INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

“Practica 5-
Análisis de Sentimientos”

-Tinoco Videgaray Sergio Ernesto

Grupo: 5BV1

Materia: Tecnologías del Lenguaje Natural

13/01/24

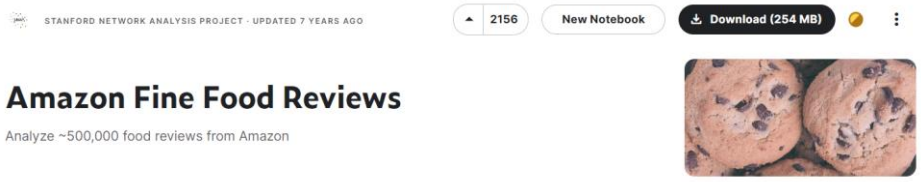
- # Introducción.

En esta práctica se van a implementar varias técnicas para analizar diferentes reseñas sobre comidas de Amazon y determinar el sentimiento que expresa cada reseña.

- # Desarrollo

1. Adquisición de datos.

Para esta práctica se utilizará un conjunto de datos el cual almacena las reseñas en Amazon de consumidores de comida “fina” (Figura 1).



(Figura 1). Vista del dataset desde la página de Kaggle.

2. Análisis exploratorio de datos

Utilizando las herramientas del módulo de Pandas, se van a mostrar los datos del corpus con el que se va a trabajar, así como los metadatos de este.

En primera instancia, se observa que el dataset con el que se va a trabajar en esta práctica contiene 10 columnas (Figura 2):

- Id: ID del registro (Figura 3).
- Product ID: Identificador único del producto (Figura 4).
- UserId: Identificador único para el usuario (Figura 5).
- ProfileName: Nombre de perfil del usuario (Figura 6).
- HelpfulnessNumerator: Número de usuarios que encontraron útil la reseña (Figura 7).
- HelpfulnessDenominator: Número de usuarios que indicaron si encontraron útil la reseña o no (Figura 8).
- Score: Calificación entre 1 y 5 (Figura 9).
- Time: Marca de tiempo para la revisión (Figura 10).
- Summary: Breve resumen de la reseña (Figura 11).
- Text: Texto de la reseña (Figura 12).

```
Data columns (total 10 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Id                      568454 non-null  int64
 1   ProductId               568454 non-null  object
 2   UserId                  568454 non-null  object
 3   ProfileName             568438 non-null  object
 4   HelpfulnessNumerator    568454 non-null  int64
 5   HelpfulnessDenominator  568454 non-null  int64
 6   Score                   568454 non-null  int64
 7   Time                    568454 non-null  int64
 8   Summary                 568427 non-null  object
 9   Text                    568454 non-null  object
```

(Figura 2). Vista de los metadatos del dataset de comida de Amazon.

```
0             1
1             2
2             3
3             4
4             5
           ...
568449    568450
568450    568451
568451    568452
568452    568453
568453    568454
Name: Id, Length: 568454, dtype: int64
```

(Figura 3). Vista de la columna Id.

```
0         B001E4KFG0
1         B00813GRG4
2         B000LQOCH0
3         B000UA0QIQ
4         B006K2ZZ7K
             ...
568449    B001EO7N10
568450    B003S1WTCU
568451    B004I613EE
568452    B004I613EE
568453    B001LR2CU2
Name: ProductId, Length: 568454, dtype: object
```

(Figura 4). Vista de la columna ProductId.

```
0          A3SGXH7AUHU8GW
1          A1D87F6ZCVE5NK
2           ABXLMWJIXXAIN
3          A395BORC6FGVXV
4          A1UQRSCLF8GW1T
                ...
568449     A28KG5XORO54AY
568450     A3I8AFVPEE8KI5
568451     A121AA1GQV751Z
568452      A3IBEVCTXKNOH
568453     A3LGQPJCZVL9UC
Name: UserId, Length: 568454, dtype: object
```

(Figura 5). Vista de la columna UserId.

```
0                               delmartian
1                                   dll pa
2         Natalia Corres "Natalia Corres"
3                                     Karl
4           Michael D. Bigham "M. Wassir"
                        ...
568449                  Lettie D. Carter
568450                        R. Sawyer
568451                    pksd "pk_007"
568452           Kathy A. Welch "katwel"
568453                         srfell17
Name: ProfileName, Length: 568454, dtype: object
```

(Figura 6). Vista de la columna ProfileName.

```
0         1
1         0
2         1
3         3
4         0
         ..
568449    0
568450    0
568451    2
568452    1
568453    0
Name: HelpfulnessNumerator, Length: 568454, dtype: int64
```

(Figura 7). Vista de la columna HelpfulnessNumerator.

```
0         1
1         0
2         1
3         3
4         0
         ..
568449    0
568450    0
568451    2
568452    1
568453    0
Name: HelpfulnessDenominator, Length: 568454, dtype: int64
```

