



PROYECTO 1

IAAP

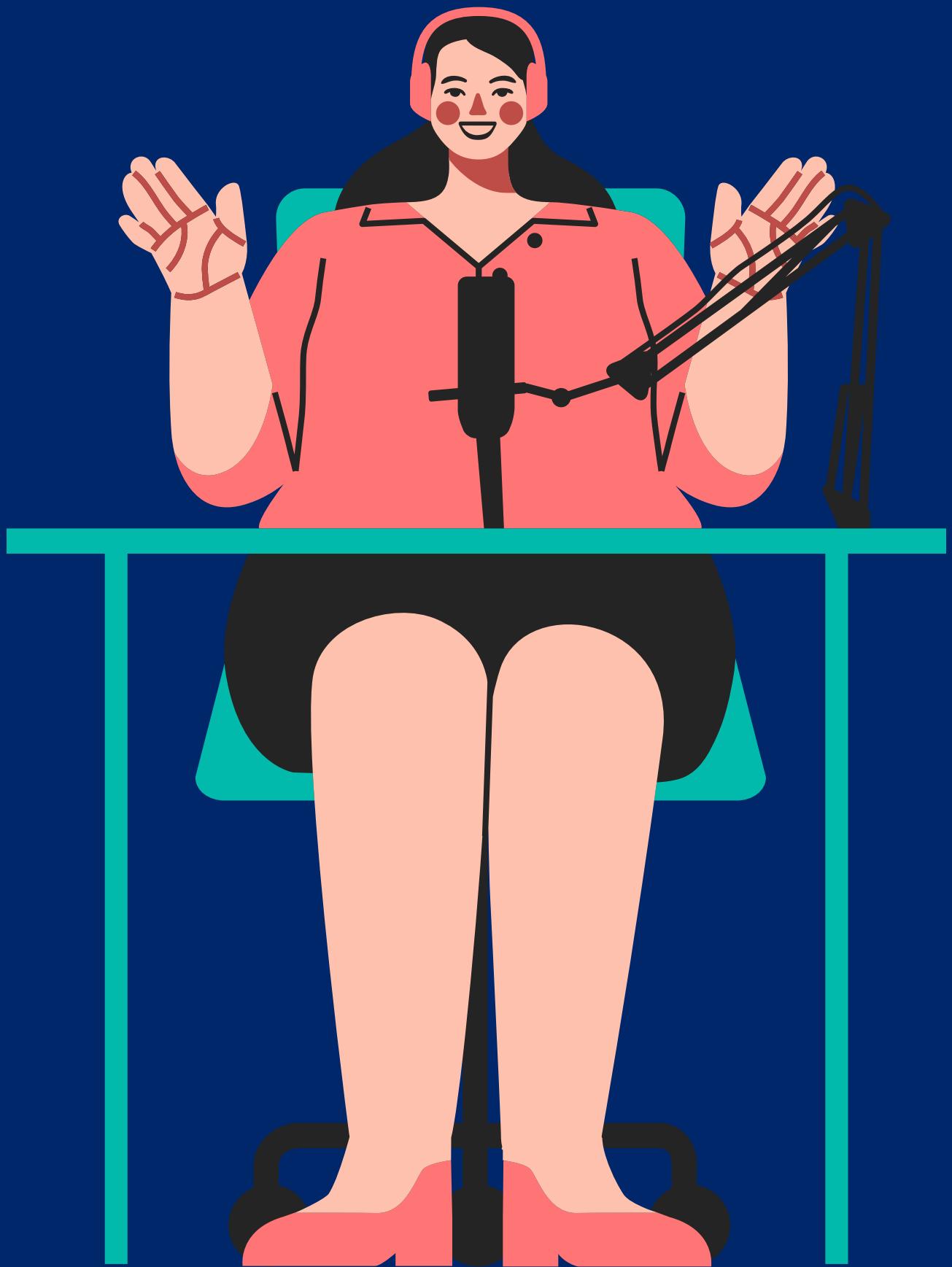


PRESENTED BY: JUAN PABLO DAZA PEREIRA
JUAN SEBASTIAN CAMARGO SANCHEZ

NETWORK NEWS



¿CUÁL ES EL PROBLEMA?



- 01** Identificación de noticias generadas por humanos frente a inteligencia artificial

- 02** Identificar las “fake news” frente a la automatización de información sesgada generada por IA

¿POR QUÉ ES IMPORTANTE RESOLVERLO?



