

Etapla 2 Proyecto 1 Inteligencia de Negocios

Pipeline

1. Carga de documentos

En primer lugar, en nuestro notebook, realizamos la carga de los datos originales y hacemos un Split con el 20% para poder ir probando las funciones de procesamiento de texto (más adelante)

```
|: data=pd.read_excel("ODScat_345.xlsx")  
data_t, data_v = train_test_split(data, test_size=0.2, random_state=0)
```

```
|: data_t.head()
```

```
|:                                     Textos_espanol  sdg
```

2651	La movilización de las mujeres en manifestacio...	5
169	El papel combinado como comprador y proveedor ...	3
2993	El desempleo es particularmente alto entre los...	5
1148	Además, el espíritu empresarial es un vehículo...	4
2270	Aunque Sudáfrica tenía una Oficina de la Condi...	5

Luego utilizamos las funciones que mencione:

```
def quitarStopwords(words):
    new_words = []
    for word in words:
        if word is not None:
            if word not in spanish_stopwords:
                new_words.append(word)
    return new_words

def preProcesamiento(words):
    words = aMinuscula(words)
    words = eliminarNumeros(words)
    words = quitarPuntuacion(words)
    words = quitarStopwords(words)
    return words

#Esta version elimina convierte a misnusculas, quita puntuacion, quita stopwords
def preprocessing_text(texto):
    texto.dropna()
    texto.drop_duplicates()
    texto['Textos_espanol'] = texto['Textos_espanol'].apply(contractions.fix)
    texto['words'] = texto['Textos_espanol'].apply(word_tokenize)
    texto['words'] = texto['words'].apply(preProcesamiento)
    texto['words'] = texto['words'].apply(lambda x: ' '.join(map(str, x)))
    return texto['words']

def quitarPuntuacion(words):
    new_words = []
    for word in words:
        if word is not None:
            # Adjusted regular expression pattern to exclude colon
            new_word = re.sub(r'^\w\s:', '', word)
            if new_word != '':
                new_words.append(new_word)
    return new_words

def aMinuscula(words):
    new_words = []
    for word in words:
        if word is not None:
            new_word = word.lower()
            if new_word != '':
                new_words.append(new_word)
    return new_words

def eliminarNumeros(words):
    new_words = []
    for word in words:
        if not contieneNumero(word):
            new_words.append(word)
    return new_words

def contieneNumero(s):
    pattern = re.compile(r'\d')
    return bool(pattern.search(s))
```

Luego ponemos la vectorización, y la transformación

```
class Transformer_Representacion_Seleccion:
    def __init__(self, count_vectorizer):
        self.count_vectorizer = count_vectorizer
        self.palabras = None
        self.palabras_deseadas = None # Inicialización de las palabras deseadas

    def fit(self, X, y=None):
        # Ajustar el CountVectorizer y obtener las palabras (características)
        X_transformed = self.count_vectorizer.fit_transform(X)
        self.palabras = self.count_vectorizer.get_feature_names_out()

        # Crear un DataFrame temporal para realizar las selecciones de palabras relevantes
        X_todas_palabras = pd.DataFrame(X_transformed.toarray(), columns=self.palabras)

        # Seleccionar las palabras relevantes usando un ciclo tradicional
        self.palabras_deseadas = []
        for nombre in X_todas_palabras.columns:
            # Contar cuántas filas tienen valores distintos de 0 para cada palabra
            dato = (X_todas_palabras[nombre] != 0).sum()
            if dato > 1:
                # Si la palabra aparece en más de una fila, la agregamos a la lista
                self.palabras_deseadas.append(nombre)

        return self

    def transform(self, X):
        # Transformar los datos usando CountVectorizer
        X_transformed = self.count_vectorizer.transform(X)

        # Crear DataFrame con todas las palabras
        X_todas_palabras = pd.DataFrame(X_transformed.toarray(), columns=self.palabras)

        # Seleccionar solo las palabras relevantes
        palabras_a_usar = pd.Index(self.palabras_deseadas).intersection(X_todas_palabras.columns)

        # Retornar el DataFrame con las palabras relevantes
        return X_todas_palabras[palabras_a_usar].copy()
```