(Figura 8). Vista de la columna HelfulnessDenominator.

```
0         5
1         1
2         4
3         2
4         5
         ..
568449    5
568450    2
568451    5
568452    5
568453    5
Name: Score, Length: 568454, dtype: int64
```

(Figura 9). Vista de la columna Score.

```
0         1303862400
1         1346976000
2         1219017600
3         1307923200
4         1350777600
              ...
249995    1341619200
249996    1341532800
249997    1341446400
249998    1341446400
249999    1344902400
Name: Time, Length: 250000, dtype: int64
```

(Figura 10). Vista de la columna Time.

```
0                       Good Quality Dog Food
1                           Not as Advertised
2                       "Delight" says it all
3                              Cough Medicine
4                                 Great taffy
                              ...
568449                    Will not do without
568450                            disappointed
568451              Perfect for our maltipoo
568452    Favorite Training and reward treat
568453                             Great Honey
Name: Summary, Length: 568454, dtype: object
```

(Figura 11). Vista de la columna Summary.

```
0         I have bought several of the Vitality canned d...
1         Product arrived labeled as Jumbo Salted Peanut...
2         This is a confection that has been around a fe...
3         If you are looking for the secret ingredient i...
4         Great taffy at a great price.  There was a wid...
                                ...
568449    Great for sesame chicken..this is a good if no...
568450    I'm disappointed with the flavor. The chocolat...
568451    These stars are small, so you can give 10-15 o...
568452    These are the BEST treats for training and rew...
568453    I am very satisfied ,product is as advertised,...
Name: Text, Length: 568454, dtype: object
```

(Figura 12). Vista de la columna Text.

De las columnas mostradas y analizadas, se va a trabajar únicamente con las columnas de Score y Text ya que son las que contienen la información relevante para fines de esta práctica, por lo que se van a omitir las demás columnas.

De igual forma, haciendo un breve análisis se puede observar que el dataset cuenta con mas de 500,000 registros, lo cual es una cantidad masiva de datos para procesar, por lo que para fines de esta práctica se van a trabajar con únicamente 2000 registros por cada sentimiento (6000 registros en total).

3. **Preprocesamiento**.

Una vez cargado y analizado el dataset en el programa, se va a trabajar únicamente con las columnas correspondientes al texto de la reseña y a la puntuación asignada al producto por el cliente.

Una vez seleccionadas dichas columnas se procederán con la obtención de los sentimientos basado en la puntuación que recibió cada producto en una escala del 1 al 5. Por lo que si se le dio una puntuación de más de 3 se le considera como positiva y si recibe una puntuación menor de 3 se le

considera negativa y en caso de ser exactamente 3 se le considera neutral. Dichos sentimientos van a servir como clases o etiquetas para los modelos a implementar posteriormente (Figura 13).

```
df = df[['Text', 'Score']]
df['Sentiment'] = df['Score'].apply(lambda x: 'positive' if x > 3
                                    else 'negative' if x < 3
                                    else 'neutral')
```

(Figura 13). Código donde se obtienen las polaridades de cada reseña basándose en la puntuación que recibió el producto.

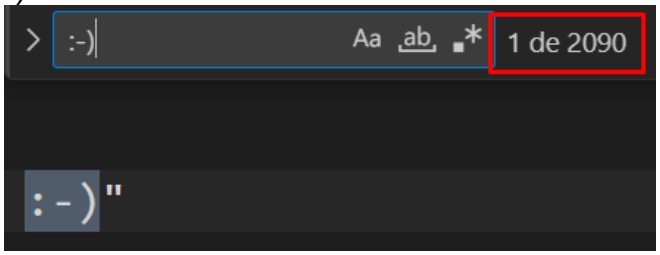
Para ayudar a los modelos a que se van a implementar más adelante obtener una buena exactitud, se va a realizar un balance de clases a través de una función que va a recolectar el mismo número de reseñas tanto positivas como negativas y neutras (Figura 14).

```
df_b = pd.DataFrame({'Text': [],
                     'Sentiment': []})
p,n,neu=0,0,0   #Contadores
s=samples #Muestras por clase
for row in range(len(dataframe)):
    sentiment=dataframe.iloc[row,1]
    if sentiment == "positive" and p<s:
        df_b=df_b._append([dataframe.loc[row,["Text","Sentiment"]]],
        p+=1
    elif sentiment == "negative" and n<s:
        df_b=df_b._append([dataframe.loc[row,["Text","Sentiment"]]],
        n+=1
    elif sentiment == "neutral" and neu<s:
        df_b=df_b._append([dataframe.loc[row,["Text","Sentiment"]]],
        neu+=1
    else:
        if p<s or n<s or neu<s:
            continue
```

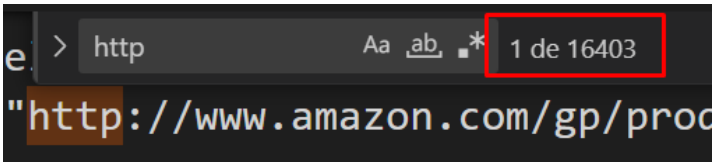
(Figura 14). Código de la función que balancea el número de muestras de cada sentimiento.

