Librerias

```
# Tratamiento de la Información
import numpy as np
import pandas as pd
import string
import re
from datetime import datetime

# Gráficos de análisis
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

# Preprocesado y modelado
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk

nltk.download('stopwords')
from nltk.corpus import stopwords
```

Cargar y Transformar Base de Datos

```
# Cargar la base de datos
Hey   = pd.read_csv("Datathon_HeyBanco.csv")

# Modificar tipo de datos de columnas
Hey['date'] = pd.to_datetime(Hey['date'])

# Agregar Columnas de Fechas
Hey["year"] = Hey["date"].dt.year
Hey["month"] = Hey["date"].dt.month

#Visualizar los primeros registros
Hey.head()
```

```
/usr/local/lib/python3.10/dist-packages/google/colab/_dataframe_summarizer.py:88: UserWa
  cast_date_col = pd.to_datetime(column, errors="coerce")
```

	date	time	tweet	year	month
0	2023-01-01	NaN	Resuelto, muchísimas gracias , excelente servi...	2023	1
1	2023-01-02	NaN	Muchas gracias, espero su dm	2023	1
2	2023-01-02	NaN	Muchas gracias!	2023	1
3	2023-01-02	NaN	Algo similar me paso. Quería renovar mi token ...	2023	1
4	2023-01-02	NaN	Yeeeei! a través de mi cuenta en acabo de cont...	2023	1

Eliminar Acentos

```
# Eliminar acentos de los Tweets
Hey['tweet'] = Hey['tweet'].str.normalize('NFKD').str.encode('ascii', errors='ignore').str.decode('utf-8')

Hey.head()
```

Proceso de Tokenización

```
# Proceso de Tokenizer
def tokenizar(texto):

    # Se convierte todo el texto a minúsculas
    n_texto = texto.lower()
    # Eliminación de páginas web
    n_texto = re.sub('http\S+', ' ', n_texto)
    # Eliminación de signos de puntuación
    regex = '[\\!\\"\\#\\$\\%\\&\\'\\(\\)\\*\\+\\,\\-\\.\\/\\:\\;\\<\\=\\>\\?\\@\\[\\\\\\]\\^_\\`\\{\\|\\}\\~]'
    n_texto = re.sub(regex , ' ', n_texto)
    # Eliminación de números
    n_texto = re.sub("\d+", ' ', n_texto)
    # Eliminación de espacios en blanco múltiples
    n_texto = re.sub("\\s+", ' ', n_texto)
    # Tokenización por palabras individuales
    n_texto = n_texto.split(sep = ' ')
    # Eliminación de tokens con una longitud < 3
    n_texto = [token for token in n_texto if len(token) > 2]

    return(n_texto)

# Tokenizar Bases de Datos
Hey['texto_tokenizado'] = Hey['tweet'].apply(lambda x: tokenizar(x))


db = Hey.explode(column='texto_tokenizado')
# db = db.drop(columns='tweet')
db = db.rename(columns={'texto_tokenizado':'token'})


# Contar las palabras usadas x Mes
db.groupby(by= ['year','month'])['token'].count()
```

