

Comienza a programar o [generar](#) con IA.

> prompt: montar el drive

[] ↪ 1 celda oculta

> Importa las librerías necesarias

[] ↪ 1 celda oculta

> Generacion de archivo a analizar

[] ↪ 2 celdas ocultas

✓ EDA - Analisis exploratorio

Haz doble clic (o ingresa) para editar

df_10000.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    texto      10000 non-null  object
1    sentimiento 10000 non-null  object
2    compañía   10000 non-null  object
dtypes: object(3)
memory usage: 234.5+ KB
```

Convertir los tipos Object

```
df_10000['texto'] = df_10000['texto'].astype('string')
df_10000['sentimiento'] = df_10000['sentimiento'].astype('category')
df_10000['compañía'] = df_10000['compañía'].astype('category')
```

Visualizar información general
print(df_10000.info())

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    texto      10000 non-null  string
1    sentimiento 10000 non-null  category
2    compañía   10000 non-null  category
dtypes: category(2), string(1)
memory usage: 98.0 KB
None
```

Consulta si hay valores faltantes, True si hay valores faltantes. En este caso
print(df_10000.isna().any())

```
texto      False
sentimiento False
compañía    False
dtype: bool
```

from sklearn.model_selection import train_test_split

```
# Separar en entrenamiento (80%) y test (20%)
data_train, data_dev = train_test_split(df_10000,
                                         test_size=0.2,          #20% de los datos se usan para test.
                                         random_state=42,         # Semilla para reproducibilidad, hace que siempre salga
```

ModuleNotFoundError X



Explica el error:

ModuleNotFoundError: No module named 'evaluate'



Hypothesis

The error message ModuleNotFoundError: No module named 'evaluate' indicates that the Python interpreter cannot find the evaluate module. This is likely due to the fact that the library is not installed in the user's current environment. The user likely needs to install this library to continue their work.

Suggested Changes

Install evaluate with pip:

```
!pip install evaluate
```

[Usa el código con cuidado](#)

► Fuentes

Califica esta respuesta



```
stratify=df_10000['sentimiento'] # garantiza que las clases positivas, negat
)

# Mostrar tamaños
print(f"Entrenamiento: {len(data_train)} filas")
print(f"Test: {len(data_dev)} filas")
```

↗ Entrenamiento: 8000 filas
Test: 2000 filas

```
data_train.info()
```

↗

```
<class 'pandas.core.frame.DataFrame'>
Index: 8000 entries, 9415 to 3655
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   texto       8000 non-null   string
1   sentimiento  8000 non-null   category
2   compañía    8000 non-null   category
dtypes: category(2), string(1)
memory usage: 140.9 KB
```

```
data_dev.info()
```

↗

```
<class 'pandas.core.frame.DataFrame'>
Index: 2000 entries, 8846 to 3764
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   texto       2000 non-null   string
1   sentimiento  2000 non-null   category
2   compañía    2000 non-null   category
dtypes: category(2), string(1)
memory usage: 35.4 KB
```

> !pip install datasets

[] ↳ 1 celda oculta

✓ Mapeo de etiquetas y creación del dataset Hugging Face

```
data_train.head()
```

↗

	texto	sentimiento	compañía
9415	Gracias @PersonalPy, resolvieron mi problema m...	positivo	@PersonalPy
8307	No me molesta usar @TigoParaguay, pero tampoco...	neutro	@TigoParaguay
7941	Con @TigoParaguay no he tenido novedades impor...	neutro	@TigoParaguay

Próximos pasos:

[Generar código con data_train](#)

[Ver gráficos recomendados](#)