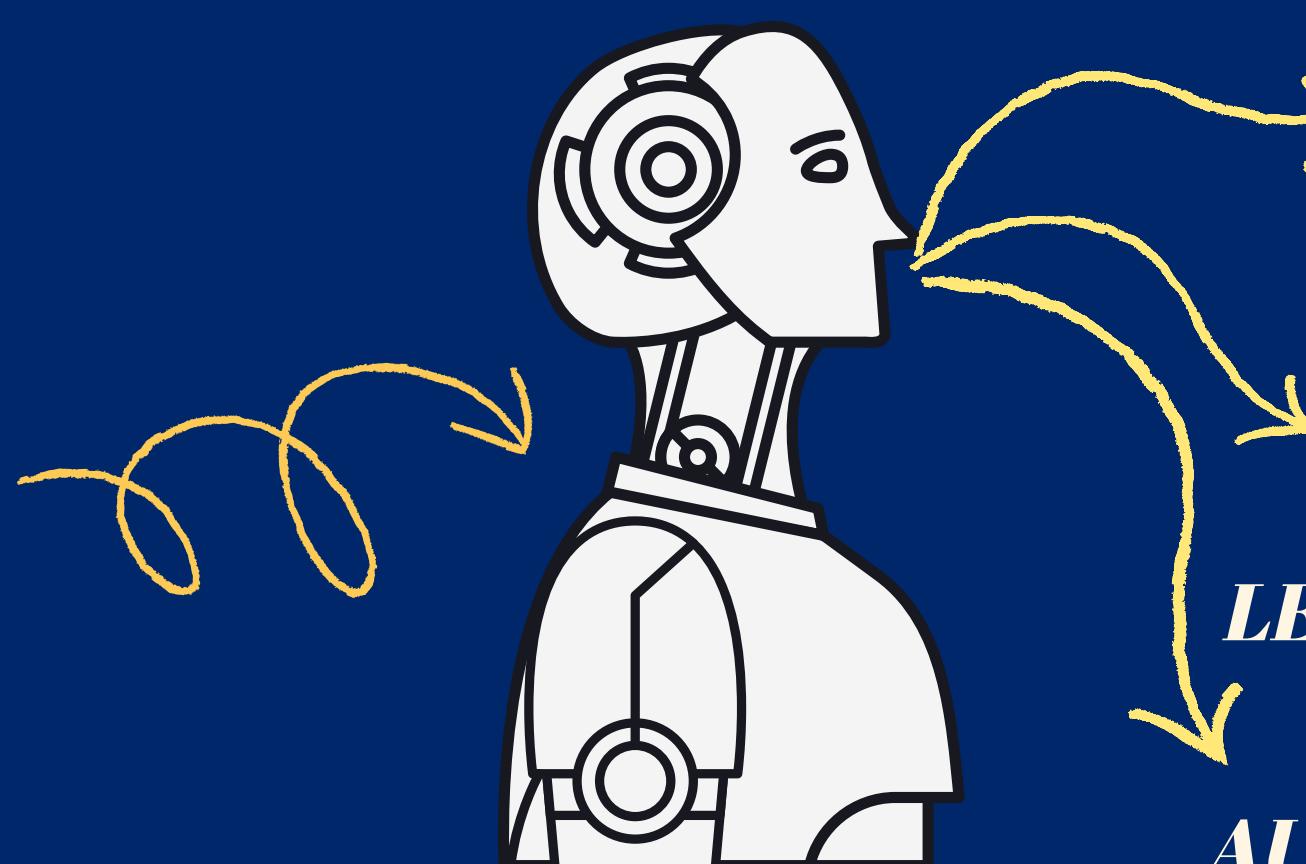
. Comprobar el origen de un texto informativo como una noticia por varias razones fundamentales. En primer lugar, la verificación del origen ayuda a garantizar la autenticidad y la credibilidad de la información presentada. En un contexto donde la desinformación y las noticias falsas proliferan en línea, es esencial poder distinguir entre contenido generado por humanos y aquel generado por inteligencia artificial (IA)

¿QUÉ SE HA HECHO AL RESPECTO?