Categorización y sentimientos de Tweets

Categorización x Razón de Tweet

```
def words (texto):
  lower = texto.lower()
  no_numbers = re.sub(r'\d+','',lower)
  no_puc= re.sub(r"[^\w\s]",'', no_numbers)
  no_space = no_puc.strip()
  word = [no_space][0].split()
  stop_words = set(stopwords.words("english"))
  no_stopwords_str = ""

  for i in word:
    if not i in stop_words:
      no_stopwords_str += i+ " "

  no_stopwords_str = no_stopwords_str[:-1]
  word = [no_space][0].split()

  return no_stopwords_str

Hey['texto_tokenizado2'] = Hey['tweet'].apply(lambda x: words(x))


# Definir listas de palabras clave para cada Razón del Tweet

Servicios_Financieros = ['transacciones', 'transferir', 'activacion','contratacion','retiro','compra',
                         'banco', 'cuenta', 'tarjeta', 'credito', 'inversion', 'transferencias', 'debito', 'anualidad',
                         'cashback', 'tdc', 'tdd','dinero', 'inversion', 'rendimiento', 'interes', 'pagos', 'plazo',
                         'comisiones', 'inflacion', 'prestamo', 'seguro','servicio','ahorro','nomina']
Aplicaciones_Plataformas = ['app', 'uala', 'heybanco', 'heymedia', 'heycoins', 'heypoints', 'heygarantizada', 'heygpt', 'heypro', 'finsus','
Tecnologia = ['fintech', 'neobanks', 'blockchain', 'crypto', 'token', 'api', 'digital','innovacion', 'tecnologia', 'avance', 'desarrollo','t
Bancos = ['banamex', 'bbva', 'scotiabank', 'santander', 'bancomer', 'banorte', 'coppel']
Experiencia_Cliente = ['ayuda', 'contacto', 'chat', 'correo','interfaz', 'experiencia', 'facilidad','uso','funcionamiento','proceso',
                       'usabilidad', 'usuario', 'atencion', 'opinion', 'feedback', 'satisfaccion', 'problema','solucion', 'resuelto','respue
Promocion = ['promocion', 'publicidad', 'oferta', 'descuento', 'beneficios', 'recompensas', 'regalo', 'cupon', 'anuncio', 'boletos',
             'playera','playeras','gano','ganar']
Tiempo_Eventos = ['dia', 'dias', 'hoy', 'semana', 'mes', 'meses', 'año', 'anual', 'temporada', 'momento','años','tiempo','limitado']
Agradecimiento = ['gracias', 'muchas', 'agradece', 'agradecida', 'agradecido', 'agradables', 'agradecimiento', 'felicidades',
                  'excelente', 'recomendado', 'satisfaccion','thks','feliz','muy']
Servicio_Cliente = ['atencion', 'ayuda', 'problema', 'solucion', 'respuesta', 'cliente','confirmacion','espero','espera','mensaje',
                    'llamar','chat','dm','sugerencias','sugerencia','atento','comentarios','comunicacion','opiniones']

Beneficios = ['cashback', 'beneficios', 'promocion', 'premio', 'recompensas', 'descuentos','descuento','gano','gane','ganar',
              'pago','oxxo','tienda','luz','internet']
Seguridad_Confianza = ['seguridad', 'confianza', 'proteccion', 'garantia', 'privacidad', 'confidencialidad','seguro','informacion',
                       'conocimiento','educacion','financiera','transparencia','asesoria','orientacion']


#Inicializar Categoría y % de Cumplimiento
Cat = None
Porcentaje = 0
```

```
# Función de Clasificación de Texto
def classify_text (texto):
  Serv_Fin = 0
  App = 0
  Tecno = 0
  Banco = 0
  Exp_Clien = 0
  Promo = 0
  Tiempo = 0
  Agrada = 0
  Serv_Clien = 0
  Bene = 0
  Seg = 0

  #Recorrer las palabras en el texto y actualización de puntuaciones
  for word in texto.split():
    if word in Servicios_Financieros:
      Serv_Fin += 1
    elif word in Aplicaciones_Plataformas:
      App += 1
    elif word in Tecnologia:
