Sentiment Analysis.

Franco Urbinati

***Resumen* — El presente informe describe brevemente en qué consiste el campo de NLP llamado Sentiment Analysis y como realizar un ejemplo práctico en Python utilizando la librería TextBlob.**

***Palabras claves* — NLP, Sentiment Analysis, Python, TextBlob, Naive Bayes classifier.**

I. INTRODUCCIÓN

El análisis del sentimiento, o la minería de opinión, es un área activa de estudio en el campo del procesamiento del lenguaje natural que analiza las opiniones, sentimientos, evaluaciones, actitudes de la gente, y emociones a través del tratamiento computacional de la subjetividad en el texto.

En un mundo en el que se generan 2,5 trillones de bytes de datos cada día, el análisis de los sentimientos se ha convertido en una herramienta clave para dar sentido a esos datos. Esto ha permitido a las empresas obtener información clave y automatizar todo tipo de procesos.

II. MOTIVACIÓN

Actualmente, el análisis de sentimiento es un tema de gran interés y desarrollo ya que tiene muchas aplicaciones prácticas. Dado que la información pública y privada disponible a través de Internet está en constante crecimiento, una gran cantidad de textos que expresan opiniones están disponibles en sitios de revisión, foros, blogs y redes sociales.

Con la ayuda de los sistemas de análisis de sentimientos, esta información no estructurada puede transformarse automáticamente en datos estructurados de opiniones públicas sobre productos, servicios, marcas, política o cualquier tema sobre el que las personas puedan expresarse. Estos datos pueden ser muy útiles para aplicaciones comerciales, como análisis de marketing, relaciones públicas, revisiones de productos, puntuación de promotor neto, comentarios de productos y servicio al cliente. Por ejemplo, el gobierno de Obama utilizó el análisis del sentimiento para medir la opinión pública de los anuncios de políticas y mensajes de campaña antes de las elecciones presidenciales de 2012. Ser capaz de ver rápidamente los sentimientos en las publicaciones en un foro, redes sociales y hasta los artículos de noticias, significa estar en mejores condiciones para planificar estrategias.

Como ejemplo sencillo, utilizando la librería TextBlob, se hará “Brand Monitoring” o monitoreo de marca sobre la empresa Tesla en la red social Twitter.

III. ¿QUÉ ES SENTIMENT ANALYSIS?

El análisis del sentimiento también conocido como Opinion Mining es un campo dentro del procesamiento del lenguaje natural (NLP) que construye sistemas que intentan identificar y extraer opiniones dentro del texto. Por lo general, además de identificar la opinión, estos sistemas extraen atributos de la expresión, por ejemplo:

- Polarity (polaridad): si el hablante expresa una opinión positiva o negativa.
- Subject (asunto): lo que se está hablando.
- Opinion holder: la persona o entidad que expresa la opinión.

El análisis de opinión, al igual que muchos otros problemas de NLP (natural language processing), se puede modelar como un problema de clasificación donde se deben resolver dos sub-problemas:

1. Clasificar una oración como subjetiva u objetiva, conocida como “subjectivity classification”.
2. Clasificar una oración expresando una opinión positiva, negativa o neutral, conocida como clasificación de polaridad o “polarity classification”.

Se estima que el 80% de los datos del mundo no están estructurados y no están organizados de una manera predefinida. La mayoría de esto proviene de datos de texto, como correos electrónicos, tickets de soporte, chats, redes sociales, encuestas, artículos y documentos. Estos textos suelen ser difíciles, lentos y caros de analizar, comprender y ordenar.

Los sistemas de análisis de sentimiento permiten a las empresas dar sentido a este mar de texto no estructurado mediante la automatización de procesos, obteniendo información procesable y ahorrando horas de procesamiento de datos manual, en otras palabras,

haciendo que los equipos sean más eficientes. Algunas de las ventajas del análisis de sentimiento incluyen lo siguiente:

- Escalabilidad: El análisis de sentimiento permite procesar datos a escala de una manera eficiente y rentable.
- Análisis en tiempo real: Se puede utilizar el análisis de sentimientos para identificar información crítica que permita el conocimiento situacional durante escenarios específicos en tiempo real.
- Criterio consistente: Los humanos no poseen criterios claros para evaluar el sentimiento de una pieza de texto. Se estima que diferentes personas solo acuerdan alrededor del 60-65% de las veces al juzgar el sentimiento de una pieza de texto en particular. Es una tarea subjetiva que está fuertemente influenciada por experiencias personales, pensamientos y creencias. Al utilizar un sistema de análisis de sentimiento centralizado, las compañías pueden aplicar los mismos criterios a todos sus datos. Esto ayuda a reducir los errores y mejorar la coherencia de los datos.

