1 AVANCES TRABAJO FINAL

Para comenzar con los avances relacionados al trabajo, decidí indagar en cómo era la utilización de la API que brinda Mercado Libre para obtener los comentarios, y realizar con los comentarios obtenidos un **Análisis Exploratorio** de dichos comentarios para tener un mejor entendimiento de los datos.

Para llevar a cabo este procedimiento se requirió de la utilización de una serie de herramientas que serán detalladas a continuación:

```
jupyter==1.0.0
pandas==1.3.4
plotly==5.4.0
requests==2.26.0
nltk==3.6.5
```

```
NORMAL  +43 ~0 -0  develop  requeriments.txt[+]
```

- Jupyter: Jupyter Notebook es una aplicación web de código abierto que permite crear y compartir documentos que contienen código en vivo, ecuaciones, visualizaciones y texto narrativo. Se utiliza para limpieza y transformación de datos, simulación numérica, modelado estadístico, visualización de datos, aprendizaje automático, entre otras cosas.
- Pandas: herramienta de manipulación y análisis de datos de código abierto construida sobre el lenguaje de programación Python.
- Plotly: librería de código abierto basada en Python que crea gráficos interactivos con calidad de publicación.
- Requests: librería basada en Python que facilita realizar peticiones HTTP
- NLTK: es un conjunto de bibliotecas y herramientas para el procesamiento del lenguaje natural para el lenguaje de programación Python. Cabe destacar que esta herramienta es la única que provee un listado de “stopwords” en el idioma español.

Una vez que se han detallado todas las herramientas utilizadas, es posible proseguir con el análisis de los datos en sí. Para ello se creo un repositorio en GitHub, el cual contiene todos los avances logrados hasta la fecha.

Para realizar el análisis se creó un nuevo documento en Jupyter Notebook denominado ‘Analisis_Comentarios_ML.ipynb’ en el cual se realizó el siguiente análisis:

Se comenzó realizando las importaciones de todas las herramientas necesarias y descargando el listado de “stopwords” que la herramienta NLTK provee. También se añadió el path de dicho archivo porque sin el mismo, no puede ser encontrado.

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import requests
import json
import plotly.graph_objects as go
import plotly.express as px
import nltk
import re
from unicodedata import normalize
from nltk.corpus import stopwords
pd.set_option('display.max_colwidth', None)
```

In [2]:
```
# Esta linea es importante para descargar el archivo con las stopwords en el path indicado
# Una vez descargado ya se puede comentar o eliminar la linea
# La dejo a modo de documentación

nltk.download('stopwords', download_dir='/home/agu/Desktop/tesis/stopwords')
```

In [3]:
```
# Con esta linea se agrega el path en el que se descargaron las stopwords
# Es importante ya que sino no va a encontrar el archivo

nltk.data.path.append("/home/agu/Desktop/tesis/utils/stopwords") # Ubuntu Virtualbox
#nltk.data.path.append("/home/agu/repos/tesis/utils/stopwords") # Notebook
```

Para continuar se definieron una serie de funciones que serán útiles para el preprocesado del texto de los comentarios y para realizar las gráficas necesarias.

```python
def convert_response_to_list(reviews):
    list_reviews = []
    for review in reviews:
        new_json = {
            "titulo": review['title'].lower(),
            "comentario": review['content'].lower(),
            "valoracion": review['rate']
        }

        list_reviews.append(new_json)
    return list_reviews

def obtener_df_reviews(idProd):
    url_api = "https://api.mercadolibre.com/reviews/item/"
    url = url_api + idProd
    args = {'limit': 200}

    response = requests.get(url, params=args)
    if response.status_code != 200:
        return ""
    response_json = json.loads(response.text)
    reviews = convert_response_to_list(response_json['reviews'])
    df = pd.DataFrame.from_records(reviews)
    return df
```

```python
def preprocesar_comentarios_df(df):
    # Elimino signos como guines, dos puntos, etc.
    df["comentario"] = df['comentario'].str.replace('[^\w\s]','')