      Tecno += 1
    elif word in Bancos:
      Banco += 1
    elif word in Experiencia_Cliente:
      Exp_Clien += 1
    elif word in Promocion:
      Promo += 1
    elif word in Tiempo_Eventos:
      Tiempo += 1
    elif word in Agradecimiento:
      Agrada += 1
    elif word in Servicio_Cliente:
      Serv_Clien += 1
    elif word in Beneficios:
      Bene += 1
    elif word in Seguridad_Confianza:
      Seg += 1

  #Determinar la puntuación Máxima
  max_Cat = max(Serv_Fin,App, Tecno , Banco, Exp_Clien, Promo, Tiempo, Agrada, Serv_Clien, Bene, Seg)


  # Asignar los valores de categoría y porcentaje
  if max_Cat == Serv_Fin:
    Cat = "Servicios Financieros"
  elif max_Cat == App:
    Cat = "Servicios de App"
  elif max_Cat == Tecno:
    Cat = "Tecnologia"
  elif max_Cat == Banco:
    Cat = "Comparacion Bancos"
  elif max_Cat == Exp_Clien:
    Cat = "Experiencia del Cliente"
  elif max_Cat == Promo:
    Cat = "Promociones"
  elif max_Cat == Tiempo:
    Cat = "Eventos Limitados"
  elif max_Cat == Agrada:
    Cat = "Agradecimientos"
  elif max_Cat == Serv_Clien:
    Cat = "Servicio al Cliente"
  elif max_Cat == Bene:
    Cat = "Beneficios"
  elif max_Cat == Seg:
    Cat = "Seguridad y Educacion Financiera"

  return Cat

Hey['Razon/Categoria'] = Hey['texto_tokenizado2'].apply(lambda x: classify_text(x))
```

```
# Función de Clasificación de Texto
def classify_text2 (texto):
  Serv_Fin = 0
  App = 0
  Tecno = 0
  Banco = 0
  Exp_Clien = 0
  Promo = 0
  Tiempo = 0
  Agrada = 0
  Serv_Clien = 0
  Bene = 0
  Seg = 0

  #Recorrer las palabras en el texto y actualización de puntuaciones
  for word in texto.split():
    if word in Servicios_Financieros:
      Serv_Fin += 1
    elif word in Aplicaciones_Plataformas:
      App += 1
    elif word in Tecnologia:
      Tecno += 1
    elif word in Bancos:
      Banco += 1
    elif word in Experiencia_Cliente:
      Exp_Clien += 1
    elif word in Promocion:
      Promo += 1
    elif word in Tiempo_Eventos:
      Tiempo += 1
    elif word in Agradecimiento:
      Agrada += 1
    elif word in Servicio_Cliente:
      Serv_Clien += 1
    elif word in Beneficios:
      Bene += 1
    elif word in Seguridad_Confianza:
      Seg += 1

  #Determinar la puntuación Máxima
  max_Cat = max(Serv_Fin,App, Tecno , Banco, Exp_Clien, Promo, Tiempo, Agrada, Serv_Clien, Bene, Seg)


  # Asignar los valores de categoría y porcentaje
  if max_Cat == Serv_Fin:
    Porcentaje = (round(Serv_Fin / len(texto.split()),2))*100
  elif max_Cat == App:
    Porcentaje = (round(App / len(texto.split()),2))*100
  elif max_Cat == Tecno:
    Porcentaje = (round(Tecno / len(texto.split()),2))*100
  elif max_Cat == Banco:
    Porcentaje = (round(Banco / len(texto.split()),2))*100
  elif max_Cat == Exp_Clien:
    Porcentaje = (round(Exp_Clien / len(texto.split()),2))*100
  elif max_Cat == Promo:
    Porcentaje = (round(Promo / len(texto.split()),2))*100
  elif max_Cat == Tiempo:
    Porcentaje = (round(Tiempo / len(texto.split()),2))*100
  elif max_Cat == Agrada:
    Porcentaje = (round(Agrada / len(texto.split()),2))*100
  elif max_Cat == Serv_Clien:
    Porcentaje = (round(Serv_Clien / len(texto.split()),2))*100
  elif max_Cat == Bene:
    Porcentaje = (round(Bene / len(texto.split()),2))*100
  elif max_Cat == Seg:
    Porcentaje = (round(Seg / len(texto.split()),2))*100

  return Porcentaje

  return Porcentaje
Hey['Porcentaje'] = Hey['texto_tokenizado2'].apply(lambda x: classify_text2(x))


Hey.head()
```