LIMITACIONES

*FAKE
NEWS*

*BOTS
SOCIALES*



*VIRALIZACIÓN
DE NOTICIAS
FALSAS*

*ADAPTACIÓN
AL
LENGUAJE NATURAL*

AUTOMATIZACIÓN

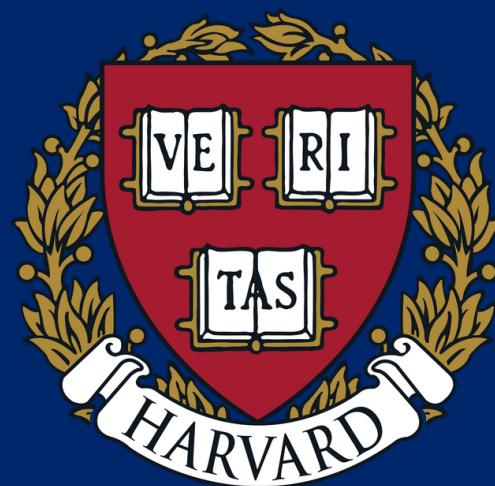
*RETO PARA LAS
COMBINACIONES
DE VALORES
DIFERENCIALES*

*TEXTOS FICTICIOS
MÁS
VERACES*

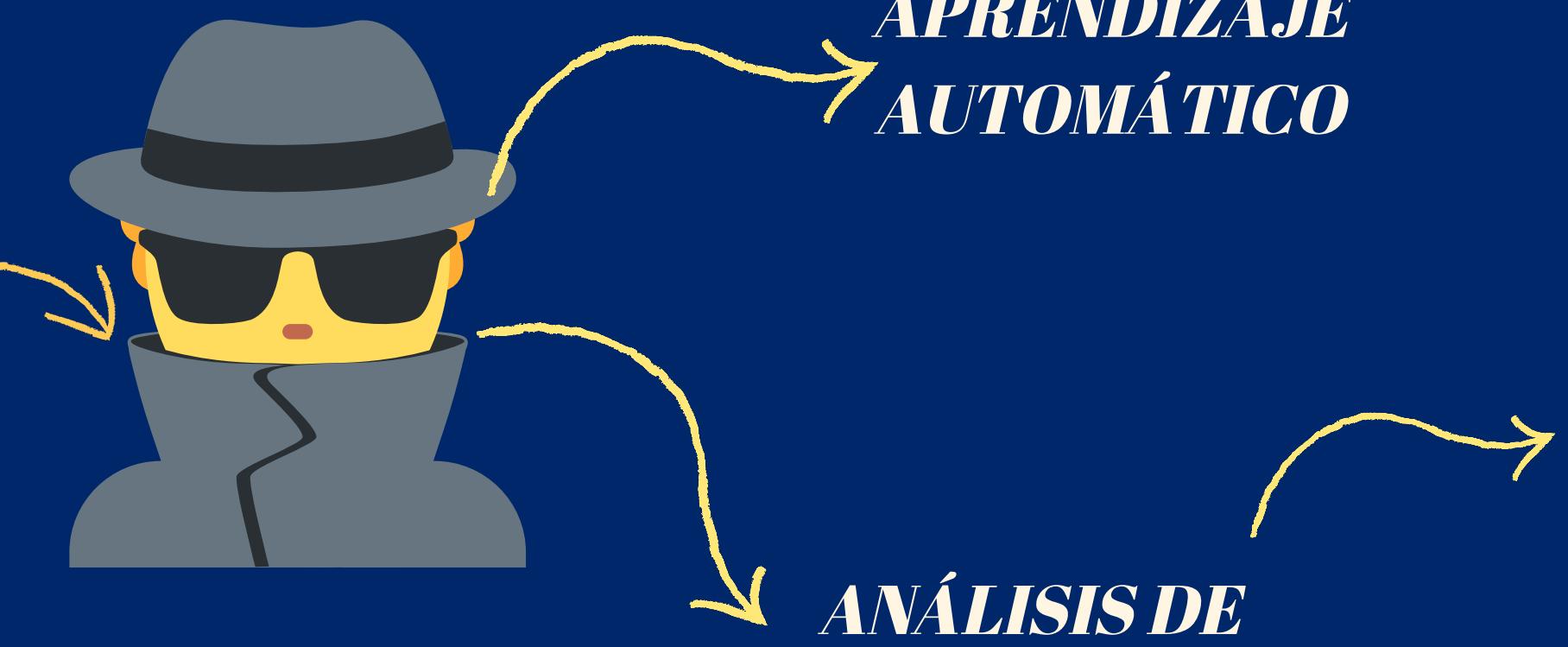
¿QUÉ SE HA HECHO AL RESPECTO?

LIMITACIONES

*DETECCIÓN
AUTOMÁTICA DE
BOTS*



*Misinformation
Susceptibility Test*



Google Fact Check Tools



¿QUÉ SE HA HECHO AL RESPECTO? LIMITACIONES



*ESTUDIO DE LAS REDES QUE FORMAN
LAS PLATAFORMAS DE BOTS.*

¿CUÁL FUE LA IDEA DE SOLUCIÓN?



