
Specificity                 0.9562           0.998930          0.8501         1.000000         0.98683          0.99762
Pos Pred Value              0.9322           0.000000          0.8308         1.000000         0.50000          0.97260
Neg Pred Value              0.9545           0.993617          0.9451         0.985011         0.98037          0.96659
Prevalence                  0.3932           0.006376          0.4400         0.022317         0.03188          0.10627
Detection Rate              0.3656           0.000000          0.4123         0.007439         0.01275          0.07545
Detection Prevalence        0.3921           0.001063          0.4963         0.007439         0.02550          0.07758
Balanced Accuracy           0.9430           0.499465          0.8936         0.666667         0.69341          0.85381
```

FIGURA.0.8.2. MATRIZ DE CONFUSIÓN CON UNA FRECUENCIA BAJA IGUAL A TREINTA.

Este algoritmo actúa de forma rápida y sencilla teniendo como mejor resultado tomar en cuenta una frecuencia baja igual a treinta, ya que se analizó una predicción de la clasificación de forma aceptable y correcta. El algoritmo *Naïve Bayes* resulta ser uno de los algoritmos más utilizados en el aprendizaje automático gracias a su rapidez y facilidad de implementación.

Discusión

Cada uno de los algoritmos analizados en este documento contiene factores a favor y factores en contra que influyen en el momento de emitir un dictamen positivo o negativo acerca de su exactitud, rapidez y facilidad de ejecución, dado que estos factores son los que concierne debatir.

El algoritmo *K-means* resulto ser un algoritmo, sencillo al implementar, con requerimientos mínimos de hardware, originando que el algoritmo fuese de los más veloces en tiempo de ejecución. El algoritmo *K-means* utiliza la formula siguiente:

$$min_s E(\mu_i) = min_s \sum_{i=1}^{k} \sum_{x_j \in s_i} \|x_j - \mu_i\|^2$$

Donde la variable *s* corresponde al conjunto de datos cuyos elementos son los objetos x_j, los cuales son representados, como se ha mencionado anteriormente, como vectores, se obtuvieron un número de *k* clústers relacionados con el centroide μ_i.

Se realizó la observación acerca del reajuste de cada centroide en función matemática con la siguiente formula:

$$\frac{\partial E}{\partial \mu_i} = 0 \Rightarrow \mu_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{x_i \in S_i^{(t)}} x_j$$

Dado esto se resume que el algoritmo es un método sencillo de implementar y sin duda es un método ágil, es decir, rápido en ejecución. No obstante, su desventaja radica en la exactitud que proporciona al clasificar los elementos del conjunto de datos ya que el constante ajuste que se realiza a los centroides proporciona márgenes de error considerables.

En contra parte el algoritmo *K-nearest Neighbors,* resulto ser el algoritmo con un tiempo de ejecución muy amplio, alrededor de quince minutos por cada ejecución del modelo propuesto.

En el caso del algoritmo *K-nn* se empleó la denominada distancia euclidiana, para calcular la distancia de vecindad entre un elemento y el conjunto de elementos ya agrupados o bien conjunto de datos de entrenamiento. La formula de la distancia euclidiana se muestra a continuación.

$$\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

En el lado intermedio de ello se encuentra el algoritmo *Naïve Bayes,* el cual no resulta ser el más rápido en tiempos de ejecución, sin embargo, tampoco resulta ser el algoritmo que más demora para obtener los resultados de la clasificación.

En términos de exactitud, el algoritmo que mayor exactitud (*Acuraccy*) presenta es el algoritmo de *Naïve Bayes*, con una exactitud de clasificación de los *Tweets* muy aproximada a los valores reales expresados, en comparación con los dos algoritmos presentados K-*nearest Neighbors* y *K-means*.

El algoritmo *Naïve Baye*s se consideran espacios probabilísticos ligados a dos eventos, en los cuales se calcula la probabilidad condicional de que ocurra el evento *A* dado el evento *B.* Esta expresión matemática esta dada por la siguiente formula:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

A base de los elementos y hechos mencionados se dictamina que el algoritmo que presenta mejores resultados es el algoritmo de *Naïve Bayes* esto porque se demostró que presenta una adaptabilidad al modelo de datos presentados, resultando que el aprendizaje del algoritmo fuese rápido y certero al clasificar los *Tweets.* Los dos algoritmos restantes se adaptaron de forma correcta al modelo de datos, no obstante, presentaron deficiencias en funciones esenciales como el tiempo de ejecución y de la clasificación correcta de los elementos.