[New interacti](#)


```
# 1. Resetear índice
data_train = data_train.reset_index(drop=True)
data_dev = data_dev.reset_index(drop=True)

# 2. Reemplazar sentimientos
data_train['sentimiento'] = data_train['sentimiento'].replace({
    'positivo': 'POS',
    'neutro': 'NEU',
    'negativo': 'NEG'
})
data_dev['sentimiento'] = data_dev['sentimiento'].replace({
    'positivo': 'POS',
    'neutro': 'NEU',
    'negativo': 'NEG'
})
```

```
# 2. Reemplazar compañía - using the correct column name with the accent mark
data_train['compañía'] = data_train['compañía'].replace({ # changed 'compañia' to
 '@PersonalPy': 'Personal',
 '@TigoParaguay': 'Tigo',
 '@ClaroPY': 'Claro'
})
data_dev['compañía'] = data_dev['compañía'].replace({ # changed 'compañia' to 'co
 '@PersonalPy': 'Personal',
 '@TigoParaguay': 'Tigo',
 '@ClaroPY': 'Claro'
})

# (opcional) 3. Visualizar
data_train.head()
```


 <ipython-input-188-b9ee2efa1df3>:6: FutureWarning: The behavior of Series.rep
data_train['sentimiento'] = data_train['sentimiento'].replace({
<ipython-input-188-b9ee2efa1df3>:11: FutureWarning: The behavior of Series.re
data_dev['sentimiento'] = data_dev['sentimiento'].replace({
<ipython-input-188-b9ee2efa1df3>:18: FutureWarning: The behavior of Series.re
data_train['compañía'] = data_train['compañía'].replace({ # changed 'compañ
<ipython-input-188-b9ee2efa1df3>:23: FutureWarning: The behavior of Series.re
data_dev['compañía'] = data_dev['compañía'].replace({ # changed 'compañía'


	texto	sentimiento	compañía	
0	Gracias @PersonalPy, resolvieron mi problema m...	POS	Personal	
1	No me molesta usar @TigoParaguay, pero tampoco...	NEU	Tigo	
2	Con @TigoParaguay no he tenido novedades impor...	NEU	Tigo	
-	-	Claro	
4	@TigoParaguay tiene la mejor relación calidad-...	POS	Tigo	

Próximos
pasos:

[Generar código con data_train](#)
[Ver gráficos recomendados](#)
[New interacti](#)

```
# (opcional) 3. Visualizar
data_dev.head()
```



	texto	sentimiento	compañía	
0	@TigoParaguay tiene el mejor internet que he p...	POS	Tigo	
1	Con @PersonalPy tengo lo que espero, nada más	NEU	Personal	
2	@PersonalPy cumple lo justo con lo que promete	NEU	Personal	
3	Nunca tienen solución para mis problemas en @P...	NEG	Personal	
4	Demasiado caro para un servicio tan malo el de...	NEG	Tigo	

Próximos
pasos:

[Generar código con data_dev](#)
[Ver gráficos recomendados](#)
[New interactive](#)

```
# Diccionario de mapeo de etiquetas
mapeo_etiquetas_sent = {'NEG': 0, 'NEU': 1, 'POS': 2}
mapeo_etiquetas_comp = {'Tigo': 0, 'Personal': 1, 'Claro': 2}

from datasets import load_dataset, Dataset, Features, ClassLabel, Sequence, Value
# Define features with the correct column names from your DataFrame
features = Features({
    'texto': Value('string'), # Assuming 'texto' is the text column in your Data
    'sentimiento': ClassLabel(num_classes=3, names=['NEG', 'NEU', 'POS']), # Ass
    'compañía': ClassLabel(num_classes=3, names=['Tigo', 'Personal', 'Claro']) #
})

data_train['sentimiento'] = data_train['sentimiento'].map(mapeo_etiquetas_sent)
data_dev['sentimiento'] = data_dev['sentimiento'].map(mapeo_etiquetas_sent) # map

data_train['compañía'] = data_train['compañía'].map(mapeo_etiquetas_comp)
data_dev['compañía'] = data_dev['compañía'].map(mapeo_etiquetas_comp) # map for d

# (opcional) 3. Visualizar
data_train.head()
```



	texto	sentimiento	compañía
0	Gracias @PersonalPy, resolvieron mi problema m...	2	1
1	No me molesta usar @TigoParaguay, pero tampoco...	1	0
2	Con @TigoParaguay no he tenido novedades impor...	1	0
3	No es la mejor ni la peor, @ClaroPY está bien	1	2
4	@TigoParaguay tiene la mejor relación calidad-...	2	0