IV. ALGORITMOS DE SENTIMENT ANALYSIS

Los métodos y algoritmos para implementar sistemas de análisis de sentimiento se pueden clasificar como:

- Rule-based systems: Sistemas basados en reglas que realizan análisis de sentimientos basados en un conjunto de reglas elaboradas manualmente.
- Automatic systems: Sistemas automáticos que se basan en técnicas de aprendizaje automático o machine learning.
- Hybrid systems: Sistemas híbridos que combinan ambos enfoques previos.

A. Rule-based systems

Por lo general, este tipo de enfoques definen un conjunto de reglas en algún tipo de lenguaje de scripting que identifica la subjetividad, la polaridad o el tema de una opinión.

Las reglas pueden usar una variedad de inputs, como por ejemplo los "lexicons". Un léxico de sentimiento o "sentiment lexicon" es una lista de características léxicas (por ejemplo, palabras) que generalmente se etiquetan según su orientación semántica como positivas o negativas. Mientras que crear y validar tales listas se encuentran entre los métodos más sólidos y robustos, también es uno de los que consume más tiempo.Un ejemplo básico de una implementación basada en reglas sería:

1. Definir dos listas de palabras polarizadas (por ejemplo, palabras negativas como mala, peor, fea, etc. y palabras positivas como bueno, mejor, hermoso, etc.).
2. Dado un texto: Contar la cantidad de palabras positivas y negativas que aparecen en el texto.
3. Si el número de apariciones positivas de palabras es mayor que el número de apariciones negativas de palabras, devuelve un sentimiento positivo. De manera inversa, devuelve un sentimiento negativo. De no cumplirse ninguna de ambas condiciones, devuelve neutral.

B. Automatic Systems

Los métodos automáticos no se basan en reglas elaboradas manualmente, sino en técnicas de aprendizaje automático. La tarea de análisis de sentimiento generalmente se modela como un problema de clasificación donde un clasificador se alimenta con un texto y devuelve la categoría correspondiente. Dicho clasificador de aprendizaje automático generalmente se puede implementar con los siguientes pasos y componentes:

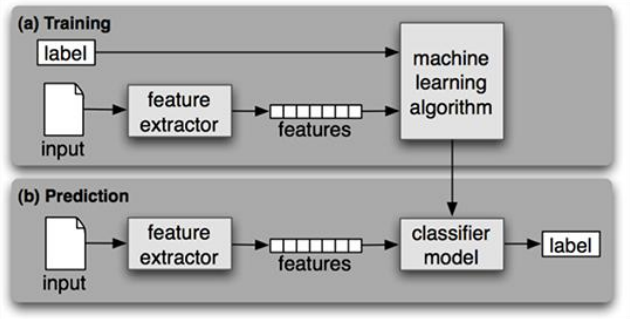

Figura 1: Training and Prediction

En el proceso de entrenamiento(a), el modelo aprende a asociar una entrada particular (es decir, un texto) a la salida correspondiente (etiqueta) en función de las muestras de prueba utilizadas para el entrenamiento. El extractor de funciones (feature extractor) transfiere la entrada de texto a un vector, es decir, transforma el texto en una representación numérica. Los pares de vectores de características y etiquetas (por ejemplo, positivo, negativo o neutral) se introducen en el algoritmo de aprendizaje automático para generar un modelo.

En el proceso de predicción(b), el feature extractor se usa para transformar entradas de texto en vectores de características. Estos vectores de características se

introducen en el modelo, que genera etiquetas (positivas, negativas o neutrales) como resultado.