4. Limpieza de datos

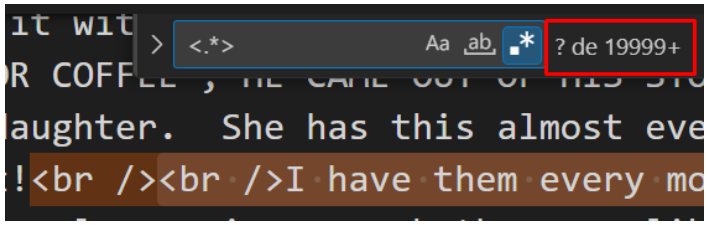
En esta etapa se van a omitir ciertos caracteres que puedan generar ruido a los modelos.
Haciendo un breve análisis con el editor de Visual Studio Code se puede observar la presencia de ciertos caracteres como los emojis, etiquetas de HTML, números, direcciones, caracteres especiales, signos de puntuación, etc (Figura 15 a 21).



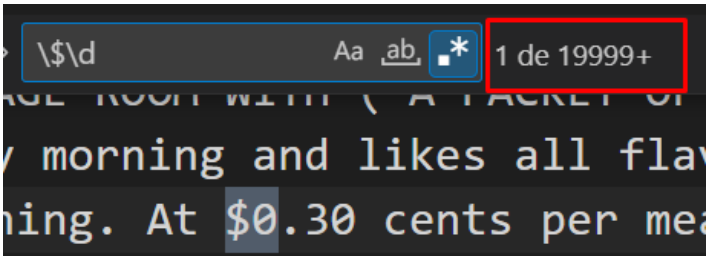
(Figura 15). Búsqueda de emojis en el corpus.



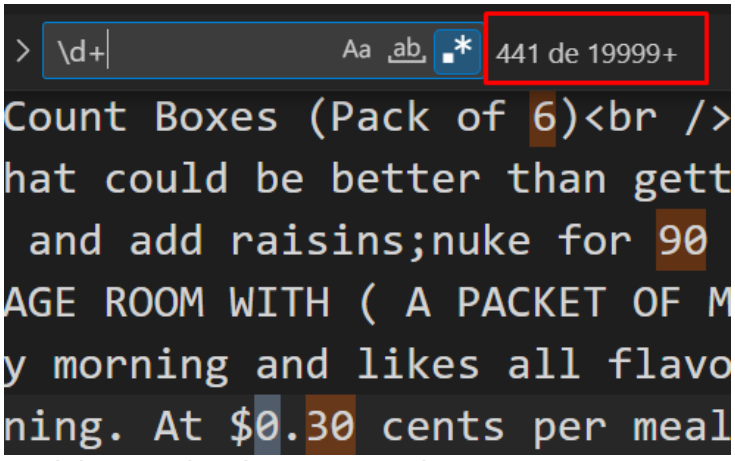
(Figura 16). Búsqueda de hipervínculos en el corpus.



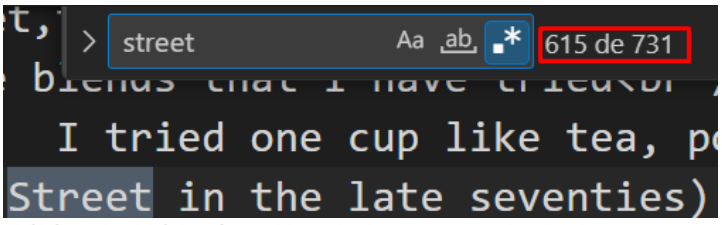
(Figura 17). Búsqueda de etiquetas HTML en el corpus.



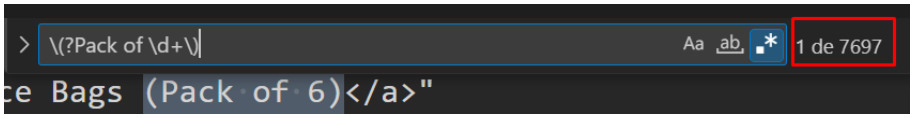
(Figura 18). Búsqueda de precios en el corpus.



(Figura 19). Búsqueda de números en el corpus.



(Figura 20). Búsqueda de direcciones en el corpus.