Conclusiones

En este proyecto se presentaron los resultados obtenidos tras ejecutar tres algoritmos de clasificación*, K-means, Naïve Bayes, K-Nearest Neighbor*, para el agrupamiento de delitos sucedidos en la Zona Metropolitana de la Ciudad de México y reportados en *Twitter*. Los resultados evidencian que cuando se configuran, o encuentran, los parámetros adecuadamente de los algoritmos empleados, éstos nos permiten obtener meta información relevante de un conjunto de datos. Para nuestro caso, el conjunto de datos utilizado son los comentarios reportados en Twitter de delitos y de ellos se obtuvo, a manera de ejemplo, la semejanza existente entre las palabras utilizadas en uno u otro delito. Debido a la dispersión existente entre las palabras de los *Tweets* se observó más sencillez de implementación en los algoritmos *de K-Means y Naïve Bayes* con respecto a la *K-Nearest Neighbor*, esto debido al funcionamiento diferente de los algoritmos. El mejor resultado obtenido fue de 87.35% para el caso del algoritmo, de clasificación, *Naïve Bayes*; de lo cual podemos concluir que dicho algoritmo es el que mejor responde para la tarea que tenemos como objetivo en este proyecto terminal, la clasificación de delitos.

Es importante destacar la efectividad de cada algoritmo, al realizar el aprendizaje automático, debido a que cada uno de ellos cuenta con parámetros diferentes para tomar en cuenta al momento de realizar la clasificación. No obstante, poniendo en comparación los algoritmos *Naïve Bayes y K-Nearest Neighbor* se puede diferenciar y destacar la efectividad en torno a la clasificación de los *Tweets* del algoritmo *Naïve Bayes* frente a *K-Nearest Neighbor*. Debido a que ambos algoritmos se basan en el aprendizaje supervisado. Por otro lado, el algoritmo *k-Means* tiene un buen comportamiento frente a los dos algoritmos mencionados, sin embargo, el algoritmo que tiene una mejor presencia y exactitud al clasificar los textos es *Naïve Bayes*.

Como perspectivas, se tiene que es posible probar otros algoritmos de clasificación y con un conjunto de datos diferente pero que corresponda al mismo dominio, los delitos. Además, hoy en día nos encontramos en una turbulencia financiera, por ejemplo, esto debido a las situaciones políticas y económicas mundiales. Por lo cual se considera de suma importancia la capacidad de poder predecir acontecimientos financieros en torno a los mercados y bolsa de valores, los algoritmos de aprendizaje automático son una herramienta sustancial que nos permitirán poder realizar una predicción aceptable en este rubro y poder con ello contribuir de manera activa en el avance financiero del país. Entonces, los algoritmos empleados en este proyecto terminal podrán ser explotados en este campo dado que hemos observado que su método de predicción es fehaciente y aceptable.

Bibliografía

[1] "Clasificación Estadística de Delitos (CED)", www3.inegi.org.mx, 2017. [Online]. Disponible: http://www3.inegi.org.mx/sistemas/clasificaciónes/delitos.aspx. [Accedido: 01- Nov- 2017].

[2] G. Cherubini, J. Jelitto, and V. Venkatesan, "Cognitive Storage for Big Data", IEEE, vol. 49, no. 4, pp. 40–51, Abril 2016.

[3] J. Paniagua "Algoritmos de aprendizaje automático para el análisis de opiniones a partir de textos en español" proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2017.

[4] J. A Zinzun. "Comparativa de la clasificación de tumores obtenida por medio de los algoritmos k-Means, PAM, AGNES." proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2016.

[5] J. López. "Sistema para la clasificación de artículos científicos mediante el algoritmo K-means utilizando características semánticas" proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2015.

[6] P. Khanna, "Sentiment Analysis: An Approach to Opinion Mining from Twitter Data Using R," International Journal of Advanced Research in Computer Science, vol. 8, no. 8, pp. 1–5, Octubre 2017.

[7] J. Khairnar, M. Kinikar "Machine Learning Algorithms for Opinion Mining and Sentiment Classification". IJSRP, vol. 3, no. 6, pp. 1–6, Junio 2013.

[8] G. Sneka, "Algorithms for Opinion Mining and Sentiment Analysis: An Overview" International Journal of Advanced Research in Computer Science and Software Engineering, vol. 6, no. 2, pp.1-5, Febrero 2016.

[9] "R: Past and Future History". [Online]. Disponible: https://cran.r-project.org/doc/html/interface98-paper/paper.html. [Accedido: 15-Oct-2017].

[10] "RStudio IDE", RStudio Blog. [Online]. Disponible: https://blog.rstudio.com/categories/rstudio-ide/Pág.e/6/. [Accedido: 29-Oct-2017].

Apéndices

Apéndice A. Código para el procesamiento del Conjunto de datos