El paso de clasificación generalmente involucra un modelo estadístico como Naïve Bayes, Regresión logística, Máquinas de vectores de soporte o Redes neuronales.

Métricas y evaluación del análisis de sentimiento

Hay muchas maneras en que se pueden obtener métricas de rendimiento para evaluar un clasificador y para entender qué tan preciso es un modelo de análisis de sentimiento. Uno de los más utilizados se conoce como validación cruzada o cross-validation.

El procedimiento tiene un único parámetro llamado k que se refiere a la cantidad de grupos en los que se divide una muestra de datos determinada. Como tal, el procedimiento a menudo se llama k-fold cross-validation. Es un método popular porque es simple de entender y porque generalmente da como resultado una estimación menos sesgada o menos optimista de la habilidad del modelo que otros métodos, como una simple división de entrenamiento / prueba. El procedimiento general es el siguiente:

1. Mezclar aleatoriamente el conjunto de datos.
2. Dividir el conjunto de datos en k grupos.
3. Para cada grupo único:
    - Considerar al grupo como un conjunto de datos de prueba.
    - Tomar los grupos restantes como un conjunto de datos de entrenamiento.
    - Ajustar un modelo con el conjunto de entrenamiento y evaluarlo con el conjunto de prueba.
    - Conservar el puntaje de evaluación y descartar el modelo.
4. Calcular un promedio de las métricas obtenidas.

Es importante destacar que cada observación en la muestra de datos se asigna a un grupo individual y permanece en ese grupo durante la duración del procedimiento. Esto significa que cada muestra tiene la oportunidad de usarse en el conjunto de prueba una vez y se usa para entrenar el modelo k-1 veces.

C. Hybrid Systems

El concepto de métodos híbridos es muy intuitivo: simplemente se combina lo mejor de ambos mundos (el basado en reglas y el automático). Por lo general, al combinar ambos enfoques, los métodos pueden mejorar tanto en precisión como en exactitud.

V. EJEMPLO PRÁCTICO SENCILLO

A continuación se describen los pasos para realizar un ejemplo práctico sencillo en Python 3.6, donde se obtienen una determinada cantidad Tweets sobre un tema en particular y se utiliza la librería TextBlob para aplicar análisis de sentimientos al texto de cada uno, obteniendo un índice de polaridad para determinar si la naturaleza de los mismos es positiva, neutral o negativa. En este ejemplo, se calcula el porcentaje de tweets con naturaleza positiva, neutral y negativa para la empresa Tesla. Este es un caso de uso conocido como "Brand Monitoring" donde la empresa puede determinar a grandes rasgos qué concepto tiene la gente de la misma.

En la figura 2 se muestra el código de la función principal del ejemplo.

```
125  def main():
126      # creating object of TwitterClient Class
127      api = TwitterClient()
128      # connect to twitter and get tweets
129      analyzed_tweets = api.get_tweets('#Tesla',1000)
130      if(len(analyzed_tweets)!= 0):
131          # generate csv with tweets (just for reading reference)
132          api.generate_csv(analyzed_tweets)
133
134          #print tweets in console
135          for analyzed_tweet in analyzed_tweets:
136              print("Tweet text: {0} - Polarity: {1} - {2}".format(analyze
137
138          #print percentege
139          api.print_percentege(analyzed_tweets)
140
141          #genete naives bayes model
142          api.naives_bayes(analyzed_tweets)
143
144
145  if __name__ == "__main__":
146      # calling main function
147      main()
```

Figura 2: Función main

El primer paso es importar todas las librerías necesarias. Se crea una instancia de la clase TwitterClient, cuyo constructor intenta realizar la conexión con twitter utilizando la librería Tweepy (figura 3). Para obtener tweets a través de la API de Twitter es necesario registrar una aplicación a través de una cuenta de twitter. Luego se deben obtener todas las claves necesarias en la pestaña "Keys and Access Tokens" del sitio.

```python
from textblob import TextBlob
from textblob import classifiers
import tweepy
import re
import csv
class TwitterClient(object):

    def __init__(self):
        # keys and tokens from the Twitter Dev Console
        consumer_key = '***'
        consumer_secret = '***'
        access_token = '***'
        access_token_secret = '***'

        # attempt authentication
        try:
            auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
            auth.set_access_token(access_token, access_token_secret)
            self.api = tweepy.API(auth)
        except:
            print("Error: Authentication Failed")