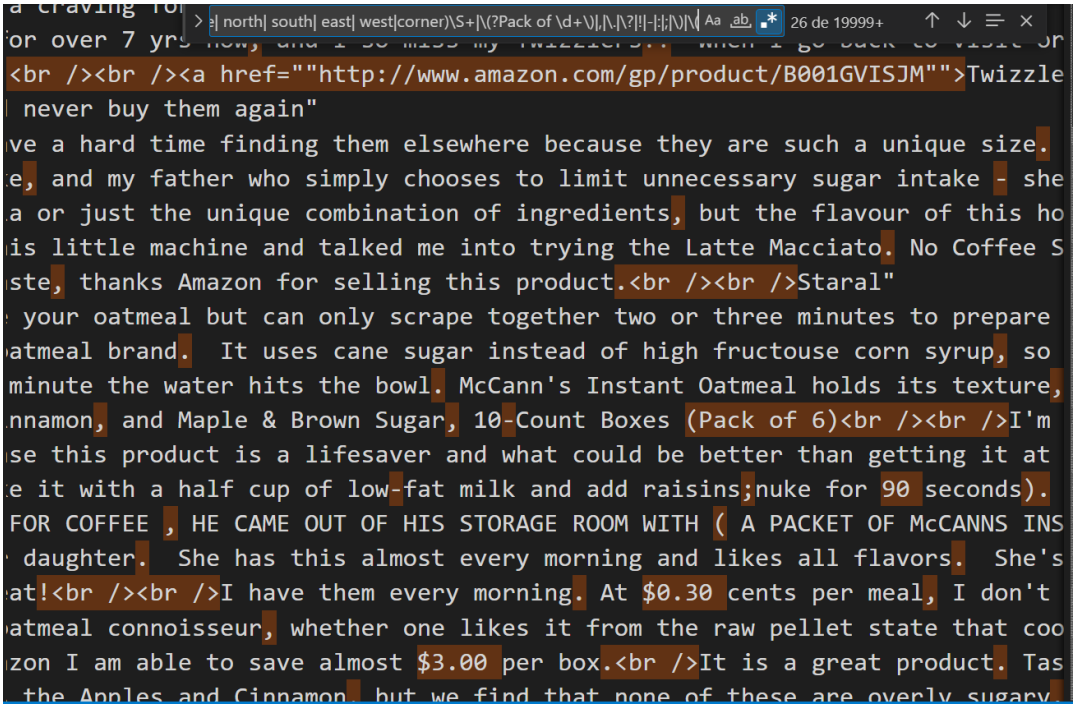


(Figura 21). Búsqueda de descripciones de cada paquete en el corpus.

Para ello se van a omitir dichos caracteres por medio de una expresión regular que englobe los ya mencionados anteriormente (Figura 22 y 23).

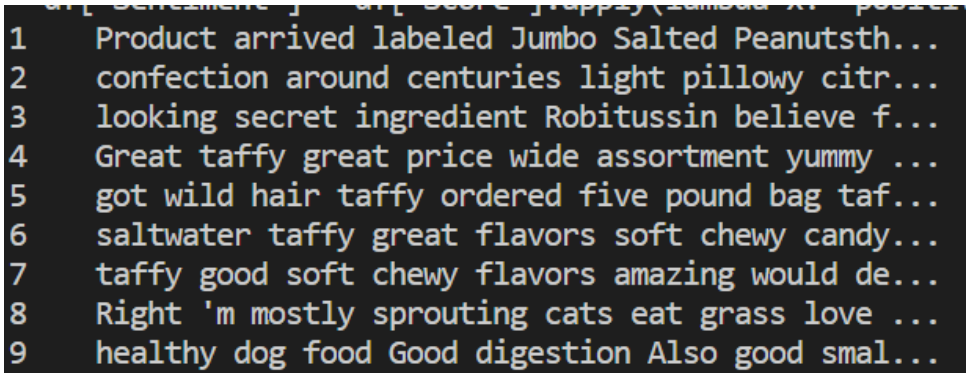
```
for reseña in dataframe["Review"]:
    reg_ex=':-\)|:\)|: \)|<.{1,2} />|<a.+">|</a>|\$?\d+\.?\d*\S \
    |( street|avenue| north| south| east| west|corner)\S+|\(?Pack of \d+\)\
        |,|\.|\?|!|-|:|;|\)|\(|"'+"|\s'"+"|'\s"+"|`"
    text=re.sub(reg_ex, '', reseña)
    tokens = word_tokenize(text)
    tokens= [w for w in tokens if not w.lower() in stop_words]
    text=' '.join(tokens)
    text_clean.append(text)
dataframe["Review"]=text_clean
```

(Figura 22). Código de la función donde se aplica una expresión regular para eliminar caracteres y cadenas que puedan generar ruido a los modelos.