¿CUÁL FUE LA IDEA DE SOLUCIÓN?

kaggle



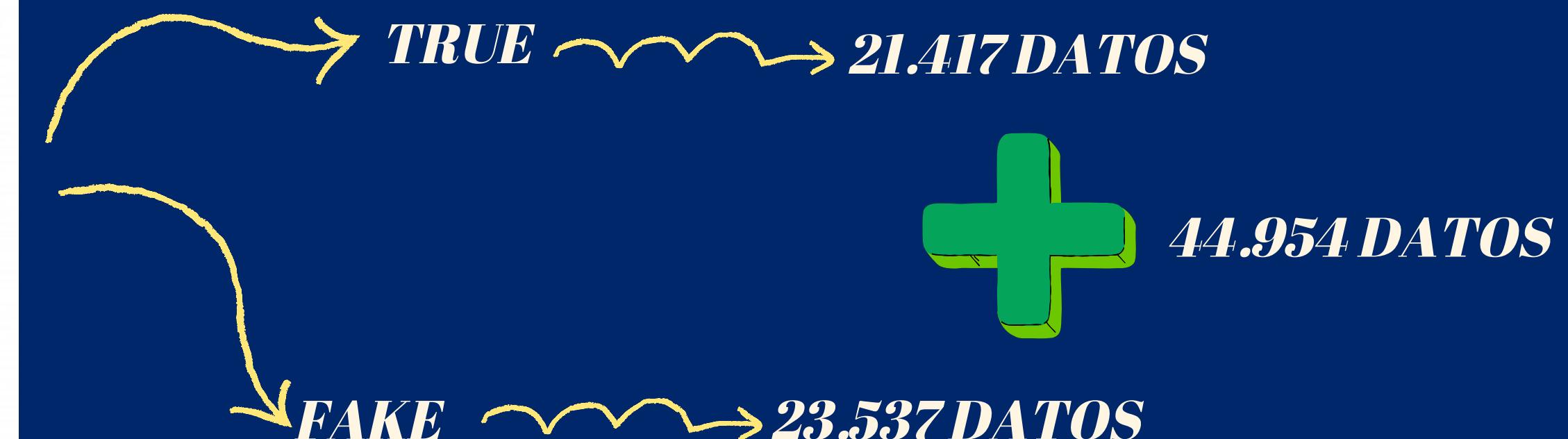
FAKE NEWS
DETECTION

¿CUÁL FUE LA IDEA DE SOLUCIÓN?

13/01/2016 - 31/12/2017



FAKE NEWS
DETECTION



31/03/2015 - 19/02/2018

¿CUÁL FUE LA IDEA DE SOLUCIÓN?



- Procesar la Data
- Desarrollar una red neuronal usando técnicas de optimización y regularización
- Poner a prueba el resultado de nuestra red con noticias generadas por IA.
- Si es el caso llevar un nuevo proceso de entrenamiento generando mas Data.



Como medida de precisión tenemos un 93.8% que fue uno de los resultados de una de las redes neuronales generadas sobre la DB. Este resultado se logró con un clasificador ingenuo de Bayes para modelos multinomiales.

Pensamos llevar a cabo nuestra red con funciones de activación ReLU y con una función de salida Sigmoide de clasificación binaria.

¿CUÁL FUE LA SOLUCIÓN?



TRATAMIENTO DE DATOS

```
0.25      85.0
0.50      211.0
0.75      306.0
0.90      454.0
0.95      527.0
Name: text, dtype: float64
0.25      136.0
0.50      195.0
0.75      265.0
0.90      357.0
0.95      441.0
Name: text, dtype: float64
```

Se hace una limpieza a nuestra data queda es procesar el texto para que pueda ser entendida por una red neuronal. Para eso usamos una función de tokens, que va a asignarle un identificador del vocabulario en inglés, a cada palabra para después convertir todo el texto en una matriz de TF-IDF

TRATAMIENTO DE DATOS

```
4 #Create a new column with their fake values
5 fake['fake'] = 1.0
6 true['fake'] = 0.0
7
8 #Drop null values
9 fake = fake.dropna()
10 true = true.dropna()
11
12 #Drop duplicates values
13 true = true.drop_duplicates()
14 fake = fake.drop_duplicates()
15
16 #Balance the data
17 fake = fake[:20000]
18 true = true[:20000]
19
20 #Unify the fake data with true data
21 dTF = pd.concat([fake,true],ignore_index=True)
22
23 #Drop null values again
24 dTF = dTF.dropna()
25
26 dTF.shape
27
28 #Drop useless columns
29 dTF = dTF.drop(['date'], axis=1)
30 dTF = dTF.drop(['subject'], axis=1)
31
32 dTF.info()

40000, 5)
class 'pandas.core.frame.DataFrame'
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
 --- 
 0   title    40000 non-null  object 
 1   text     40000 non-null  object 
 2   fake     40000 non-null  float64
```

```
from nltk.stem import SnowballStemmer

# SnowballStemmer
stemmer = SnowballStemmer('english')

def stem_text(text):
    # Tokenize the input text into individual words
    tokens = nltk.word_tokenize(text)

    # Stem each token using the SnowballStemmer
    stemmed_tokens = [stemmer.stem(token) for token in tokens]

    # Join the stemmed tokens back into a single string
    return ' '.join(stemmed_tokens)

# Apply stemming to the text data
dTF['text']=dTF['text'].apply(stem_text)
```