```

Figura 3: Constructor. Conexión con twitter

El siguiente paso es llamar a la función get_tweets. Esta función recibe el tema o hashtag de los cuales se buscarán tweets y la cantidad que se obtendrán (figura 4)

```python
# Return tweets dictionary with tweet text,sentiment,polarity
def get_tweets(self,keyword,tweetCount):

    #Stores tweet text with Polarity (positive/negative)
    tweets = []
    try:
        print('Getting tweets')
        for tweet in tweepy.Cursor(self.api.search, keyword, lang="en").items(tweetCount):

            # empty dictionary to store required params of a tweet
            parsed_tweet = {}

            # saving text of tweet
            parsed_tweet['text'] = tweet.text
            parsed_tweet['clean_text'] = self.clean_tweet(tweet.text)
```

Figura 4: Función get_tweets

El texto del tweet es enviado a la función clean_tweet, la cual se encarga de eliminar caracteres especiales y enlaces. En otras palabras, hace un preprocesamiento del texto.

Luego se utiliza la librería TextBlob para obtener el análisis deseado. Se debe pasar el texto del tweet como parámetro para crear un objeto TextBlob (figura 5). Este objeto posee varias propiedades, entre ellas “sentiment”. Esta propiedad posee dos índices, uno de polaridad y otro de subjetividad. En función del índice de polaridad, se clasifica al tweet como positivo, negativo o neutral. En la siguiente sección del informe se explica con más detalle el funcionamiento de esta librería.

```python
            # clean tweet for better analysis
            analysis = TextBlob(self.clean_tweet(tweet.text))

            # saving tweet sentiment
            parsed_tweet['sentiment'] = analysis.sentiment

            #saving tweet polarity
            if analysis.sentiment.polarity > 0:
                parsed_tweet['polarity'] = 'positive'
            elif analysis.sentiment.polarity < 0:
                parsed_tweet['polarity'] = 'negative'
            else:
                parsed_tweet['polarity'] = 'neutral'
```

Figura 5: Función get_tweets

Finalmente se llama a la función print_percenteges, la cual se encarga de imprimir los resultados obtenidos en consola. De los mil tweets recuperados sobre la empresa Tesla, los resultados fueron los siguientes:

```
Total: Positive 44.02810304449649% - Negative:20.843091334894613% - Neutral:35.1280562060089%
Naives Bayes Classifier - accuracy:0.8
Most Informative Features
          contains(car) = True           positi : negati =     10.5 : 1.0
          contains(buy) = True           positi : negati =      8.4 : 1.0
        contains(tweet) = True           negati : positi =      8.1 : 1.0
       contains(Model3) = True           positi : negati =      7.7 : 1.0
      contains(Twitter) = True           negati : positi =      6.9 : 1.0
         contains(news) = True           negati : positi =      6.9 : 1.0
          contains(SEC) = True           negati : positi =      6.5 : 1.0
          contains(CEO) = True           negati : positi =      5.6 : 1.0
         contains(back) = True           negati : positi =      5.6 : 1.0
           contains(m) = True            negati : positi =      5.6 : 1.0
```

Figura 6: Resultados

Como se puede observar en la salida por consola, el porcentaje de tweets positivos es ampliamente superior a los negativos.

El texto de los tweets junto con los índices de polaridad y subjetivdad son impresos por consola. Además para visualizar los tweets con mayor claridad, se invoca a la función generate_csv , la cual genera un archivo csv con la información mencionada anteriormente.