Próximos
pasos:

[Generar código con data_train](#)[Ver gráficos recomendados](#)[New interacti](#)

```
# (opcional) 3. Visualizar
data_dev.head()
```



	texto	sentimiento	compañía
0	@TigoParaguay tiene el mejor internet que he p...	2	0
1	Con @PersonalPy tengo lo que espero, nada más	1	1
2	@PersonalPy cumple lo justo con lo que promete	1	1
3	Nunca tienen solución para mis problemas en @P...	0	1
4	Demasiado caro para un servicio tan malo el de...	0	0



Próximos
pasos:

[Generar código con data_dev](#)[Ver gráficos recomendados](#)[New interactive](#)

```
#transformamos de dataframe a dataset
dataset_train = Dataset.from_pandas(data_train, features=features)
dataset_dev = Dataset.from_pandas(data_dev, features=features)
```

```
dataset_train = dataset_train.rename_column("texto", "text")
dataset_train = dataset_train.rename_column("sentimiento", "label")
dataset_dev = dataset_dev.rename_column("texto", "text")
dataset_dev = dataset_dev.rename_column("sentimiento", "label")
```

```
#creamos diccionario de dataset
dataset = DatasetDict({'train': dataset_train, 'test': dataset_dev})
dataset
```



```
DatasetDict({
  train: Dataset({
    features: ['text', 'label', 'compañía'],
    num_rows: 8000
  })
  test: Dataset({
    features: ['text', 'label', 'compañía'],
    num_rows: 2000
  })
})
```

```
# Mostrar cada ejemplo individualmente
for example in dataset['train'].select(range(10)):
    print(example)
```



```
{'text': 'Gracias @PersonalPy, resolvieron mi problema muy rápido', 'label': 2, 'compañía': 1}
{'text': 'No me molesta usar @TigoParaguay, pero tampoco me entusiasma', 'label': 1, 'compañía': 0}
{'text': 'Con @TigoParaguay no he tenido novedades importantes', 'label': 1, 'compañía': 0}
{'text': 'No es la mejor ni la peor, @ClaroPY está bien', 'label': 1, 'compañía': 2}
{'text': '@TigoParaguay tiene la mejor relación calidad-precio', 'label': 2, 'compañía': 0}
{'text': '@ClaroPY me cobra de más cada mes', 'label': 0, 'compañía': 2}
{'text': 'Estoy considerando cambiar de proveedor, @TigoParaguay no cumple', 'label': 0, 'compañía': 0}
{'text': 'Servicio estándar de @TigoParaguay, cumple con lo básico', 'label': 1, 'compañía': 1}
{'text': '@TigoParaguay tiene el mejor internet que he probado', 'label': 2, 'compañía': 0}
{'text': 'Todo perfecto con @ClaroPY, buena señal y atención', 'label': 2, 'compañía': 2}
```



```
# Mostrar cada ejemplo individualmente
for example in dataset['test'].select(range(10)):
    print(example)
```

```
import random
import pandas as pd
from datasets import ClassLabel
from IPython.display import display, HTML

def show_random_elements(dataset, num_examples=10):
    "Taken from https://github.com/huggingface/notebooks/blob/master/examples/text_classification.ipynb"
    assert num_examples <= len(dataset), "Can't pick more elements than there are"
    picks = []
    for _ in range(num_examples):
        pick = random.randint(0, len(dataset)-1)
        while pick in picks:
            pick = random.randint(0, len(dataset)-1)
        picks.append(pick)

    df = pd.DataFrame(dataset[picks])
    for column, typ in dataset.features.items():
        if isinstance(typ, ClassLabel):
            df[column] = df[column].transform(lambda i: typ.names[i])
    display(HTML(df.to_html()))

show_random_elements(dataset['train'])
```