Una vez se procesa todos los textos lo vectorizamos para la entrada a la red La
limpieza

TRATAMIENTO DE DATOS

```
#Prepare the data Stage: Training, Developing and Testing
y=dTF['fake']

X_train, X_devtest, y_train, y_devtest = train_test_split(X, y, test_size=0.2, random_state=42)

X_trainStage, X_trainTestStage, y_trainStage, y_trainTestStage = train_test_split(X_train, y_train, test_size=0.2, random_state=42)

X_dev,X_test,y_dev,y_test = train_test_split(X_devtest, y_devtest, test_size=0.5, random_state=42)

X_devStage,X_devTestStage,y_devStage,y_devTestStage = train_test_split(X_dev, y_dev, test_size=0.2, random_state=42)

X_testStage,X_testTestStage,y_testStage,y_testTestStage = train_test_split(X_test, y_test, test_size=0.2, random_state=42)
```

Ya teniendo esto automatizamos el proceso de dividir nuestra data en 80/10/10 para Training, Developin y Testing, respectivamente. La figura muestra la ejecución y cantidad de datos para cada entrada.

TRATAMIENTO DE DATOS

Data for every Stage:

Data for training (32000, 158267)

Data for dev (4000, 158267)

Data for testing (4000, 158267)

Data for used in the stage :

Data (32000, 158267) for training stage (25600, 158267) (6400, 158267)

Data (4000, 158267) for dev stage (3200, 158267) (800,)

Data (4000, 158267) for dev stage (3200, 158267) (800,)

El código muestra como dividimos la data por etapas y la consola nos muestra el tamaño por cada etapa junto a su entrenamiento 80/20 en cada etapa.

CREACIÓN DEL MODELO

```
{'activation': 'relu',
'alpha': 0.0,
'batch_size': 1,
'beta_1': 0.9,
'beta_2': 0.999,
'early_stopping': False,
'epsilon': 1e-08,
'hidden_layer_sizes': (100,),
'learning_rate': 'constant',
'learning_rate_init': 0.001,
'max_fun': 15000,
'max_iter': 100,
'momentum': 0.0,
'n_iter_no_change': 10,
'nesterovs_momentum': False,
'power_t': 0.0,
'random_state': None,
'shuffle': True,
'solver': 'sgd',
'tol': 0.0001,
'verification_fraction': 0.0,
'verbose': False,
'warm_start': False})
```

Para la creación del modelo de nuestra red decidimos utilizar sklearn como librería, por la manera intuitiva de usarla y su documentación completa. Usaremos su modelo de MLPClassifier , que es Multy Layer Perceptron Classifier. Un modelo de clasificación, con una red a capas con un perceptrón (Aunque esto se puede modificar a más neuronas) y para clasificación.

¿QUÉ RESULTADOS HAN OBTENIDO?



TRAINING

```
MLPClassifier  
MLPClassifier(alpha=0.0, batch_size=1, max_iter=100, momentum=0.0,  
    nesterovs_momentum=False, power_t=0.0, solver='sgd',  
    validation_fraction=0.0)  
0.59
```

El primer resultado de la experimentación con este modelo duro más de dos horas y media y además sin tener ningún resultado ya que el entorno de ejecución Colab llegó a su límite. Por lo que empezaremos por ejecutar el modelo y terminar su ejecución a la hora.

TRAINING

Por lo que no está acorde con nuestra métrica de 93,8 que elegimos, modificamos el modelo optimizándolo. Aplicamos mini batch, momentum, nesterovs momentum, cambiamos el valor de rango de aprendizaje y cada capa tendrá 3 2 perceptrones más.

```
MLPClassifier(alpha=0.0, hidden_layer_sizes=(100, 3), max_iter=100, power_t=0.0,
              solver='sgd', validation_fraction=0.0)
```

0.9540625

Ya con un 95.40 podremos avanzar a la etapa de desarrollo.

DEV

```
1 #Test dev  
2 MLP_C_modelTrain.score(X_dev, y_dev)  
0.9555
```

La precisión sigue siendo mayor a la métrica por lo que seguimos a la siguiente fase de Testing.