    # Dejo solo las palabras que no se encuentren dentro de las stopwords y que no son numericas
    df["comentario"] = df["comentario"].apply(
        lambda x: ' '.join(
            [word for word in x.split() if (word.lower() not in (stop)) and (word.isnumeric() == False)]
        )
    )

    # Con estas sentencias elimino todo tilpo de tildes, dieresis, etc (a excepcion de las ñ)
    df["comentario"] = df["comentario"].apply(
        lambda x: normalize(
            'NFC',
             re.sub(
                r"([^n\u0300-\u036f]|n(?!\u0303(?![\u0300-\u036f])))[\u0300-\u036f]+", r"\1",
                normalize("NFD", x), 0, re.I
            )
       )
    )

    return df[["comentario"]]

def contar_frecuencia_palabras_df(df):
    # Cuento la frecuencia con la que aparece cada palabra
    df_frecuencia = df.comentario.str.split(expand=True).stack().value_counts().reset_index()
    df_frecuencia.columns = ['palabra', 'frecuencia']
    return df_frecuencia

def plotear_frecuencia_palabras(df, tope_palabras, producto):
    fig = px.bar(
        df[0:tope_palabras],
        x='palabra',
        y='frecuencia',
        title = f'Top 50 palabras con mayor frecuencia del producto "{producto}"',
        labels = {
            "frecuencia": "Cantidad de veces que aparece la palabra",
            "palabra": "Palabra"
        }
    )
    fig.show()
```

Comenzando por la función **“convert_response_to_list”**, básicamente lo que hace la misma es recibir como parámetro una lista de reviews, y dejar solo los aspectos que son de utilidad para

la iniciativa de cada una, como lo son el título, la valoración y el comentario en sí. Para luego retornar nuevamente una lista de reviews en formato JSON solo con esos datos.

La función **"obtener_df_reviews"** recibe como parámetro un ID de un producto y llama a la API de Mercado Libre para obtener las reviews de dicho producto (nótese que aquí es donde se llama a la función anteriormente mencionada). Esta función devuelve un objeto DataFrame, que se trata de una estructura de la herramienta pandas, que facilita ampliamente realizar análisis sobre dichos comentarios. Una aclaración importante en esta función es que se le otorgó un limite de 200 comentarios para que la API no devuelva todos los que el producto tenía.

Continuando con la función **"preprocesar_comentarios_df"**, recibe como parámetro un DataFrame de comentarios, y le realiza un preprocesado a los mismos, eliminándole las stopwords a los mismos (la lista de stopwords fue provista por la herramienta NLTK), y además elimina guiones, tildes, diéresis, etc. Y devuelve el mismo dataframe de datos, solo que con los comentarios con ese procesamiento.

La función **"contar_frecuencia_palabras_df"** recibe al igual que la función anterior un dataframe de comentarios (el mismo ya se encuentra preprocesado gracias a la función anterior) y cuenta la frecuencia de aparición que tiene cada palabra que se encuentra en el mismo. Esta función devuelve un dataframe que contiene dos columnas. Una columna para las palabras que contiene el Dataframe, y la otra que indica la cantidad de veces que aparece.

Para finalizar, la última función denominada **"plotear_frecuencia_palabras"** lo que hace es recibir el dataframe anterior y realiza un grafico mostrando la frecuencia de las palabras (aclaración: el grafico muestra un límite de palabras indicado por el usuario.)

Posteriormente, llamé a la función de **"obtener_df_reviews"** tres veces, con tres productos distintos. Los productos utilizados en este análisis son el Samsung A02, Samsung A12 y Samsung A32