Con el objetivo de analizar la librería TextBlob con mayor profundidad, se utilizarán los tweets recolectados junto con la polaridad obtenida para armar un conjunto de entrenamiento, y luego utilizar un clasificador Naive Bayes para predecir la polaridad de los tweets restantes. Para ello el algoritmo invoca a la función naive_bayes, la cual recibe como parámetro los tweets analizados. La primer tarea que realiza es crear un set o conjunto de entrenamiento con el 80% de los tweets y un conjunto de testing con el 20%, descartando aquellos con sentimiento neutro. Textblob proporciona un módulo de clasificadores integrados para crear un clasificador personalizado. Para ello invocamos al clasificador de Textblob con el conjunto de entrenamiento obtenido en el paso anterior (línea 114 de la figura 7). Por último se

imprime la exactitud del clasificador al utilizar el conjunto de testing. En la figura 6 podemos observar que esta es 0.8, es decir, 80% de las muestras fueron clasificadas correctamente al utilizar el modelo entrenado. Finalmente, se invoca al método show_informative_features, el cual muestra por consola la relación positivo/negativo de las palabras más influyentes (figura 6). Por ejemplo, si el texto posee la palabra "Car" hay una muy alta probabilidad de que el tweet sea de naturaleza positiva.

```
    # Function to create a classifier
    def naives_bayes(self, analyzed_tweets):
        training = []
        testing = []

        #Get the number of tweets that are either positive or negative
        length = 0
        for tweet in analyzed_tweets:
            if (tweet['polarity'] != 'neutral'):
                length = length + 1

        #Build training set (80%) and testing set (20%)
        training_limit = round(length * 0.8)
        index = 0
        for tweet in analyzed_tweets:
            if (tweet['polarity'] != 'neutral'):
                row = []
                row.append(tweet['clean_text'])
                row.append(tweet['polarity'])

                if(index < training_limit):
                    training.append(row)
                else:
                    testing.append(row)
                index = index + 1


        #Create classifier
        classifier = classifiers.NaiveBayesClassifier(training)
        print("Naives Bayes Classifier - accuracy:{0}".format(classifier.accuracy(testin
        classifier.show_informative_features(10)
```

Figura 7: Función Naive Bayes

El código fuente junto el archivo csv se puede encontrar en el siguiente repositorio de github: https://github.com/francourbi/SentimentAnalysis/tree/master/venv

VI. TEXTBLOB

Como se mencionó anteriormente TextBlob es una biblioteca de Python (2 y 3) para procesar texto. Proporciona una API simple para realizar tareas comunes de procesamiento de lenguaje natural (NLP), como análisis de sentimiento, clasificación, traducción y más.

La función de sentimiento de TextBlob (sentiment) devuelve dos propiedades, polaridad (polarity) y subjetividad (subjectivity). La polaridad es un número flotante que se encuentra en el rango de [-1,1] donde 1 es lo más positivo y -1 lo más negativo posible. La subjetividad también es un número flotante que se encuentra en el rango de [0,1], siendo 0 lo más objetivo posible y 1 lo más subjetivo.

Se revisó el código en GitHub, donde se puede observar que TextBlob usa un conjunto de datos de críticas de películas en el que las reseñas ya se han etiquetado como positivas o negativas (imagen 8). Las características positivas y negativas se extraen de cada crítica positiva y negativa, respectivamente. Luego se utiliza un clasificador Naive Bayes con los datos previamente mencionados como conjunto de entrenamiento para obtener un modelo. Finalmente el texto del que se quiere obtener el índice de polaridad y subjetividad es utilizado como input en el modelo.

```
@requires_nltk_corpus
def train(self):
    """Train the Naive Bayes classifier on the movie review corpus."""
    super(NaiveBayesAnalyzer, self).train()
    neg_ids = nltk.corpus.movie_reviews.fileids('neg')
    pos_ids = nltk.corpus.movie_reviews.fileids('pos')
    neg_feats = [(self.feature_extractor(
        nltk.corpus.movie_reviews.words(fileids=[f])), 'neg') for f in neg_ids]
    pos_feats = [(self.feature_extractor(
        nltk.corpus.movie_reviews.words(fileids=[f])), 'pos') for f in pos_ids]
    train_data = neg_feats + pos_feats
    self._classifier = nltk.classify.NaiveBayesClassifier.train(train_data)
```

Figura 8: Función train (TextBlob en GitHub, sentiments.py)