Finalmente creamos y ajustamos el pipeline con los parámetros y funciones que creamos.

```
# ajustamos el pipeline con RandomForestClassifier y los mejores parametros
pipelon = Pipeline([
    ('preprocess', FunctionTransformer(preprocessing_text)),
    ('representacion', Transformer_Representacion_Seleccion(CountVectorizer())),
    ('clf', RandomForestClassifier(max_depth=42, max_features=102, random_state=111)),
])
```

Persistencia del modelo en .joblib

```
joblib.dump(pipelon, 'pipelon.joblib')  
  
pipelon_cargado = joblib.load('pipelon.joblib')
```

Validación con los datos que arroja el modelo de la primera etapa

```
#validacion  
x_test = pd.read_csv("ValidacionX.csv", sep=',')  
y_test = pd.read_csv("ValidacionY.csv", sep=',')  
y_test = y_test['sdg']
```

Metricas

```
y_pred = pipelon_cargado.predict(data_v)  
  
print("Metricas:\n")  
print(classification_report(data_v['sdg'], y_pred))
```

Metricas:

	precision	recall	f1-score	support
3	0.99	0.97	0.98	269
4	0.97	0.98	0.97	266
5	0.97	0.98	0.97	275
accuracy			0.98	810
macro avg	0.98	0.98	0.98	810
weighted avg	0.98	0.98	0.98	810

BACK

1. Dependencias

Este proyecto incluye varias dependencias clave relacionadas con la manipulación de texto y el preprocesamiento de datos para un pipeline de clasificación de opiniones.

-FastAPI: Framework utilizado para crear la API con dos principales endpoints: uno para realizar predicciones y otro para reentrenar el modelo.

- Pydantic: Se usa para definir y validar los datos de entrada.
- nltk: Librería utilizada para tokenización y manejo de stopwords en español.
- Scikit-learn y Joblib: Para cargar, entrenar y guardar el modelo
RandomForestClassifier

Preprocesamiento de texto

Funciones clave:

- preprocessing_text(): Esta función elimina contracciones, convierte a minúsculas, elimina números, puntuación y palabras vacías.
- Funciones auxiliares: Se incluyen funciones como aMinuscula(), quitarPuntuacion(), eliminarNumeros(), y quitarStopwords() para refinar el texto antes de su transformación en datos numéricos.

El texto preprocesado es esencial para que el modelo de clasificación funcione de manera efectiva al convertir el texto en datos numéricos que el modelo pueda interpretar.

Transformer_Representacion_Seleccion

La clase Transformer_Representacion_Seleccion utiliza CountVectorizer para transformar el texto en una representación numérica y seleccionar las palabras más relevantes basadas en su frecuencia.

Pipeline de Clasificación

El pipeline principal se estructura de la siguiente manera:

- Entrenamiento: El modelo es un RandomForestClassifier que se entrena usando un conjunto de datos de entrenamiento preprocesado y vectorizado.
- Predicción: Usa el modelo entrenado para predecir la clase de opiniones y proporciona probabilidades asociadas con cada clase (ODS 3, 4 y 5).

```
prediccion = {  
    "opinion": i+1,  
    "clase_predicha": clase_predicha,  
    "probabilidades": {  
        "ODS 3": prob_ods3+'%',
```

```
"ODS 4": prob_ods4+'%',
```