```
df_prod_1 = obtener_df_reviews("MLA17464694") # Samsung A02
df_prod_2 = obtener_df_reviews("MLA17415925") # Samsung A12
df_prod_3 = obtener_df_reviews("MLA17706115") # Samsung A32
```

Para continuar con el análisis me formulé la siguiente pregunta: **¿Hay alguna forma de obtener conocimiento fácilmente si un comentario es positivo o negativo?**

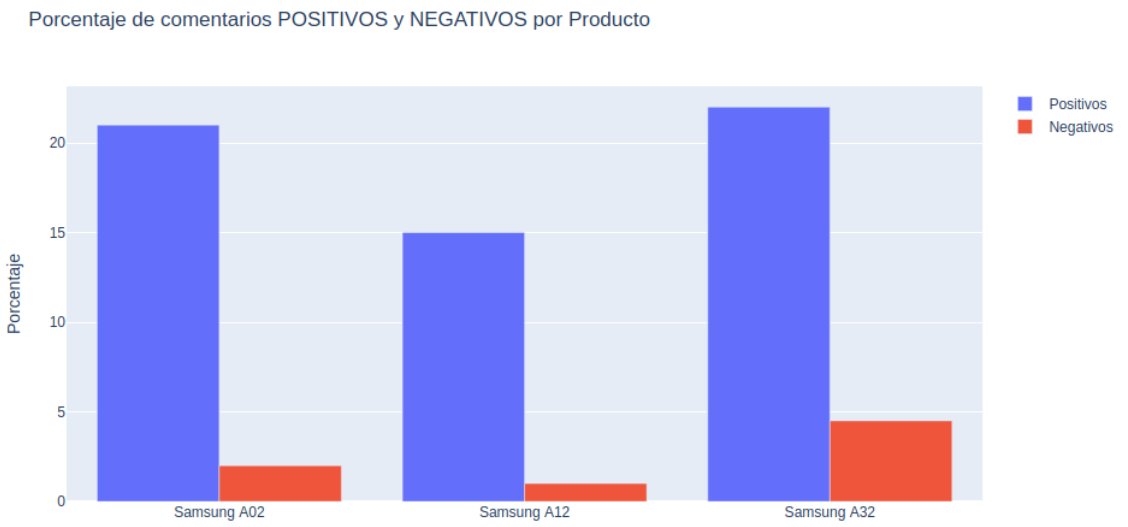
Para ello, y a fin de realizar un análisis preliminar, decidí filtrar los tres dataframes por los comentarios que contengan las palabras "bueno" y "malo", como se ve en la siguiente imagen.

```
df_prod_1_positivos = df_prod_1[df_prod_1["comentario"].str.contains("bueno")]
df_prod_1_negativos = df_prod_1[df_prod_1["comentario"].str.contains("malo")]
porc_comentarios_positivos_prod_1 = (len(df_prod_1_positivos) / len(df_prod_1)) * 100
porc_comentarios_negativos_prod_1 = (len(df_prod_1_negativos) / len(df_prod_1)) * 100


df_prod_2_positivos = df_prod_2[df_prod_2["comentario"].str.contains("bueno")]
df_prod_2_negativos = df_prod_2[df_prod_2["comentario"].str.contains("malo")]
porc_comentarios_positivos_prod_2 = (len(df_prod_2_positivos) / len(df_prod_2)) * 100
porc_comentarios_negativos_prod_2 = (len(df_prod_2_negativos) / len(df_prod_2)) * 100


df_prod_3_positivos = df_prod_3[df_prod_3["comentario"].str.contains("bueno")]
df_prod_3_negativos = df_prod_3[df_prod_3["comentario"].str.contains("malo")]
porc_comentarios_positivos_prod_3 = (len(df_prod_3_positivos) / len(df_prod_3)) * 100
porc_comentarios_negativos_prod_3 = (len(df_prod_3_negativos) / len(df_prod_3)) * 100
```

De esta manera se obtienen los porcentajes de los productos que a priori serían “positivos” y “negativos”. Obteniendo el siguiente gráfico:



Sin embargo, esto es una medida muy pobre para determinar tal valoración, ya que, como se demuestra en la siguiente imagen, no todos los comentarios catalogados como positivos los son, y viceversa.

Samsung A02