```
#*************************Packages*********************

library(twitteR)

library(NLP)

library(tm)

library(RColorBrewer)

library(ggplot2)

library(wordcloud)

library(twitteR)

library(lubridate)

library(RColorBrewer)

library(readr)

library(tm)

setwd("/Users/Stephany Anaya/Documents/Proyecto Terminal")


#**************set a seed*****************

set.seed(2018)


#**********************Read Tweets**********************

tweets = read.csv("tweets.csv", header = F, stringsAsFactors = FALSE,

                  fileEncoding = "latin1")

#To remove any missing value that might be present in the data, type this:

typeEmpty = which(tweets$V5 == "")

tweets = tweets[-c(typeEmpty), ]

id   = tweets$V1

date = tweets$V2
```

```
user = tweets$V3

text = tweets$V4

type = tweets$V5


#**************Create DataFrame*****************************

df = data.frame(id, date, user, text, type)


#**********************Transform Date**********************

#Extract current configuration

my_date = strptime(df$date, tz = 'CST', format = "%a %b %d %H:%M:%S CST %Y")

day     = day(my_date)

year    = year(mi_date)

mounth  = month(mi_date)

newDate = paste(mounth,year)

df$date = newDate

#newDate = as.Date(df$date,format = "%d %m %Y")

#df$date = newDate


#**************Plot DataFrame*****************************

#**************Define colors*****************************

colors = c("cadetblue1","lightpink",

           "aquamarine","indianred1",

           "gold","darkseagreen2")


plot(df$type,main = "ClasificaciÃ³n de Delitos", pch = 15, col = colors)
```

```
names = c("AristeguiOnline", "cibercrimen", "CNNEE", "El_Universal_Mx", "eleconomista",
        "elsolde_mexico",  "Excelsior", "GoogleNewsMX", "lajornadaonline", "MexDenuncia",
        "Milenio","MXQnoticias", "NoticiasMVS", "noticimexico", "Notimex",
        "NTelevisa_com", "OEMenlinea", "Pajaropolitico", "Reforma", "Siete24Noticias",
        "SS_Edomex","SSP_CD")

par(mai = c(1,2,1,1))
plot(df$user,main = "Cuentas de Twitter", horiz = TRUE,
     names.arg = names,col = colors,las = 1)
table(df$user)

tableUserType = table(df$user,df$type)
write.csv(tableUserType, file = "tableUserType.csv")

tableTypeUser = table(df$type,df$user)
plot(tableTypeUser,main = "ClasificaciÃ³n de Delitos por Cuenta de Twitter",
     names.arg = names,col = colors,las = 1)

tableDateType = table(df$date,df$type)
write.csv(tableDateType, file = "tableDateType.csv")

tableUserDate = table(df$user,df$date)
write.csv(tableUserDate, file = "tableUserDate.csv")

#**************Build Corpus**************************
myCorpus = Corpus(VectorSource(df$text))
```

```
myCorpus

#***********Convert to lowercase*******************
myCorpus = tm_map(myCorpus, tolower)

#***********Remove Puntuacions*********************
myCorpus = tm_map(myCorpus, removePunctuation)

#***********Remove numbers*************************
myCorpus = tm_map(myCorpus, removeNumbers)

#***********Remove URLs****************************
removeURL = function(x)
  gsub("http[^[:space:]]*", "", x)
myCorpus = tm_map(myCorpus, content_transformer(removeURL))