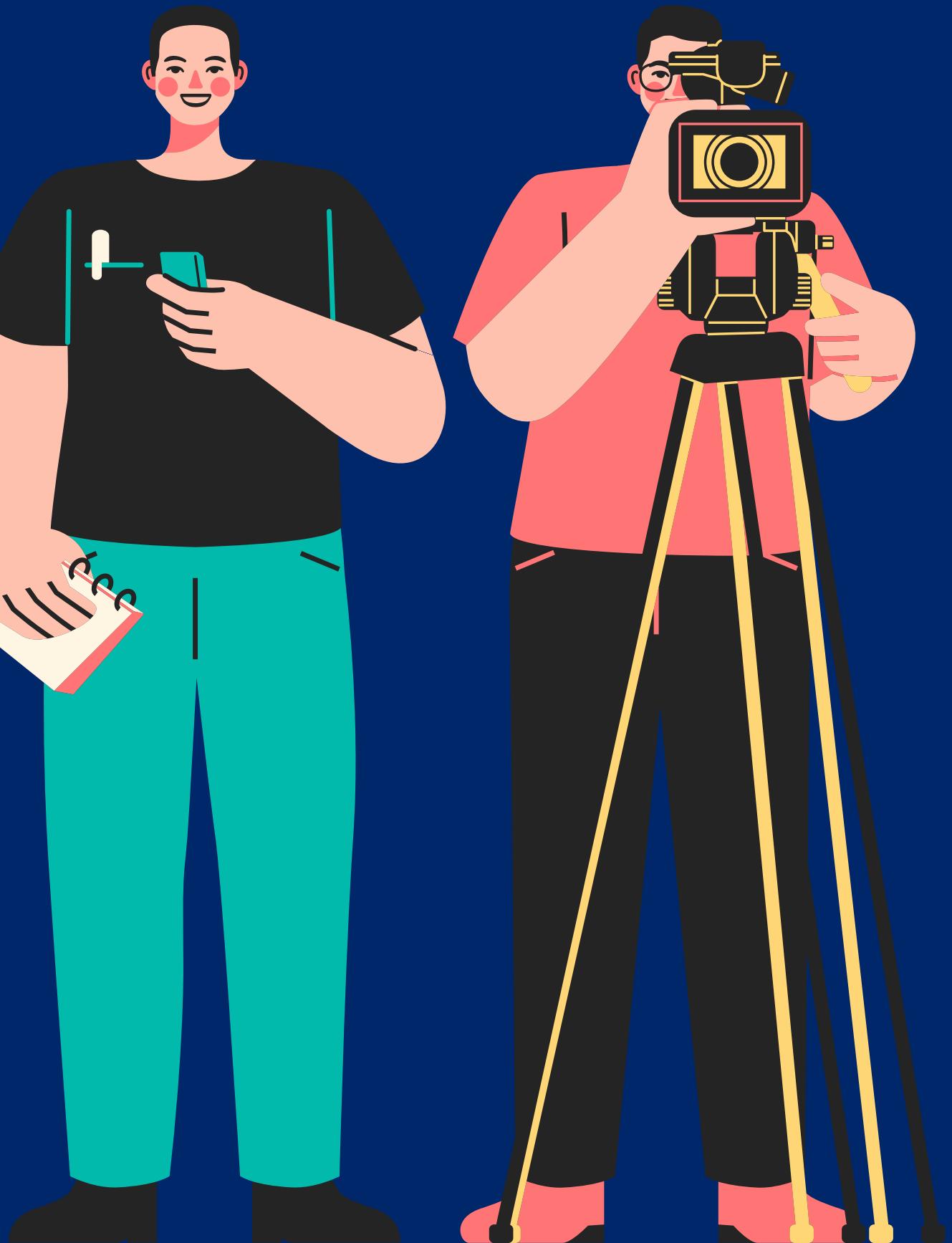
TESTING

```
1 #Test Training  
2 MLP_C_model.score(X_test, y_test)
```

```
0.959
```

La precisión sigue siendo mayor a la métrica cumpliendo con nuestro objetivo de proyecto.

¿QUÉ PODEMOS APRENDER DE LOS RESULTADOS?