```
df_prod_1_positivos.head(5)
```

	titulo	comentario	valoracion
1	muy bueno	está bueno, ya el j7 me quedaba sin espacio. ahora bien. tenia 2 de memoria y este que tiene 3 anda mas lento. todo lo mejor lo están empeorando. hablo de todo, no de samsumg ni de los celulares exclusivamente. el jamón es peor. la ropa cara es peor. la barata es para usarla una vez. la gente que mueve los hilos del mundo es peor. es lo que hay y será. peor. el único lider que nos canta la posta es jose mujica. otra cosa del celu. lo pongo a cargar la batería en el piso, porque calienta demasiado. cosa que no pasaba con el j7. resumiendo. lo único bueno es que tenía 16 de espacio y ahora tengo 64. maravilloso. la casa a la que me mudé se llueve. ah pero es mas grande.	1
5	excelente celu. buena relación costo-prestaciones	el a02 es un excelente producto, con muy buena relación costo-prestaciones. lo estoy conociendo de a poco, de modo que aún no tengo opinión definitiva. tal vez hubiera sido interesante el reconocimiento de huella dactilar, pero de todos modos es muy usable con el pin de desbloqueo. la cámara es razonable para la categoría y costo del celular, destacable el lente macro que es muy bueno.	5
6	bueno	con respecto al samsung al a07 el a10 deja mucho que desear a mi gusto tiene muchos errores de diseño , es totalmente color negro del frente no se sabe cual es la parte superior e inferior , los botones del costado no son muy cómodos creo que uno de los botones al frente para activar o desactivar estarían más cómodos , pero bueno es así. saludos.	3
13	excelente	excelente vista agradable, es funcional para un publico no tan exigente, el tamaño de la pantalla es muy bueno, las fotos realmente me sorprendieron y eso que no tiene tantos megapixels. es ideal para hacer un regalo no vas a quedar mal, o para comprarselo para un niño o una persona mayor, su precio es justo.	5
15	espectacular	hermoso, lo compré para regalar, espero no lo pierda el gil. es liviano y se ve que le gustó, la otra vez se lo pedí para ver que onda, la pantalla táctil bien, la respuesta rápida y bueno, hace un par de días lo está usando así que por ahí agrego mas de la opinión.	5

```
df_prod_1_negativos
```

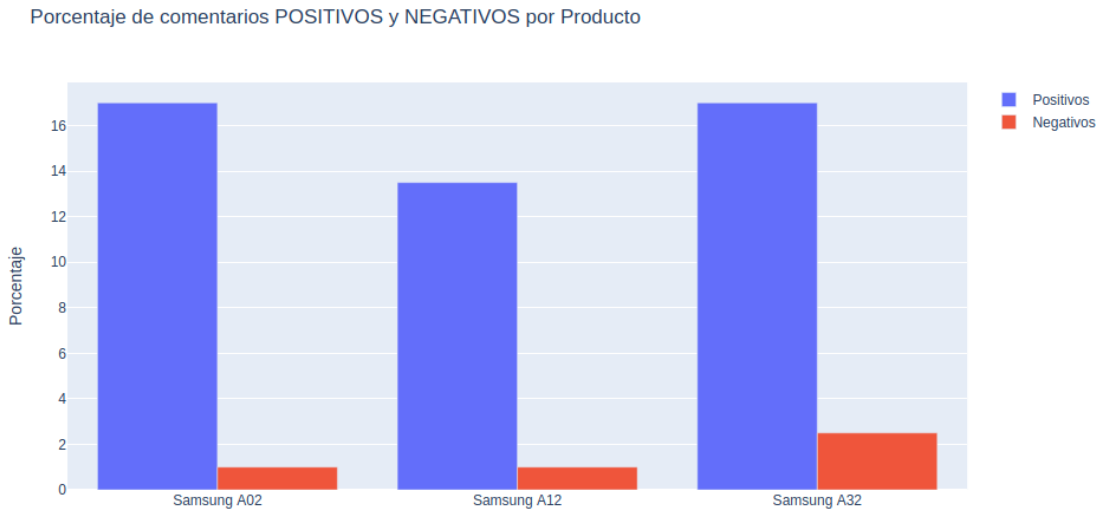
	titulo	comentario	valoracion
28	malo	funciona bien, no es malo, pero se nota que es económico. es de plastico bastante ordinario, opaco. tiene puerto mini usb (y no el más moderno usb 'c'). es bastante lento. por unos pocos pesos más se comprar un par de modelos más arriba y valdría la pena.	2
54	bueno	un celular dentro de todo bueno por el precio, lo malo es la calidad de la. imagen no son tan brillantes, pero safa, y la parte buena es que tiene buena duración de batería.	4
75	buen profucto	muy bien producto para los fines qje necesitaba. trae muchas aplicacionea on board (esi es lo malo que tiene). pero para un uso no laboral, esta bien.	5
164	muy bueno el producto	muy bueno calidad precio producto lo único malo es la calidad de la cámara después todo calidad 10.	5

Esta imagen es solo a modo de ejemplificación, pero sucede exactamente lo mismo con los otros dos dataframes. Surge entonces la siguiente pregunta: **¿Existen otras estructuras de palabras que sean indicativas de si un comentario es positivo o negativo?**

Realizando un mejor análisis se encontraron una serie de palabras que a priori parecen indicar de una manera mas exacta sin un comentario es positivo o negativo.

- Palabras positivas: muy buen celular - el celular es excelente - muy bueno - todo bueno
- Palabras negativas: malísimo - No recomandable – desastre – pobre – malísima - el teléfono es muy malo

El grafico obtenido utilizando esas palabras fue el siguiente. Se puede notar que es muy similar al gráfico anterior, solo que ahora hay mas certeza cuando un comentario es positivo o negativo.



Para ir concluyendo con el análisis, surgió la siguiente pregunta: **¿Cuál es la distribución de palabras más frecuentemente utilizada?**

Esta pregunta me la plantee ya que el hecho de saber cuáles son las palabras más utilizadas me puede dar un indicio de si el producto es bueno o no. Para responder esa pregunta, se utilizaron todas las funciones anteriormente descriptas. Se utilizó la función de preprocesado del texto para eliminar stopwords y demás. Se utilizó la función de contar la frecuencia de las palabras y también se utilizó la función para graficar dichos datos.