(Figura 23). Vista del corpus donde se utiliza la expresión regular a utilizar para eliminar caracteres que puedan generar ruido en los modelos.

Una vez aplicada dicha expresión regular para limpiar el texto de estos caracteres se puede observar la ausencia de estos símbolos en las reseñas de los productos (Figura 24).



(Figura 24). Vista del corpus una vez aplicada la limpieza.

5. Transformación / normalización de datos

En esta etapa se va a realizar la normalización de los datos para de igual forma, darle un formato a los datos que sea adecuado para los modelos y las técnicas que se van a implementar en una etapa posterior.

Para ello se va a realizar un proceso de tokenización, stemming, POS tagging y lematización, así como también obtener las representaciones vectoriales de las reseñas por medio de TF-IDF y Hot Encoding (Figura 25 y 26).

```
for reseña in dataframe["Review"]:
    #Tokenizar
    tokens=word_tokenize(str(reseña).lower())
    #Aplicar stemming
    tokens=[ps.stem(token) for token in tokens]
    #Aplicar POS-tagging
    tokens=pos_tag(tokens)  #Usando metodo pos tag
    #Reasignar etiquetas con formato wordnet
    tokens= list(map(lambda x: (x[0], pos_tagger(x[1])), tokens))
    #Aplicar lematizacion utilizando las etiquetas asignadas en el POS-tagging
    tokens=[lematizer.lemmatize(token,tag) for token,tag in tokens]
    text_normalized.append(' '.join(tokens))
#Definir tf-idf vectorizer
vectorizer = TfidfVectorizer()
#Ajustar y transformar vectorizador
X = vectorizer.fit_transform(text_normalized)
#Obtener tokens del diccionario
tokens = vectorizer.get_feature_names_out()
```

(Figura 25). Código donde se normaliza el corpus.

	Text	Sentiment	Text_Vectorized
0	bought several vitality canned dog food produc...	positive	[buy, sever, vital, can, dog, food, product, f...
1	product arrived labeled jumbo salted peanutsth...	negative	[product, arriv, label, jumbo, salt, peanutsth...
2	confection around centuries light pillowy citr...	positive	[confect, around, centuri, light, pillowi, cit...
3	looking secret ingredient robitussin believe f...	negative	[look, secret, ingredi, robitussin, believ, fi...
4	great taffy great price wide assortment yummy ...	positive	[great, taffi, great, price, wide, assort, yum...

(Figura 26). Vista del corpus una vez normalizado el texto.

Haciendo un nuevo análisis exploratorio del dataframe normalizado se puede observar que ahora se cuentan con 3 columnas (Figura 27).

- Text: Contiene el texto normalizado de la reseña (Figura 28).
- Sentiment: Almacena la polaridad correspondiente de cada reseña (Figura 29).
- TD_IDF: almacena el vector TF-IDF de cada reseña (Figura 30).

```
RangeIndex: 6000 entries, 0 to 5999
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  6000 non-null   int64
 1   Review      6000 non-null   object
 2   Sentiment   6000 non-null   object
 3   TF_IDF      6000 non-null   object
```

(Figura 27). Vista del dataframe normalizado.

```
0       excel nice rich creami pistachio nut middl mmm...
1       bad flavor coffe ever tri aroma flavor chocola...
2       cereal best cereal market mani flavor includ n...
3       buy nutiva organ chia seed two pouch ounc tax ...
4       receiv unders sickli plant could nurtur back h...
                              ...
5995    bring 2 bottl one carri pocket home fell love ...
5996    find two famili member celiac diseas give lico...
5997    know chocol turn light brown get old 's top li...
5998    look nutriti snack fiber come across 've tri v...
5999    order wolf gang puck 's k cup larg reput exper...
Name: Review, Length: 6000, dtype: object
```

(Figura 28). Vista de la columna Review.

```
0       positive
1       negative
2       positive
3       negative
4       negative
          ...
5995    positive
5996    positive
5997    negative
5998    positive
5999    negative
Name: Sentiment, Length: 6000, dtype: object
```

(Figura 29). Vista de la columna Sentiment.

```
0       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
1       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
2       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
4       [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
                                ...
5995    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5996    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5997    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5998    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
5999    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
Name: TF_IDF, Length: 6000, dtype: object
```

(Figura 30). Vista de la columna TF-IDF.

6. Análisis de sentimientos usando diccionarios:

En esta etapa se van a implementar 2 técnicas de análisis de sentimientos utilizando diccionarios los cuales permiten asignarle un puntaje a cada reseña utilizando un conjunto de listas de palabras que ya tienen predefinidos los términos positivos y negativos.

- Harvard IV-4.