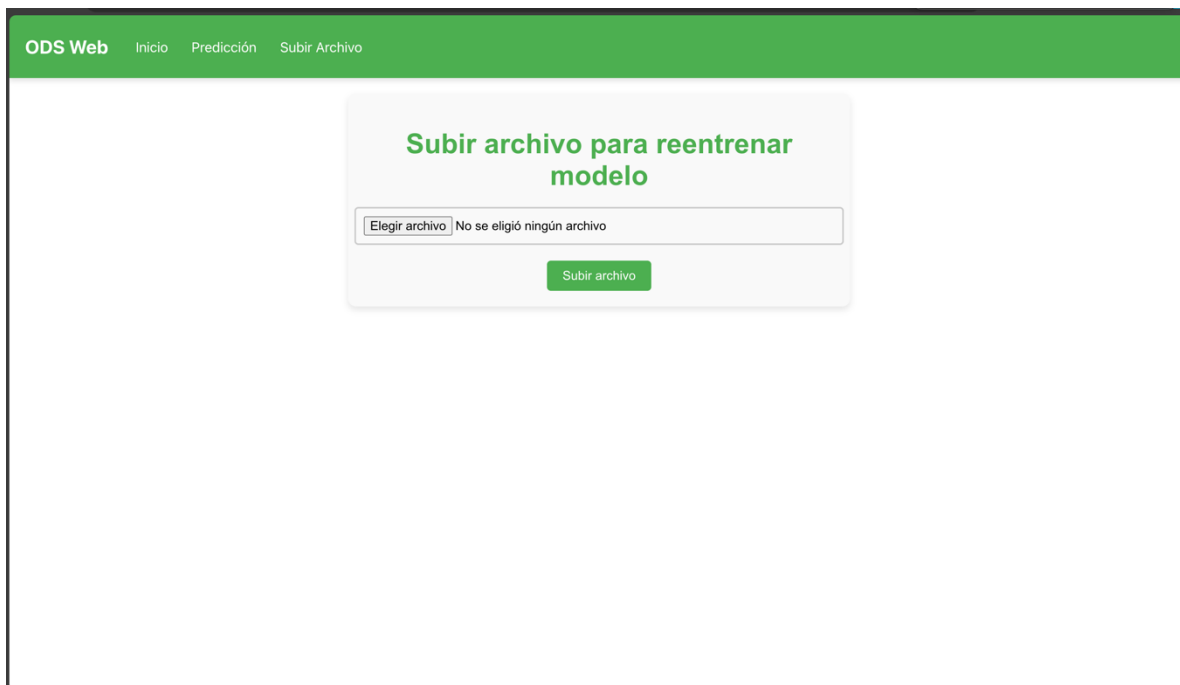
```
"ODS 5": prob_ods5+'%'
```

```
}
```

```
}
```


-Reentrenamiento: Permite reentrenar el modelo con un nuevo archivo de datos subido por el usuario, combinándolo con datos previos para mejorar el modelo.

FRONT:




The screenshot shows the 'ODS Web' interface with a green header bar containing the navigation menu: 'Inicio', 'Predicción', and 'Subir Archivo'. The main content area features a light gray box with the heading 'Subir archivo para reentrenar modelo'. Below this heading is a file selection interface with a text input field containing 'Elegir archivo' and 'No se eligió ningún archivo'. A green 'Subir archivo' button is positioned below the input field.

[ODS Web](#) [Inicio](#) [Predicción](#) [Subir Archivo](#)




Objetivos de Desarrollo Sostenible (ODS)

Los Objetivos de Desarrollo Sostenible son un llamado universal a la acción para poner fin a la pobreza, proteger el planeta y mejorar las vidas y las perspectivas de las personas en todo el mundo.




ODS 3: Salud y Bienestar

Garantizar una vida sana y promover el bienestar para todos en todas las edades es esencial para el desarrollo sostenible.



ODS 4: Educación de Calidad

Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos.



ODS 5: Igualdad de Género

Lograr la igualdad entre los géneros y empoderar a todas las mujeres y las niñas.

[ODS Web](#) [Inicio](#) [Predicción](#) [Subir Archivo](#)

Predicción de ODS

Opinión 1:

[Agregar otra opinión](#) [Enviar](#)

3 Entrenamientos.

Re-entrenamiento completo con todos los datos

Consiste en reentrenar el modelo utilizando tanto los datos antiguos como los nuevos en su totalidad, combinando ambos conjuntos de datos para crear un nuevo modelo desde cero. En este enfoque, se considera que los datos antiguos siguen siendo relevantes para el desempeño del modelo, junto con la nueva información que se ha agregado.