```
stop = stopwords.words('spanish')
```

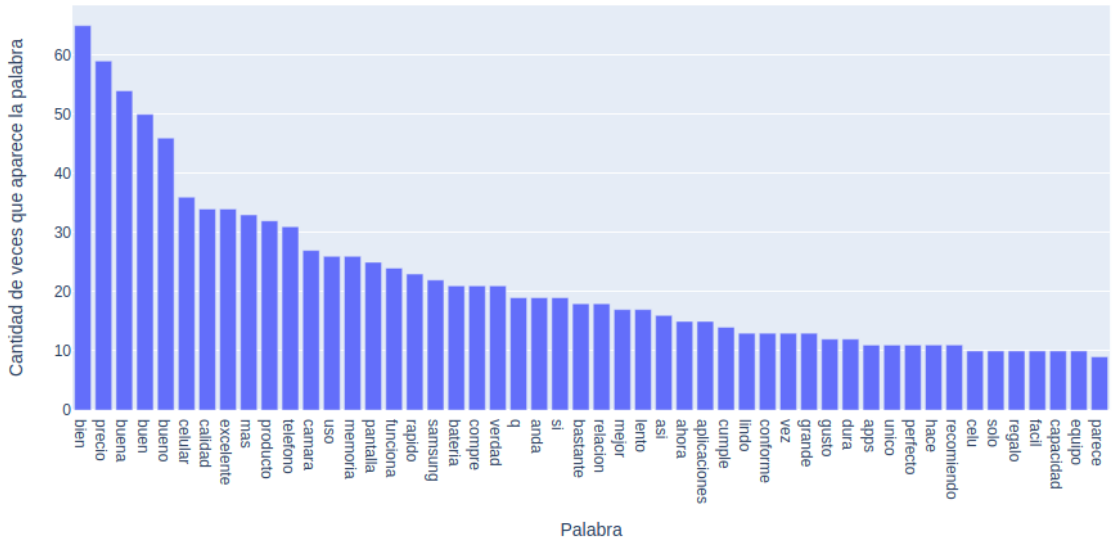
```
df_prod_1_procesado = preprocesar_comentarios_df(df_prod_1)
df_prod_2_procesado = preprocesar_comentarios_df(df_prod_2)
df_prod_3_procesado = preprocesar_comentarios_df(df_prod_3)
```