#***********Remove Stopwords***********************
myStopWords = c(stopwords('spanish'), "available", "via")
myCorpus    = tm_map(myCorpus, removeWords, myStopWords)
myCorpus
inspect(myCorpus)
summary(myCorpus)

#***************Steaming****************************
dictCorpus = myCorpus
myCorpus   = tm_map(myCorpus, stemDocument)
```

```
inspect(myCorpus[1:3])

#*********Build Matriz Document-Term**************

tdm = TermDocumentMatriz(myCorpus, control = list(minWordLength = 1))

inspect(tdm[266:270, 31:40])

#***************Frequent Terms and Associations*******************

frecuencyTerm = findFreqTerms(tdm, lowfreq = 10)

termFrequency = rowSums(as.matriz(tdm))

termFrequency = subset(termFrequency, termFrequency >= 50)

DFtermFreq    = data.frame(term = names(termFrequency), freq = termFrequency)

write.csv(DFtermFreq, file = "DFtermFreq.csv")

ggplot(DFtermFreq, aes(x = term, y = freq))+ geom_bar(stat = "identity") + xlab("TérminoTérminos
Frecuentes") + ylab("Frecuencia") + coord_flip()

#***************WordCloud***********************************************

m = as.matriz(tdm)

#***********Calculate Frequency of Words*********************************

v = sort(rowSums(m), decreasing = TRUE)

nombres = names(v)

d = data.frame(word = nombres, freq = v)

write.csv(d, file = "wordsInWordCloud.csv")

#***********Plot WordCloud*************************************************

pal1 = brewer.pal(8,"Spectral")

pal2 = brewer.pal(8,"BrBG")
```

```
pal3 = brewer.pal(8,"RdGy")

wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=50,max.words=Inf, rot.per=.15, colors=pal3)

wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=40,max.words=Inf, rot.per=.15, colors=pal1)

wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=30,max.words=Inf, rot.per=.15, colors=pal2)

wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=20,max.words=Inf, rot.per=.15, colors=pal3)

wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=10,max.words=Inf, rot.per=.15, colors=pal1)

wordcloud(d$word, d$freq, scale=c(8,.2),min.freq=6,max.words=Inf, rot.per=.15, colors=pal2)
```

Apéndice B. Código para el procesamiento con el Algoritmo *K-means*

```
#**************************Packages*******************************

library(gmodels)

library(tidytext)

library(syuzhet)

library(cluster)

setwd("/Users/Stephany Anaya/Documents/Proyecto Terminal/Kmeans")


#**************set a seed*****************

set.seed(2018)


#**************remove sparse terms*****************************

inspect(myCorpus[1:3])

removeSparseTDM = removeSparseTerms(tdm, sparse = 0.95)

inspect(removeSparseTDM)

matrizRemoveSparseTDM = as.matriz(removeSparseTDM)

write.csv(matrizRemoveSparseTDM, file = "matrizRemoveSparseTDM.csv")


#**************cluster terms***********************************

distMatriz = dist(scale(matrizRemoveSparseTDM))
```

```
DistMatriz = as.matriz(distMatriz)
write.csv(DistMatriz, file = "DistMatriz.csv")
fit = hclust(distMatriz, method = "ward.D")
plot(fit)

#**************Cut tree into 6 clusters************************
rect.hclust(fit, k = 6)
(groups = cutree(fit, k = 6))
write.csv(groups, file = "GroupsWords.csv")

#**************transpose the matriz to cluster documents (tweets)*****************
m3 = t(matrizRemoveSparseTDM)

#**************k-means clustering of tweets k = 3*****************
kmeansResult3 = kmeans(m3, 3)
kmeansResult3
tablekmeansResult3 = table(kmeansResult3$cluster)
write.csv(tablekmeansResult3, file = "tablekmeansResult3.csv")

#**************cluster centers**********************************
roundKmeans3 = data.frame(round(kmeansResult3$centers, digits = 3))
write.csv(roundKmeans3, file = "roundKmeans3.csv")

#**************cluster centers**********************************
for (i in 1:k){
  cat(paste("cluster ", i, ":  ", sep = ""))
```

```
  s = sort(kmeansResult3$centers[i, ], decreasing = T)
  cat(names(s)[1:5], "\n")
}