Primeramente, se va a utilizar el diccionario de Harvard en su versión 4 el cual va a asignar un puntaje a cada texto basándose en su propio conjunto de términos positivos y negativos, dicho puntaje va a estar definido en un rango de -1 a 1 siendo -1 un sentimiento negativo, 1 un sentimiento positivo y 0 un sentimiento neutro (Figura 32 y 33).

```
for reseña in reseñas:
    tokens = hiv4.tokenize(reseña)
    scores.append(hiv4.get_score(tokens)["Polarity"])
dataframe["Harvard_Polarity"]=scores

print(dataframe.loc[:,["Sentiment","Harvard_Polarity"]])
predict = dataframe["Harvard_Polarity"].apply(lambda x: 2 if x > 0.4
                                              else 0 if x < -0.4
                                              else 1)
```

(Figura 32). Código de la función donde se obtienen los puntajes de cada reseña utilizando el diccionario Harvard 4.

```
Matriz de Confusión:
[[ 110 1181  709]
 [  79 1019  902]
 [  70  796 1134]]
Exactitud del modelo:
0.37716666666666665
     Sentiment Harvard_Polarity
0     negative          neutral
1     positive         positive
2     positive          neutral
3     negative         positive
4     negative         negative
...        ...              ...
5995  negative         positive
5996  positive         positive
5997   neutral         positive
5998   neutral          neutral
5999  negative         positive
```

(Figura 33). Visualización de la exactitud del modelo por medio de la matriz de confusión y la comparación entre la columna Sentiment con la polaridad asignada por Harvard 4.

Como se puede observar la técnica utilizando el diccionario de Harvard 4 obtuvo una exactitud del 37.71% lo cual es un modelo bastante pobre.

Visualizando la matriz de confusión se observa que 110 reseñas negativas fueron precedidas de manera correcta, 1019 reseñas neutrales fueron acertadas y 1134 reseñas positivas fueron estimadas correctamente.

- Opinion Lexicon.

De forma análoga se va a utilizar el diccionario de opinión lexicón que se encuentra en el paquete de nltk.

A diferencia de Harvar 4, opinión lexicón no asigna directamente un puntaje a cada texto, sino que cuenta con una lista de términos positivos y una lista de términos negativos. Es por ello que, para obtener el puntaje de cada reseña en el corpus, se van a contar el número de términos positivos y negativos en cada reseña y la categoría que tenga más términos en la reseña será la que determine el sentimiento de la reseña (Figura 34 y 35).

```
for reseña in reseñas:
    tokens = hiv4.tokenize(reseña)
    lst_positive=opinion_lexicon.positive()
    lst_negative=opinion_lexicon.negative()
    p = sum(el in tokens for el in lst_positive)
    n=sum(el in tokens for el in lst_negative)
    sentiment.append("positive" if p>n else "negative" if p<n else "neutral")
```

(Figura 34). Código de la función que calcula la polaridad de cada reseña utilizando el diccionario de opinión lexicón.

```
Matriz de Confusión:
[[ 776  570  654]
 [ 497  579  924]
 [ 260  502 1238]]
Exactitud del modelo:
0.43216666666666664
     Sentiment OP_Lexicon_Polarity
0     negative            negative
1     positive            positive
2     positive            positive
3     negative             neutral
4     negative             neutral
...        ...                 ...
5995  negative            negative
5996  positive            positive
5997   neutral             neutral
5998   neutral            positive
5999  negative            positive

[6000 rows x 2 columns]
```

(Figura 35). Visualización de la exactitud del modelo por medio de la matriz de confusión y la comparación entre la columna Sentiment con la polaridad obtenida por medio de opinión lexicón.

Como se puede observar, la técnica utilizando el diccionario de opinión lexicón obtuvo una exactitud de 43.21% lo cual es un modelo medianamente malo a pesar de que la polaridad asignada por Opinion Lexicon coincide bastante con la polaridad asignada a través del puntaje que recibió el producto con un pequeño margen de error.

Visualizando la matriz de confusión se observa que 776 reseñas negativas fueron precedidas de manera correcta, 579 reseñas neutrales fueron acertadas y 1238 reseñas positivas fueron estimadas correctamente.

7. Análisis de sentimientos usando algoritmos de aprendizaje de máquina:

En esta etapa se va a realizar el análisis de sentimiento de las reseñas en el corpus utilizando diferentes modelos de aprendizaje de máquina, para ello se va a hacer uso de los vectores TF-IDF de cada reseña.

Para una correcta implementación, se va a dividir el conjunto de datos en prueba y entrenamiento, siendo el 80% de las reseñas asignado para el entrenamiento de cada modelo y el 20% restante para pruebas (Figura 36).

```
#Dividir conjunto de datos en pruebas y entrenamiento
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data, label, \
test_size=0.2,random_state=3)
```

(Figura 36). Código donde se divide el conjunto de datos en prueba y entrenamiento.