- Ventaja: Permite aprovechar al máximo toda la información disponible, mejorando la capacidad del modelo para generalizar en diferentes situaciones.

- Desventaja: Este enfoque puede ser computacionalmente costoso, especialmente si el conjunto de datos ha crecido significativamente con el tiempo, lo que puede llevar a tiempos de entrenamiento prolongados.

Re-entrenamiento incremental

El re-entrenamiento incremental se enfoca en actualizar el modelo existente con los nuevos datos, sin necesidad de volver a entrenar desde cero. Aquí, el modelo ajusta sus parámetros en función de los nuevos ejemplos, pero mantiene las características aprendidas de los datos antiguos.

- Ventaja: Es eficiente en términos de tiempo y recursos, ya que solo requiere procesar los nuevos datos en lugar de reentrenar todo el conjunto de datos.

- Desventaja: Existe el riesgo de que el modelo se vea influenciado desproporcionadamente por los nuevos datos y pierda la capacidad de generalizar bien en contextos más amplios.

Re-entrenamiento con un subconjunto representativo de los datos antiguos

Este enfoque utiliza una selección de los datos antiguos junto con los nuevos para reentrenar el modelo. El subconjunto de datos antiguos es seleccionado de manera que sea representativo del comportamiento general que se desea mantener, reduciendo así la necesidad de procesar el conjunto de datos completo.

- Ventaja: Balancea el uso de datos históricos sin sobrecargar el proceso de entrenamiento, permitiendo que el modelo se mantenga actualizado sin perder información clave.

- Desventaja: La selección de un subconjunto representativo puede ser difícil y, si no se hace adecuadamente, puede llevar a un modelo que no generalice bien.

En esta etapa del proyecto, se implementó el enfoque de re-entrenamiento completo con todos los datos, utilizando tanto los datos antiguos como los nuevos. Este enfoque asegura que el modelo siga aprovechando el contexto previo mientras incorpora la nueva información, garantizando una mayor capacidad de generalización en diferentes escenarios.

Trabajo en Equipo - División de roles y tareas

Leonardo Rangel:

- Roles:

Lider de proyecto: Está a cargo de la gestión del proyecto. Define las fechas de reuniones, pre-entregables del grupo y verifica las asignaciones de tareas para que la carga sea equitativa. Se encarga de subir la entrega del grupo. Si no hay consenso sobre algunas decisiones, tiene la última palabra.

Ingeniero de datos : Es responsable de velar por la calidad del proceso de automatización relacionado con la construcción del modelo analítico.

Ingeniero de software responsable de desarrollar la aplicación final: Se encarga de gestionar el proceso de construcción de la aplicación.

Tareas:

Crear el pipeline y ajustarlo. Conectar el back y el front.

Retos: Implementar el pipeline y conectar back y front.

Puntos: 33,33 puntos

Horas trabajadas en el proyecto: 15

Francois Morales:

- Roles:

Ingeniero de software responsable del diseño de la aplicación y resultados: Se encarga de liderar el diseño de la aplicación y de la generación del video con los resultados obtenidos.

Ingeniero de software responsable de desarrollar la aplicación final: Se encarga de gestionar el proceso de construcción de la aplicación.

Tareas:

Crear el Backend.

Retos: Con el joblib, implementar un código que use el modelo

Puntos: 33,33 puntos

Horas trabajadas en el proyecto: 15

Santiago Molina:

- Roles:

Ingeniero de software responsable del diseño de la aplicación y resultados: Se encarga de liderar el diseño de la aplicación y de la generación del video con los resultados obtenidos.

Ingeniero de software responsable de desarrollar la aplicación final: Se encarga de gestionar el proceso de construcción de la aplicación.

Tareas:

Crear el Front.

Retos: Hacer un front bonito y que se conecte al back.

Puntos: 33,33 puntos

Horas trabajadas en el proyecto: 15