```
frecuencia_palabras_prod_1 = contar_frecuencia_palabras_df(df_prod_1_procesado)
frecuencia_palabras_prod_2 = contar_frecuencia_palabras_df(df_prod_2_procesado)
frecuencia_palabras_prod_3 = contar_frecuencia_palabras_df(df_prod_3_procesado)
```

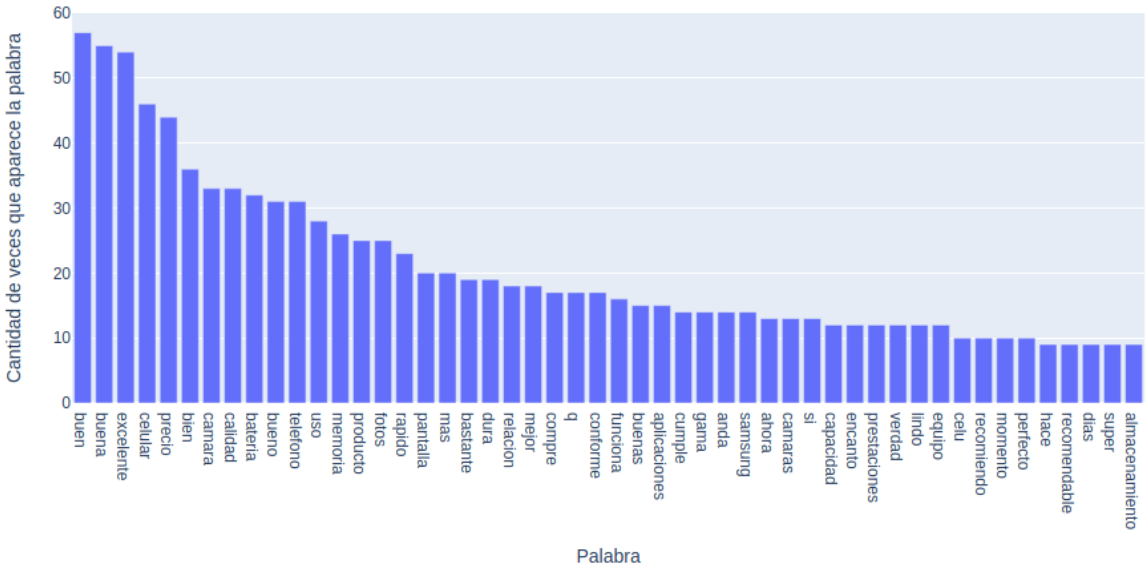
```
plotear_frecuencia_palabras(frecuencia_palabras_prod_1, 50, "Samsung A02")
plotear_frecuencia_palabras(frecuencia_palabras_prod_2, 50, "Samsung A12")
plotear_frecuencia_palabras(frecuencia_palabras_prod_3, 50, "Samsung A32")
```

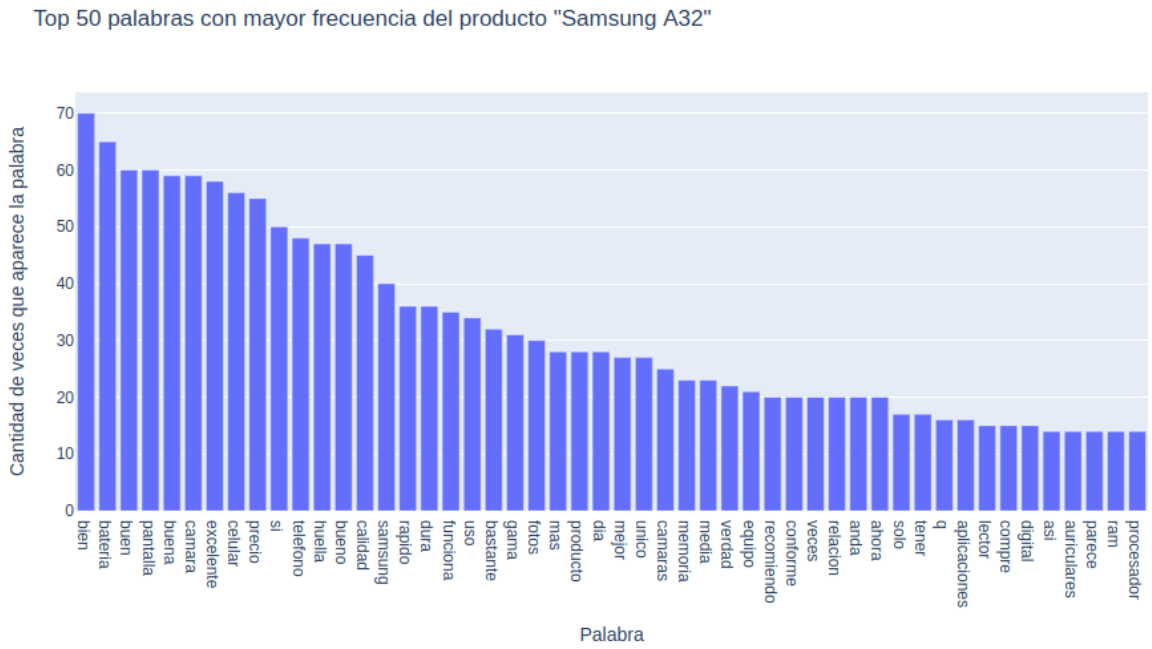
Se obtuvieron los siguientes resultados:

Top 50 palabras con mayor frecuencia del producto "Samsung A02"



Top 50 palabras con mayor frecuencia del producto "Samsung A12"





Una vez finalizado el Análisis Exploratorio de datos, decidí investigar cuales eran los enfoques de Machine Learning que podían ser de utilidad para esta iniciativa.

1.1 Enfoques Basados en Machine Learning

1.1.1 Aprendizaje Supervisado

Los métodos de aprendizaje supervisado dependen de la existencia de documentos de formación etiquetados, es decir, requieren de labels o etiquetas con la salida o el "target" de lo que se quiere predecir. Dentro de estos algoritmos de aprendizaje supervisado encontramos:

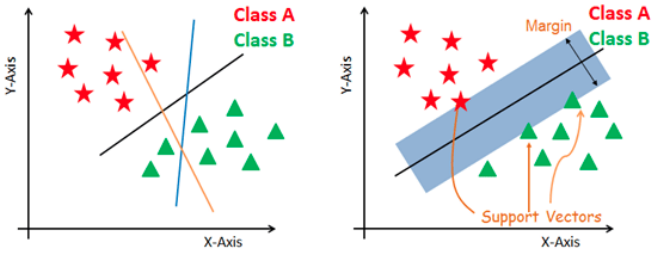
1.1.1.1 Clasificador de Naive Bayes