#**************Table cluster**********************************

tablekmeansResultClasific3 = table(kmeansResult3$cluster,df$type)
write.csv(tablekmeansResultClasific3, file = "tablekmeansResultClasific3.csv")
crossTableKmeans3 = CrossTable(kmeansResult3$cluster,df$type)
write.csv(crossTableKmeans3, file = "crossTableKmeans3.csv")

kmeansResult3$cluster = as.factor(kmeansResult3$cluster)
ggplot(df, aes(date,user, color = kmeansResult3$cluster)) + geom_point()
clusplot(m3, kmeansResult3$cluster, main = "K-means K = 3",
         color = T, shade = T, labels = 1, lines = 0)

#**************k-means clustering of tweets k = 6**************
kmeansResult6 = kmeans(m3, 6)
kmeansResult6
tablekmeansResult6 = table(kmeansResult6$cluster)
write.csv(tablekmeansResult6, file = "tablekmeansResult6.csv")

#**************cluster centers*********************************
roundKmeans6 = data.frame(round(kmeansResult6$centers, digits = 6))
write.csv(roundKmeans6, file = "roundKmeans6.csv")
```

```
#**************cluster centers*********************************
for (i in 1:k){
  cat(paste("cluster ", i, ":  ", sep = ""))
  s = sort(kmeansResult6$centers[i, ], decreasing = T)
  cat(names(s)[1:5], "\n")
}

#**************Table cluster***********************************
tablekmeansResultClasific6 = table(kmeansResult6$cluster,df$type)
write.csv(tablekmeansResultClasific6, file = "tablekmeansResultClasific6.csv")
crossTableKmeans6 = CrossTable(kmeansResult6$cluster,df$type)
write.csv(crossTableKmeans6, file = "crossTableKmeans6.csv")

kmeansResult6$cluster = as.factor(kmeansResult6$cluster)
ggplot(df, aes(date,user, color = kmeansResult6$cluster)) + geom_point()
clusplot(m3, kmeansResult6$cluster,main = "K-means K = 6",
         color = T, shade = T, labels = 1, lines = 0)

#**************k-means clustering of tweets k = 2***************
kmeansResult2 = kmeans(m3, 2)
kmeansResult2
tablekmeansResult2 = table(kmeansResult2$cluster)
write.csv(tablekmeansResult2, file = "tablekmeansResult2.csv")

#**************cluster centers*********************************
roundKmeans2 = data.frame(round(kmeansResult2$centers, digits = 2))
```

```
write.csv(roundKmeans10, file = "roundKmeans2.csv")


#**************cluster centers************************************
for (i in 1:k){
  cat(paste("cluster ", i, ":  ", sep = ""))
  s = sort(kmeansResult2$centers[i, ], decreasing = T)
  cat(names(s)[1:5], "\n")
}


#**************Table cluster************************************
tablekmeansResultClasific2 = table(kmeansResult2$cluster,df$type)
write.csv(tablekmeansResultClasific10, file = "tablekmeansResultClasific10.csv")
crossTableKmeans2 = CrossTable(kmeansResult2$cluster,df$type)
write.csv(crossTableKmeans2, file = "crossTableKmeans2.csv")


kmeansResult2$cluster = as.factor(kmeansResult2$cluster)
ggplot(df, aes(date,user, color = kmeansResult2$cluster)) + geom_point()
clusplot(m3, kmeansResult2$cluster,main="K-means K = 2",
         color = T, shade = T, labels = 1, lines = 0)


#**************get rid of the problem characters********************
df$text = sapply(df$text,function(row) iconv(row, "latin1", "UTF-8", sub=""))
Tweet_sentiment = get_nrc_sentiment(df$text)
write.csv(Tweet_sentiment, file = "Tweetsentiment.csv")
```

Apéndice C. Código para el procesamiento con el Algoritmo K-Nearest Neighbor

```
#**************************Packages*********************************
library(tm)        # Text mining: Corpus and Document Term Matriz
```

```
library(class)     # KNN model
library(SnowballC) # Stemming words
library(plyr)
library(ggplot2)
setwd("/Users/Stephany Anaya/Documents/Proyecto Terminal/Knn")

#**************set a seed*****************************************
set.seed(2018)