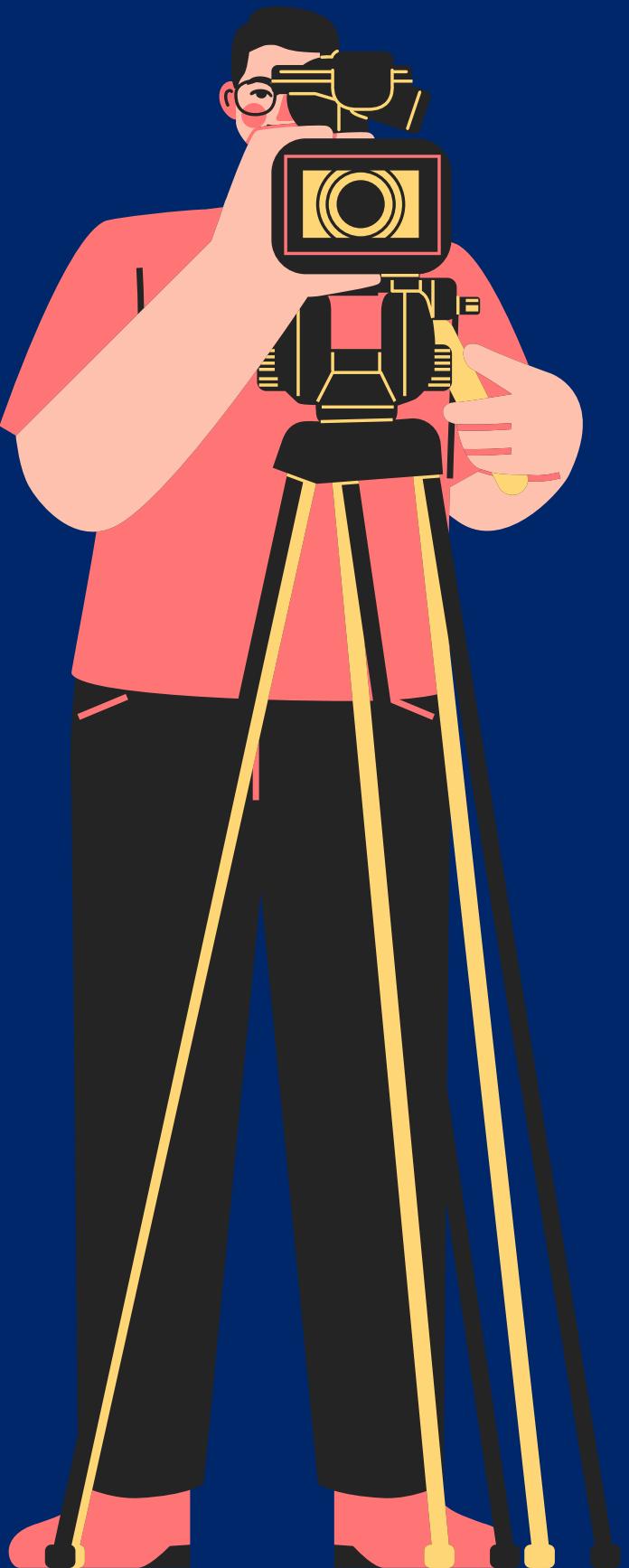
¿QUÉ PODEMOS APRENDER DE LOS RESULTADOS?

En el transcurso de este proyecto, logramos alcanzar los resultados esperados gracias a un buen desarrollo que se hizo en el modelo, un modelo que fue respaldado por buenas técnicas de optimización. Aunque la regularización no fue empleada directamente, el proceso de desarrollo en etapas y la aplicación de la optimización contribuyeron a la mejora de la precisión del modelo.



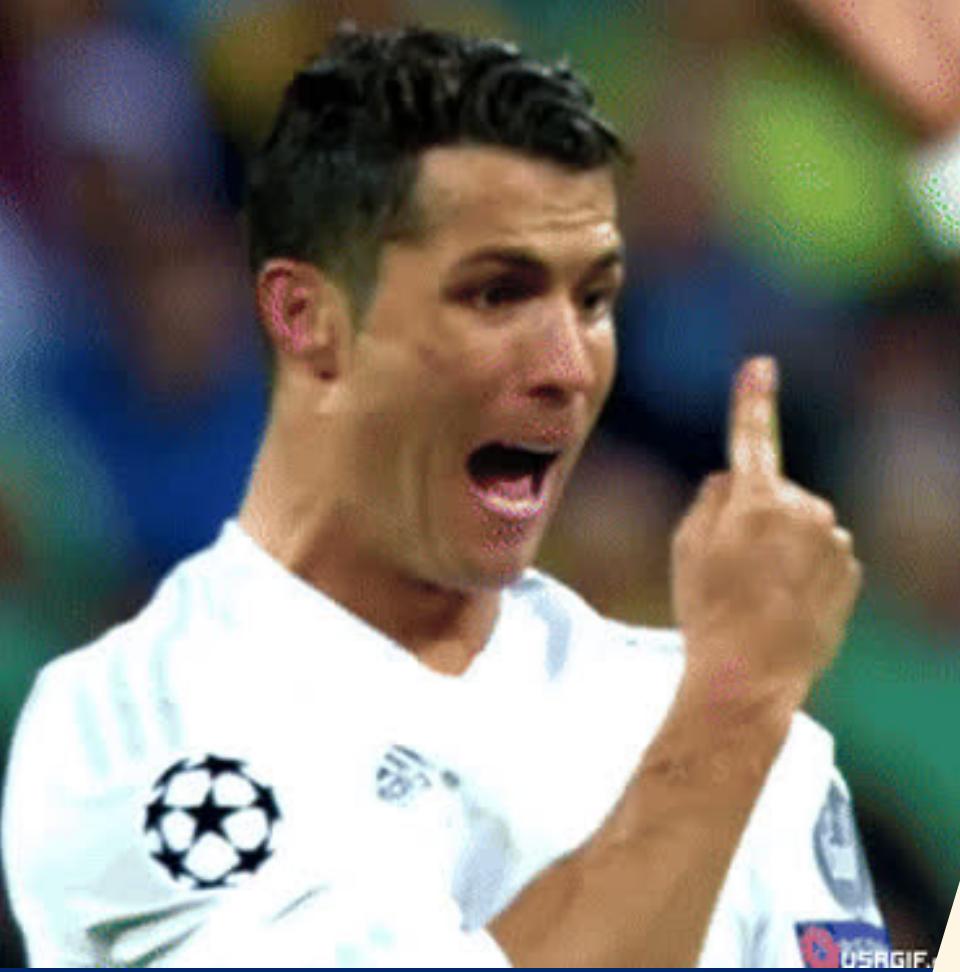
¿CUÁLES SON LAS CONTRIBUCIONES PRINCIPALES?