Sentimiento y Escala de Satisfacción

Sentiment Analisis Espanol

```
# Instala los paquetes necesarios
!pip install sentiment-analysis-spanish
!pip install keras tensorflow

print(Hey)

from sentiment_analysis_spanish import sentiment_analysis
import keras
import tensorflow as tf

sentiment = sentiment_analysis.SentimentAnalysisSpanish()

# Crea una función para calcular el puntaje de sentimiento
def calcular_puntaje(tweet):
    return sentiment.sentiment(tweet)

# Supongamos que tienes una base de datos llamada 'db' con la columna 'tweet'
# Crear una nueva columna 'puntaje' que contiene los puntajes de sentimiento
Hey['puntaje'] = Hey['tweet'].apply(calcular_puntaje)

# Imprimir la base de datos con la nueva columna 'puntaje'
print(Hey)
```

VADER traducido al ingles y con emoticones

```
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from googletrans import Translator
from textblob import TextBlob

# Download NLTK resources if not already downloaded
nltk.download('vader_lexicon')

# Initialize the VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Initialize the translator
translator = Translator()

# Load the DataFrame
Hey = pd.read_csv("Datathon_HeyBanco.csv")

# Function to translate text to English
def translate_to_english(text):
    try:
        translation = translator.translate(text, dest='en')
        return translation.text
    except Exception as e:
        print("Translation error:", e)
        return None

# Apply translation to the "tweet" column and create a new column "tweet_eng"
Hey['tweet_eng'] = Hey['tweet'].apply(translate_to_english)

# Function to get sentiment score using VADER
def get_sentiment_score(text):
    try:
        score = sid.polarity_scores(text)['compound']
        return score
    except Exception as e:
        print("Sentiment analysis error:", e)
        return None

# Apply sentiment analysis to the "tweet_eng" column and create a new column "sentiment_score"
Hey['sentiment_score'] = Hey['tweet_eng'].apply(get_sentiment_score)

# Display the DataFrame with the new columns
print(Hey)
```

```
  File "<ipython-input-23-b2b3b3a97d7b>", line 4
    import "googletrans"
           ^
SyntaxError: invalid syntax
```

```
!pip install deep-translator
```

```
!pip install googletrans==4.0.0-rc1
```

```
!pip install google-cloud-translate
```

```
# ESTA CONVERSION EN INGLES ES SOLO PARA QUE LO CORRA EL VADER

from googletrans import Translator
import pandas as pd

# Function to translate text
def translate_text(text):
    translator = Translator()
    translation = translator.translate(text, src='es', dest='en')
    return translation.text

# Translate the 'tweet' column
Hey['tweet_english'] = Hey['tweet'].apply(translate_text)

# Display the DataFrame with translated column
print(Hey)



!pip install vaderSentiment


import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Initialize the VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Function to analyze sentiment including emojis and slang
def analyze_sentiment(text):
    # Sentiment analysis including emojis and slang
    sentiment_score = analyzer.polarity_scores(text)
    return sentiment_score

# Apply sentiment analysis to the 'tweet_english' column
Hey['sentiment_score'] = Hey['tweet_english'].apply(analyze_sentiment)

# Divide the 'sentiment_test' column into separate columns
Hey[['neg', 'neu', 'pos', 'compound']] = Hey['sentiment_score'].apply(lambda x: pd.Series(x))

# Drop the original 'sentiment_test' column
Hey.drop(columns=['sentiment_score'], inplace=True)

def analyze_sentiment(text):
    # Sentiment analysis including emojis and slang
    sentiment_score = analyzer.polarity_scores(text)
    # If compound score is greater than 0.1, classify as positive
    if sentiment_score['compound'] > 0.1:
        return 'Positive'
    # If compound score is less than -0.1, classify as negative
    elif sentiment_score['compound'] < -0.1:
        return 'Negative'
    # Otherwise, classify as neutral
    else:
        return 'Neutral'

# Apply sentiment analysis to the 'tweet_english' column
Hey['sentiment'] = Hey['tweet_english'].apply(analyze_sentiment)

# Display the DataFrame with separate sentiment columns
print(Hey)
```

```
import spacy
import pandas as pd

# Load the English language model for SpaCy
nlp = spacy.load("en_core_web_sm")

# Load the "categorias" table from the Excel file into a DataFrame
categorias = pd.read_excel("/content/categorias.xlsx")
print(categorias)

# Function to extract entities from a tweet using SpaCy
def extract_entities(tweet):
    # Process the tweet text with SpaCy
    doc = nlp(tweet)

    # Extract entities and their labels
    entities = [(entity.text, entity.label_) for entity in doc.ents]

    # Prioritize certain words as subjects
    priority_words = categorias['word'].tolist()
```