Es el clasificador más simple y más utilizado. Calcula la probabilidad de una clase, basándose en la distribución de las palabras en el documento. El modelo funciona con la extracción de "features" que ignora la posición de la palabra en el documento y usa el Teorema de Bayes para predecir la probabilidad de que una palabra pertenece a una etiqueta en particular.

$$P(label|features) = \frac{P(label) * P(features|label)}{P(features)}$$

- P(label) es la probabilidad de una etiqueta
- P(features|label) es la probabilidad de que un conjunto de "features" dado se clasifique como una etiqueta.
- P(features)es la probabilidad de que un conjunto de "features" ocurra

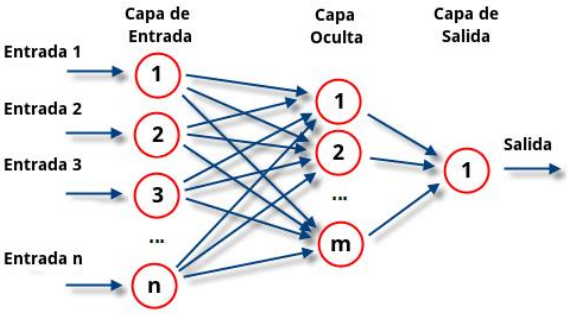
1.1.1.2 Support Vector Machine (SVM)



Separa los puntos de datos utilizando un hiperplano con la mayor cantidad de margen y encuentra un hiperplano óptimo que ayuda a clasificar nuevos puntos de datos. En este sentido, hiperplano hace referencia a un plano o vector de decisión que separa un conjunto de objetos que tienen diferentes pertenencias a una clase. Utiliza “vectores de soporte”, que son los puntos de datos más cercanos al hiperplano. Estos puntos definirán mejor la línea de separación calculando los márgenes en los que puede haber error.

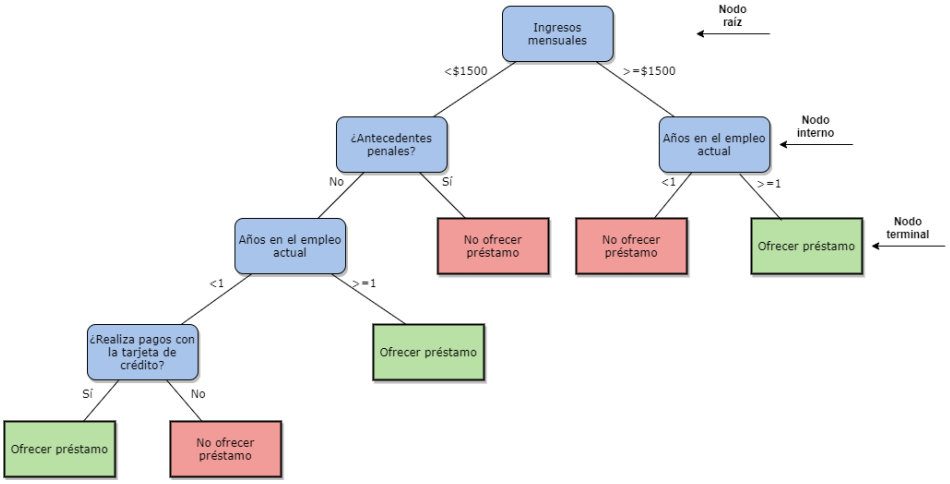
1.1.1.3 Redes Neuronales

Las entradas a las neuronas se denotan mediante un vector que representa la frecuencia de las palabras en el documento. A cada neurona se le asigna un peso numérico, que representa el nivel de importancia de cada neurona. De este peso y de una función de transferencia depende la información de entrada que se transmitirá. Para finalizar **se modela un algoritmo** que genera un resultado para cada input o información de entrada