VII. NAIVE BAYES CLASSIFIER

Es una técnica de clasificación basada en el teorema de Bayes, suponiendo independencia entre cada par de características dado el valor de la variable de clase. En términos simples, un clasificador de Naive Bayes supone que la presencia de una característica particular en una clase no está relacionada con la presencia de ninguna otra característica. Por ejemplo, una fruta puede considerarse una manzana si es roja, redonda y de aproximadamente 3 pulgadas de diámetro. Incluso si estas características dependen unas de otras o de la existencia de otras características, todas estas propiedades contribuyen de forma independiente a la probabilidad de que esta fruta sea una manzana y es por eso que se la conoce como "ingenua" o "naive". Para comprender mejor su funcionamiento se describe el siguiente ejemplo sencillo:

Se quiere determinar las probabilidades de jugar un partido en función del clima. Los pasos a seguir son:

1. Convertir el conjunto de datos en una tabla de frecuencias y crear la tabla de probabilidad (figura 9).

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Figura 9: Dataset

2. Utilizar la ecuación naive Bayesiana para calcular la probabilidad posterior para cada clase. La clase con la probabilidad posterior más alta es el resultado de la predicción:

*P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)*
P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64
*P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60*

A pesar de su simplicidad, Naive Bayes a menudo puede superar los métodos de clasificación más sofisticados. Los algoritmos de Naive Bayes se utilizan principalmente en análisis de sentimiento, filtrado de spam, sistemas de recomendación, etc. Son rápidos y fáciles de implementar, pero su mayor desventaja es que los predictores deben ser independientes. En la mayoría de los casos de la vida real, los predictores son dependientes, lo cual disminuye u obstaculiza el rendimiento del clasificador.

VIII. CONCLUSIONES

Trabajar con la librería Tweepy es sencillo, y con una pocas líneas de código se pueden realizar las conexiones necesarias para obtener un número limitado de tweets. Sin embargo, estos son suficientes para realizar análisis de prueba sencillos. Para ello se importó la librería TextBlob, que al igual que Tweepy, es simple de utilizar.

Se recuperaron 1000 tweets que contengan la palabra #Tesla. Para cada uno de ellos se hizo una limpieza de texto y se obtuvo un índice de polaridad y subjetividad. Utilizando el primero, se clasificaron en positivos (“positive”), neutros (“neutral”) o negativos (“negative”) y se calculó el porcentaje de cada clase. Los resultados indican que la empresa mayoría de los tweets son de naturaleza positiva (44%), seguidos de los neutrales (25%) y por último los negativos.

Una vez etiquetados o clasificados los tweets se utilizó un clasificador Naive Bayes con un 80% de los datos como entrenamiento, con el objetivo de poder predecir la naturaleza de los restantes. Para crear el modelo también se utilizó la librería TextBlob. Finalmente esta nos muestra la exactitud del mismo, junto con las palabras más influyentes a la hora de clasificar un tweet.

De estos resultados se pueden realizar algunas observaciones sencillas. Por ejemplo, las palabras “Car” y “Buy” tienen alto de índice de relación positivo/negativo. Es decir, si alguna de esas palabras aparecen en el tweet, hay una gran probabilidad de que estos tengan naturaleza positiva. Esto podría traducirse a que los clientes de Tesla están satisfechos con los automóviles comprados.

REFERENCIAS

[1] Conceptos de Sentiment Analysis: https://monkeylearn.com/sentiment-analysis/
[2] Sentiment Analysis con python: https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/
[3] Procesamiento de lenguaje natural con TextBlob: https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/
[4] Documentación TextBlob: https://textblob.readthedocs.io/en/dev/quickstart.html#sentiment-analysis
[5] Documentación Tweepy: https://tweepy.readthedocs.io/en/v3.5.0/
[6] TextBlob GitHub: http://github.com/sloria/TextBlob/blob/90cc87ab0f9e25f37379079840ec43aba59af440/textblob/en/sentiments.py
[7] Conceptos de Cross-Validation: https://machinelearningmastery.com/k-fold-cross-validation/
[8] Conceptos de clasificador Naive Bayes: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

Franco Urbinati

Ingeniero en Computación de la UNLP (2016). Actualmente cursando el postgrado de Especialización en Inteligencia de Datos orientada a Big Data de la UNLP