		text	label	compañía
0		@ClaroPY cumple lo justo con lo que promete	NEU	Claro
1		Llevo más de una semana esperando que @ClaroPY me llame	NEG	Claro
2		Las actualizaciones de la app de @PersonalPy mejoran mucho la experiencia	POS	Personal
3		Todo depende del lugar, con @PersonalPy es muy variable	NEU	Personal
4		@TigoParaguay podría mejorar, pero no me ha ido mal	NEU	Tigo
5		Decepcionado completamente con @ClaroPY	NEG	Claro
6		Con @TigoParaguay tengo buena señal hasta en viajes largos	POS	Tigo
7		El precio de @TigoParaguay está dentro del promedio	NEU	Tigo
8		Todo el barrio tiene problemas con @PersonalPy	NEG	Personal
9		Me cambié a @ClaroPY y fue la mejor decisión	POS	Claro

	text	label	compañía
0	Gracias @PersonalPy, resolvieron mi problema m...	2	1
1	No me molesta usar @TigoParaguay, pero tampoco...	1	0
2	Con @TigoParaguay no he tenido novedades impor...	1	0
3	No es la mejor ni la peor, @ClaroPY está bien	1	2
4	@TigoParaguay tiene la mejor relación calidad-...	2	0

```
show random elements(dataset['train'], num examples=3)
```



	text	label	compañía
0	Cada mes tengo problemas con la facturación de @PersonalPy	NEG	Personal
1	Desde que contraté @TigoParaguay todo es un desastre	NEG	Tigo
2	Excelente servicio de @TigoParaguay, muy satisfecho	POS	Tigo

✓ Tokenizar las reseñas

Modelo	Descripción
<code>distilbert/distilbert-base-multilingual-cased</code>	Ligero, multilingüe, más rápido que BERT, menor tamaño.
<code>bert-base-multilingual-cased</code>	Modelo BERT multilingüe original entrenado en más de 100 idiomas.
<code>xlm-roberta-base</code>	Modelo robusto y multilingüe (mejor rendimiento que BERTm).
<code>papluca/xlm-roberta-base-language-detection</code>	XLNet-RoBERTa para detección de idioma.
<code>facebook/mbart-large-50-many-to-many-mmt</code>	Más orientado a traducción, pero útil si tu tarea es multilingüe.

✓ bertin-project/bertin-roberta-base-spanish

```
from transformers import AutoTokenizer
```

```
model_checkpoint = "bertin-project/bertin-roberta-base-spanish"
#model_checkpoint = "distilbert/distilbert-base-multilingual-cased"
```

```
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
```

```
tokenizer.vocab_size
```



```
50262
```

```
text = ";hola, estamos muy felices practicando la tokenizacion!"
tokenized_text = tokenizer.encode(text)
```

```
for token in tokenized_text:
    print(token, tokenizer.decode([token]))
```



```
0 <s>
1465 ;
12616 hola
66 ,
2608 estamos
727 muy
13012 felices
30255 practicando
332 la
449 to
19302 ken
23869 izacion
55 !
2 </s>
```

```
encoded_text = tokenizer(text, return_tensors="pt")
encoded_text
```



```
{'input_ids': tensor([[ 0, 1465, 12616, 66, 2608, 727, 13012,
30255, 332, 449,
19302, 23869, 55, 2]]), 'attention_mask': tensor([[1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1]])}
```

```
def tokenize_reviews(examples):
    return tokenizer(examples["text"], truncation=True)
```

```
columns = dataset['train'].column_names
columns.remove("label")
encoded_dataset = dataset.map(tokenize_reviews, batched=True, remove_columns=columns)
encoded_dataset
```

```

Map: 100% 8000/8000 [00:00<00:00, 11087.72 examples/s]

Map: 100% 2000/2000 [00:00<00:00, 9162.91 examples/s]

DatasetDict({
  train: Dataset({
    features: ['label', 'input_ids', 'attention_mask'],
    num_rows: 8000
  })
  test: Dataset({
    features: ['label', 'input_ids', 'attention_mask'],
    num_rows: 2000
  })
})