Nuestros resultados ofrecen una base sólida para futuras investigaciones en este campo, destacando la importancia de seguir explorando este tema y poder abordar más a fondo para buscar nuevos enfoques innovadores en el área de la inteligencia artificial y de la sociedad digital.

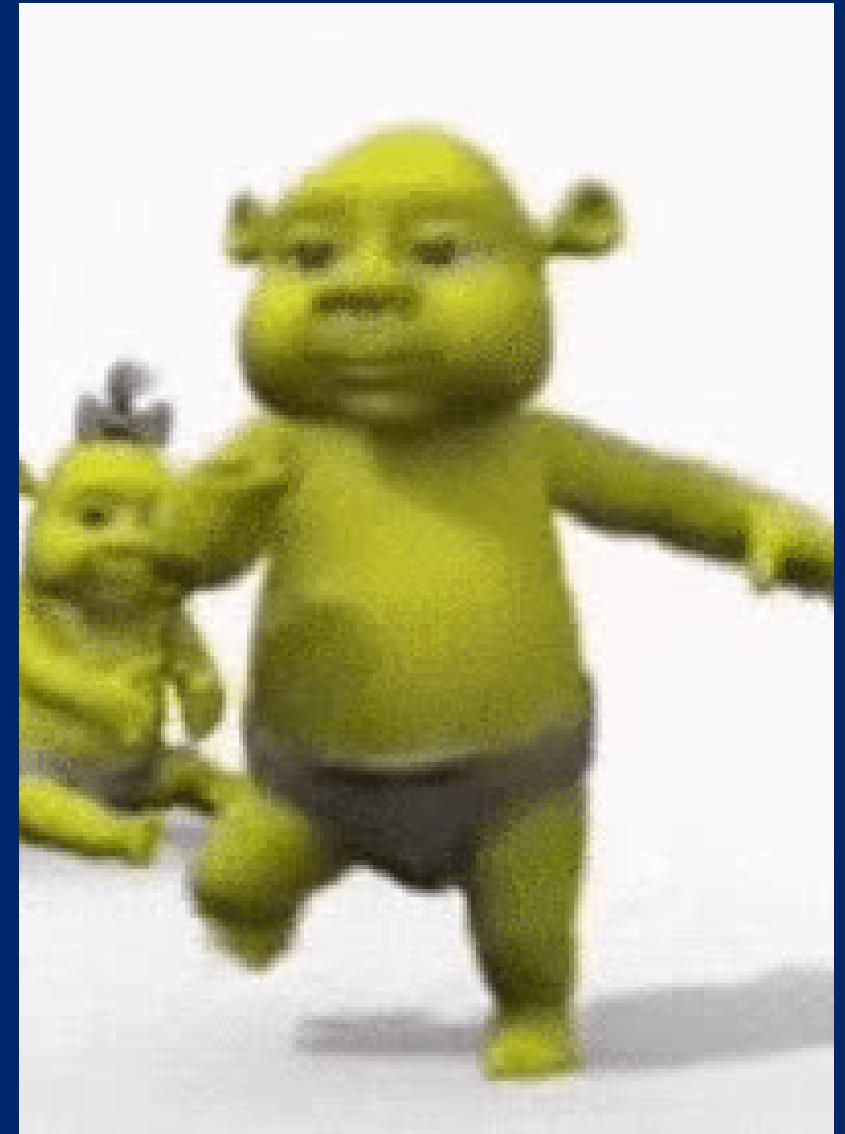


REFERENCIAS

- <https://es.wired.com/articulos/fake-news-generadas-por-ia-te-acecharan-en-proximas-elecciones>
- <https://yourmist.streamlit.app>
- <https://www.kaggle.com/datasets/bhavikjikadara/fake-news-detection>
- <https://www.kaggle.com/code/rhythmsage/fake-news-random-forest-99-naive-bayes-93-8>



**GRACIAS
POR SU
ATENCIÓN!**



¿PREGUNTAS?