#************Read myCorpus, convert to DocumentTermMatriz*************
docs = myCorpus
dtm  = DocumentTermMatriz(docs)

#************Column bind category (known classification)*************
mat.df = as.data.frame(data.matriz(dtm), stringsAsfactors = FALSE)
mat.df = cbind(mat.df, df$type)

#**********Change name of new column to "category"*******************
colnames(mat.df)[ncol(mat.df)] = "category"

#*********Split data by rownumber into two equal portions**************
train = sample(nrow(mat.df), ceiling(nrow(mat.df) * .70))
test  = (1:nrow(mat.df))[- train]

#******************Isolate classifier**********************************
cl = mat.df[, "category"]
```

```
#*************Create model data and remove "category"*****************
modeldata = mat.df[,!colnames(mat.df) %in% "category"]

#****Create model: training set, test set, training set classifier k = 3*****
knn.pred3 = knn(modeldata[train, ], modeldata[test, ], cl[train], k = 3)
knn.pred3

#********************Confusion matriz**********************************
conf.mat3 = table("Predictions" = knn.pred3, Actual = cl[test])
conf.mat3

#************************Accuracy*************************************
(accuracy3 = sum(diag(conf.mat3))/length(test) * 100)

#****Create data frame with test data and predicted category***********
df.pred3 = cbind(knn.pred3, modeldata[test, ])

#****Create model: training set, test set, training set classifier k = 6*****
knn.pred6 = knn(modeldata[train, ], modeldata[test, ], cl[train], k = 6)

#********************Confusion matriz**********************************
conf.mat6 = table("Predictions" = knn.pred6, Actual = cl[test])
conf.mat6

#************************Accuracy*************************************
```

```
(accuracy6 = sum(diag(conf.mat6))/length(test) * 100)

#****Create data frame with test data and predicted category**********
df.pred6 = cbind(knn.pred6, modeldata[test, ])

#****Create model: training set, test set, training set classifier k = 2*****
knn.pred2 = knn(modeldata[train, ], modeldata[test, ], cl[train], k = 2)

#********************Confusion matriz**********************************
conf.mat2 = table("Predictions" = knn.pred2, Actual = cl[test])
conf.mat2

#************************Accuracy*************************************
(accuracy2 = sum(diag(conf.mat2))/length(test) * 100)
#****Create data frame with test data and predicted category**********
df.pred2 = cbind(knn.pred10, modeldata[test, ])
```

Apéndice D. Código para el procesamiento con el Algoritmo *Naïve Bayes*

```
#**************************Packages*******************************
library(caret)
library(e1071)
library(tm)
library(RTextTools)
library(gmodels)
library(dplyr)
library(vars)
library(forecast)
library(doMC)
```

```
setwd("/Users/Stephany Anaya/Documents/Proyecto Terminal/NaïveBayes")

#**************set a seed*************************************
set.seed(2018)

#********Split data by rownumber into two equal portions********
#train = sample(nrow(df), ceiling(nrow(df) * .70))
#test  = (1:nrow(df))[- train]

df.train = df[1:2195,]
write.csv(df.train , file = "dataFrameTrain.csv")
df.test  = df[2195:3135,]
write.csv(df.test , file  = "dataFrameTest.csv")

tdm.train = tdm[1:2195,]
dim(tdm.train)
tdm.test  = tdm[2195:3135,]
dim(tdm.test)

myCorpus.train = myCorpus[1:2195]
myCorpus.test  = myCorpus[2195:3135]

#**************FrecuencyTerms in train************************
terFrecuency10 = findFreqTerms(tdm.train, lowfreq = 10)
write.csv(terFrecuency10 , file = "terFrecuency10.csv")
```

```
terFrecuency20 = findFreqTerms(tdm.train, lowfreq = 20)
write.csv(terFrecuency20 , file = "terFrecuency20.csv")

terFrecuency30 = findFreqTerms(tdm.train, lowfreq = 30)
write.csv(terFrecuency30 , file = "terFrecuency30.csv")

length(terFrecuency10)
length(terFrecuency20)
length(terFrecuency30)

tdm.train.nb10 = DocumentTermMatriz(myCorpus.train,
                        control = list(dictionary = terFrecuency10))
dim(tdm.train.nb10)

tdm.train.nb20 = DocumentTermMatriz(myCorpus.train,
                         control = list(dictionary = terFrecuency20))
dim(tdm.train.nb20)

tdm.train.nb30 = DocumentTermMatriz(myCorpus.train,
                        control = list(dictionary = terFrecuency30))
dim(tdm.train.nb30)