```

```
encoded_dataset['train'][0]
```

```

{'label': 2,
 'input_ids': [0,
  3822,
  1139,
  43605,
  102,
  143,
  66,
  502,
  377,
  3920,
  737,
  2017,
  727,
  5744,
  2],
 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}

```

✓ Cargar el modelo preentrenado

```
from transformers import AutoModelForSequenceClassification
```

```
num_labels = 3
```

```
model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_
```

```

config.json: 100% 674/674 [00:00<00:00, 38.1kB/s]

model.safetensors: 100% 499M/499M [00:03<00:00, 160MB/s]

Some weights of RobertaForSequenceClassification were not initialized from th
You should probably TRAIN this model on a down-stream task to be able to use

```

✓ De las input IDs a los hidden states

```
outputs = model(**encoded_text)
outputs
```

```

SequenceClassifierOutput(loss=None, logits=tensor([[ 0.0185,  0.1181,
 -0.1084]]), grad_fn=<AddmmBackward0>), hidden_states=None, attentions=None)

```

✓ Definir las métricas de rendimiento

```
!pip install evaluate
```

```

Requirement already satisfied: evaluate in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: dill in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.11/
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: multiprocessing in /usr/local/lib/python3.11/dist
Requirement already satisfied: fsspec>=2021.05.0 in /usr/local/lib/python3.11
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/pytho
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-pack

```

```
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/p
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/di
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/pyth
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-pac
```

```
import evaluate
```

```
metric = evaluate.load("accuracy")
metric
```

```
↗ EvaluationModule(name: "accuracy", module_type: "metric", features:
{'predictions': Value(dtype='int32', id=None), 'references':
Value(dtype='int32', id=None)}, usage: "")
Args:
  predictions (`list` of `int`): Predicted labels.
  references (`list` of `int`): Ground truth labels.
  normalize (`boolean`): If set to False, returns the number of correctly
classified samples. Otherwise, returns the fraction of correctly classified
samples. Defaults to True.
  sample_weight (`list` of `float`): Sample weights Defaults to None.
```

```
Returns:
  accuracy (`float` or `int`): Accuracy score. Minimum possible value is
0. Maximum possible value is 1.0, or the number of examples input, if
`normalize` is set to `True`. A higher score means higher accuracy.
```

```
Examples:
```

```
Example 1-A simple example
```

```
>>> accuracy_metric = evaluate.load("accuracy")
>>> results = accuracy_metric.compute(references=[0, 1, 2, 0, 1, 2],
predictions=[0, 1, 1, 2, 1, 0])
>>> print(results)
{'accuracy': 0.5}
```

```
Example 2-The same as Example 1, except with `normalize` set to `False`.
```

```
>>> accuracy_metric = evaluate.load("accuracy")
>>> results = accuracy_metric.compute(references=[0, 1, 2, 0, 1, 2],
predictions=[0, 1, 1, 2, 1, 0], normalize=False)
>>> print(results)
{'accuracy': 3.0}
```

```
Example 3-The same as Example 1, except with `sample_weight` set.
```

```
>>> accuracy_metric = evaluate.load("accuracy")
>>> results = accuracy_metric.compute(references=[0, 1, 2, 0, 1, 2],
predictions=[0, 1, 1, 2, 1, 0], sample_weight=[0.5, 2, 0.7, 0.5, 9, 0.4])
>>> print(results)
{'accuracy': 0.8778625954198473}
""", stored examples: 0)
```

```
import numpy as np
```

```
def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    predictions = np.argmax(predictions, axis=1)
    return metric.compute(predictions=predictions, references=labels)
```


Gemini puede cometer errores, así que verifica las respuestas y usa el código con precaución. [Más información](#)