- Regresión logística.

La regresión logística es un modelo basado en una función sigmoidal, por lo que es usada comúnmente para clasificación de 1 o 2 clases. Sin embargo, también puede ser adaptado para clasificación multiclase con diferentes enfoques como softmax que obtiene la regresión logística de cada clase y las normaliza para que la suma de estas sea de 1 (Figura 37 y 38).

```
# Creación del modelo
lr = LogisticRegression(multi_class='multinomial')
# Ajustamos el modelo a los datos de entrenamiento
lr = lr.fit(X_train, y_train)
# Realizamos prediciones
y_pred = lr.predict(X_test)
```

(Figura 37). Código donde se crea y ajusta el modelo de la regresión logística multinomial.

```
Matriz de Confusión:
[[279  88  32]
 [ 89 256  51]
 [ 31  50 324]]
Exactitud del modelo:
0.7158333333333333
     Sentiment LR_Polarity
0     positive    positive
1     negative    negative
2     positive    positive
3     negative    negative
4     negative    negative
...        ...         ...
5995  positive    positive
5996  positive    positive
5997  negative    positive
5998  positive    positive
5999  negative    negative

[6000 rows x 2 columns]
```

(Figura 38). Vista de la exactitud del modelo por medio de la matriz de confusión y las polaridades estimadas.

Como se puede observar el modelo de regresión logística multinomial obtuvo una exactitud del 71.53% la cual es una muy buena exactitud hasta ahora.
Visualizando la matriz de confusión se observa que 279 reseñas negativas fueron precedidas de manera correcta, 256 reseñas neutrales fueron acertadas y 324 reseñas positivas fueron estimadas correctamente.

- Arboles de decisiones.

Los árboles de decisiones son un modelo utilizado tanto para clasificación como para regresión, estos modelos utilizan los árboles binarios para hacer la clasificación de acuerdo con ciertos criterios como el umbral de división y la característica asociada a cada nodo (Figura 39 y 40).

```
#Crear instancia de la clase decision tree clasifier
dtC = DecisionTreeClassifier()
#Ajustar modelo al conjunto de entrenamiento
dtC.fit(X_train, y_train)
#Realizar la inferencia de datos con el modelo ajustado previamente
y_pred=dtC.predict(X_test)
```

(Figura 39). Código donde se crea y se ajusta el modelo de árboles de decisión para la clasificación.

```
Matriz de Confusión:
[[802 469 347]
 [504 697 385]
 [337 399 860]]
Exactitud del modelo:
0.4914583333333333
     Sentiment DT_Polarity
0     negative    negative
1     positive    positive
2     positive    positive
3     negative    positive
4     negative    negative
...        ...         ...
5995  negative    negative
5996  positive    positive
5997   neutral    positive
5998   neutral     neutral
5999  negative     neutral
```

(Figura 40). Vista de la exactitud del modelo de árboles de decisión por medio de la matriz de confusión y las polaridades estimadas.

En este caso la exactitud del modelo ronda alrededor del 49.14 lo que implica un modelo aceptable si lo comparamos con la regresión logística multinomial.
Visualizando la matriz de confusión se observa que 802 reseñas negativas fueron precedidas de manera correcta, 697 reseñas neutrales fueron acertadas y 860 reseñas positivas fueron estimadas correctamente.

- Máquinas de Soporte Vectorial.

De igual forma las máquinas de soporte vectorial o SVM por sus siglas en ingles son utilizadas tanto para clasificación como para regresión, generan un conjunto de vectores llamados vectores de soporte y un hiperplano el cual va a dividir de forma óptima al conjunto de datos en una dimensión n. En el caso de la clasificación, el hiperplano va a dividir el conjunto de datos maximizando la distancia con los vectores de soporte. Algunos modelos pueden utilizar un kernel que les ayuda a ajustarse mejor a datos que no siguen un comportamiento binario (Figura 41 y 42).

```
#Entrenar el modelo SVM
svC=SVC()
svC.fit(X_train,y_train)
#Prediccion del test de prueba
y_pred=svC.predict(X_test)
```

(Figura 41). Código donde se crea y se ajusta el modelo de las maquinas de soporte vectorial para clasificación.

```
Matriz de Confusión:
[[242 113  41]
 [ 88 242  54]
 [ 30 108 282]]
Precisión del modelo:
0.6383333333333333
     Sentiment SVM_Polarity
0     negative     positive
1     positive     positive
2     positive     positive
3     negative     positive
4     negative     positive
...        ...          ...
5995  negative     positive
5996  positive     positive
5997   neutral     positive
5998   neutral     positive
5999  negative     positive

[6000 rows x 2 columns]
```

(Figura 42). Vista de la exactitud del modelo de maquinas de soporte vectorial por medio de la matriz de confusión y las polaridades estimadas por el modelo.