#**************FrecuencyTerms in test************************
tdm.test.nb10 = DocumentTermMatriz(myCorpus.test, control=list(dictionary = terFrecuency10))
dim(tdm.test.nb10)
```

```
tdm.test.nb20 = DocumentTermMatriz(myCorpus.test, control=list(dictionary = terFrecuency20))
dim(tdm.test.nb20)

tdm.test.nb30 = DocumentTermMatriz(myCorpus.test,
                                   control=list(dictionary = terFrecuency30))
dim(tdm.test.nb30)

#**************Convert type to number*******************************
convert_count = function(x) {
  y = ifelse(x > 0, 1,0)
  y = factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

trainNB10 = apply(tdm.train.nb10, 2, convert_count)
write.csv(trainNB10 , file = "trainNB10.csv")
testNB10  = apply(tdm.test.nb10, 2, convert_count)
write.csv(testNB10 , file = "testNB10.csv")

trainNB20 = apply(tdm.train.nb20, 2, convert_count)
write.csv(trainNB20 , file = "trainNB20.csv")
testNB20  = apply(tdm.test.nb20, 2, convert_count)
write.csv(testNB20 , file = "testNB20.csv")

trainNB30 = apply(tdm.train.nb30, 2, convert_count)
write.csv(trainNB30 , file = "trainNB30.csv")
```

```
testNB30  = apply(tdm.test.nb30, 2, convert_count)
write.csv(testNB30 , file = "testNB30.csv")

#**************NaïveBayes10**************************************************
system.time( classifier <- NaïveBayes(trainNB10, df.train$type, laplace = 1) )
# Use the NB classifier we built to make predictions on the test set.
system.time( pred10 <- predict(classifier, newdata = testNB10) )
# Create a truth table by tabulating the predicted class labels with the actual class labels
tablePredAct10 = table("Predictions"= pred10,  "Actual" = df.test$type)
write.csv(tablePredAct10 , file = "tablePredAct10.csv")
plot(pred10,main="Predictions 10",col = colors,las=1)
crossTable10 = CrossTable(pred10,df.test$type)
crossTable10 = data.frame(crossTable10)
write.csv(crossTable10 , file = "crossTable10.csv")
conf.mat10 = confusionMatriz(pred10, df.test$type)

#**************NaïveBayes20**************************************************
system.time( classifier = NaïveBayes(trainNB20, df.train$type, laplace = 1) )
# Use the NB classifier we built to make predictions on the test set.
system.time( pred20 = predict(classifier, newdata = testNB20) )
# Create a truth table by tabulating the predicted class labels with the actual class labels
tablePredAct20 = table("Predictions" = pred20,  "Actual" = df.test$type)
write.csv(tablePredAct20 , file = "tablePredAct20.csv")
plot(pred20,main = "Predictions 20",col = colors,las=1)
crossTable20 = CrossTable(pred20,df.test$type)
crossTable20 = data.frame(crossTable20)
```

```
write.csv(crossTable20 , file = "crossTable20.csv")

conf.mat20 = confusionMatriz(pred20, df.test$type)


#**************NaïveBayes30*****************************************************

system.time( classifier30 = NaïveBayes(trainNB30, df.train$type, laplace = 1) )

# Use the NB classifier we built to make predictions on the test set.

system.time( pred30 = predict(classifier, newdata = testNB30) )

# Create a truth table by tabulating the predicted class labels with the actual class labels

tablePredAct30 = table("Predictions"= pred30,  "Actual" = df.test$type)

write.csv(tablePredAct30 , file = "tablePredAct30.csv")

plot(pred30,main = "Predictions 30",col = colors,las=1)

crossTable30 = CrossTable(pred30,df.test$type)

crossTable30 = data.frame(crossTable30)

write.csv(crossTable30 , file = "crossTable30.csv")

conf.mat30 = confusionMatriz(pred30, df.test$type)
```