1.1.1.4 Árboles de decisión

Proporciona una descomposición jerárquica de los datos, en el que se utiliza una condición para dividir los datos, la cual, en este caso, es la presencia o ausencia de una o más palabras. La división de los datos se realiza de forma recursiva hasta que los nodos hoja contengan la suficiente información para realizar la clasificación.



1.1.2 Aprendizaje No Supervisado

Los métodos de aprendizaje no supervisado no dependen de la existencia de etiquetas

2 BIBLIOGRAFÍA

- W. Medhat, A. Hassan, and H. Korashy, “Sentiment analysis algorithms and applications: A survey,” Ain Shams Eng. J., vol. 5, no. 4, pp. 1093–1113, 2014, doi: 10.1016/j.asej.2014.04.011.
- **Sklearn SVM (Support Vector Machines) with Python – DataCamp** [https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python]
- **¿Qué es una neural network? - IONOS** [https://www.ionos.es/digitalguide/online-marketing/marketing-para-motores-de-busqueda/que-es-una-neural-network/]
- **Project Jupyter | Home** [https://jupyter.org/]
- **pandas - Python Data Analysis Library** (pydata.org) [https://pandas.pydata.org/]
- **Plotly Python Graphing Library | Python | Plotly** [https://plotly.com/python/]
- **Requests: HTTP para Humanos — documentación de Requests - 1.1.0 (python-requests.org)** [https://docs.python-requests.org/es/latest/]
- **NLTK :: Natural Language Toolkit** (https://www.nltk.org/)