En este caso el modelo de maquinas de soporte vectorial obtuvo una exactitud del 63.83% lo cual es un modelo bueno considerando que no se utilizó ningún kernel para la clasificación.

Visualizando la matriz de confusión se observa que 242 reseñas negativas fueron precedidas de manera correcta, 242 reseñas neutrales fueron acertadas y 282 reseñas positivas fueron estimadas correctamente.

8. Análisis de sentimientos usando word embbedings y redes neuronales:

Finalmente se va a implementar un modelo de red neuronal para el análisis de sentimiento. Dicha red neuronal va a consistir en 3 capas (Figura 43 y 44):

- La capa de entrada la cual va a recibir el vector hot encoding que corresponde al embedding del texto de la reseña.
- La capa oculta que consiste en una capa convolucional y max pooling para reducir el mapa de características y una función de activación relu.
- La capa de salida que va a aplicar la función softmax para obtener las probabilidades multiclase a la cual va a pertenecer el texto.

```
model = Sequential()
model.add(Embedding(5000, 100, input_length=100))
model.add(Conv1D(64, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

(Figura 43). Código donde se define la arquitectura de la red neuronal.

```
Epoch 12/13
150/150 [==============================
.3767 - val_accuracy: 0.7017
Epoch 13/13
150/150 [==============================
.4850 - val_accuracy: 0.6975
38/38 [==============================]
Precision: 0.6975
188/188 [==============================
     Sentiment NN_Polarity
0     positive    positive
1     negative    negative
2     positive    positive
3     negative    positive
4     negative    negative
...        ...         ...
5995  positive    positive
5996  positive    positive
5997  negative    negative
5998  positive    positive
5999  negative    negative

[6000 rows x 2 columns]
```

(Figura 44). Vista de la precisión del modelo de red neuronal calculado a partir de la función de perdida y la polaridad estimada por el modelo.

En este caso, el modelo de la red neuronal obtuvo una precisión de 69.75% que es casi tan bueno como el modelo de la regresión logística multinomial, aun así, este podría considerarse un buen modelo para clasificación de textos con base en la polaridad.

Finalmente se exporta el dataframe con las polaridades estimadas (Figura 45 y 46).

```
#Exportar resultados
df_p=df.loc[:,["Sentiment","Harvard_Polarity","OP_Lexicon_Polarity",\
            "LR_Polarity","DT_Polarity","SVM_Polarity","NN_Polarity"]]
df_p.to_csv('dataframe_predicted.csv')
```

(Figura 45). Código donde se exporta el dataframe como archivo .csv

dataframe_predicted.csv

```
,Sentiment,Harvard_Polarity,OP_Lexicon_Polarity,LR_Polarity,DT_Polarity,SVM_Polarity,
0,negative,neutral,negative,negative,negative,positive,negative
1,positive,positive,positive,positive,positive,positive,positive
2,positive,neutral,positive,positive,positive,positive,positive
3,negative,positive,neutral,negative,positive,positive,positive
4,negative,negative,neutral,negative,negative,positive,negative
5,negative,neutral,neutral,negative,negative,positive,negative
6,positive,positive,neutral,positive,positive,positive,positive
7,positive,positive,positive,positive,positive,positive,positive
8,positive,positive,positive,neutral,neutral,positive,positive
9,neutral,neutral,negative,neutral,neutral,positive,neutral
10,neutral,neutral,positive,neutral,neutral,positive,neutral
11,neutral,positive,positive,neutral,positive,positive,neutral
12,positive,neutral,positive,positive,positive,positive,positive
13,neutral,negative,negative,neutral,neutral,positive,negative
```

(Figura 46) Vista del archivo .csv exportado.

Conclusiones

En esta práctica se realizaron varias técnicas de análisis de sentimientos las cuales permiten clasificar el texto de varias reseñas de comidas de Amazon, y se puede aplicar para muchas áreas como la Psicología, las redes sociales, la atención al cliente, la criminología, la educación, etc.

Los primeros 2 enfoques se realizaron utilizando diccionarios los cuales van a asignar un puntaje a cada reseña basándose en los términos que se mencionan y en una lista de palabras preestablecidas anteriormente para determinar la polaridad del texto.

Los primeros 3 enfoques utilizando modelos de aprendizaje de maquina se implementaron haciendo uso de los vectores TF-IDF de cada reseña, y si bien no todos dieron buenos resultados, son considerablemente precisos al ajustarse a los datos de manera directa.

Por último, la red neuronal fue un modelo bastante bueno, aunque ciertamente podría mejorar al ajustar ciertos hiperparámetros como las épocas, el tamaño del lote, o el tamaño de los vectores, se adaptó a los datos que se le asignaron de manera rápida y precisa.