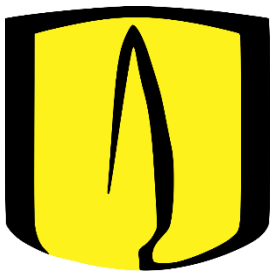
Proyecto 1 Parte 2

Fondo de Poblaciones de las Naciones Unidas (UNFPA)



Integrantes (Grupo 15)

Sara Sofia Cárdenas Rodríguez - 202214907

Daniel Felipe Diaz Moreno - 202210773

Juan Sebastián Urrea López - 201914710

Universidad de Los Andes

Departamento de Ingeniería de Sistemas y Computación

Inteligencia de Negocios - ISIS 3301

Bogotá D.C., Colombia

2024

Tabla de contenido

1. Implementación realizada por el ingeniero de datos

1.1. Proceso de automatización de la preparación de datos

El proceso de preparación de datos se dividió en 3 fases incorporadas por el pipeline de scikit-learn. Estas clases incorporaron a BaseEstimator y TransformerMixin para obtener ciertos métodos de los transformadores tradicionales para estos transformadores customizados. A continuación, se presentan las 3 fases:

Text_preprocessing: Usa la clase TextPreprocessor. Tiene los métodos fit, transform, process y normalizar palabra. Así, se utilizan las stopwords, el stemmer, el lemmatizer el cuarto método, para obtener textos tratados con palabras más útiles para el análisis. En el quinto método, se eliminan diacríticos y se usan expresiones regulares para eliminar caracteres no alfabéticos.

```python
class TextPreprocessor(BaseEstimator, TransformerMixin):

    def fit(self, X, y=None):
        return self  # No necesario para preprocesar texto

    def transform(self, X):
        return X.apply(self.process)

    def process(self, texto):
        # Corrige contracciones y tokeniza el texto
        texto = contractions.fix(texto)
        palabras = word_tokenize(texto)

        # Define stopwords y herramientas de stemmer/lemmatizer
        sw = set(stopwords.words('spanish'))
        stemmer = SnowballStemmer('spanish')
        lemmatizer = WordNetLemmatizer()

        # Realiza todas las transformaciones en un solo paso
        palabras_procesadas = [
            lemmatizer.lemmatize(stemmer.stem(self.normalizar_palabra(palabra.lower())))
            for palabra in palabras
            if palabra not in sw
        ]

        # Devuelve el texto procesado
        return ' '.join(palabras_procesadas)

    def normalizar_palabra(self, palabra):
        # Normaliza los caracteres para eliminar diacríticos (acentos, tildes)
        nfkd_form = unicodedata.normalize('NFKD', palabra)
        palabra_sin_diacriticos = ''.join([c for c in nfkd_form if not unicodedata.combining(c)])

        # Elimina cualquier carácter no alfabético (puntuación, números, etc.)
        palabra_limpia = re.sub(r'[^a-zA-Z]', '', palabra_sin_diacriticos)

        return palabra_limpia
```

Vectorization: Usa la clase TextVectorizer. Tiene los métodos init, fit y transform. Allí, se utiliza CountVectorizer con nm gramas de n = 1 y m = 3, dejando solo las palabras que aparecen en mínimo dos documentos. El resultado es un DataFrame con las columnas del vocabulario obtenido

```python
class TextVectorizer(BaseEstimator, TransformerMixin):

    def __init__(self):
        self.vectorizer = CountVectorizer(ngram_range=(1, 3), min_df = 2)

    def fit(self, X, y=None):
        return self.vectorizer.fit(X)

    def transform(self, X):
        return pd.DataFrame(self.vectorizer.transform(X).toarray(), columns=self.vectorizer.get_feature_names_out())
```

Scaling: Usa la clase FeatureScaler. Tiene los métodos init, fit y transform. Se usa StandardScaler sin centrar los datos antes de escalar. El resultado es un DataFrame con las columnas del vocabulario obtenido y los valores estandarizados

```python
class FeatureScaler(BaseEstimator, TransformerMixin):
    def __init__(self):
        self.scaler = StandardScaler(with_mean=False)

    def fit(self, X, y=None):
        return self.scaler.fit(X)

    def transform(self, X):
        return pd.DataFrame(self.scaler.transform(X), columns=X.columns)
```

1.2. Construcción del modelo

Se utilizó el mejor modelo de la entrega pasada de acuerdo con sus métricas para un algoritmo de clasificación (exactitud, precisión, recuerdo, y puntaje f1). Por ello, se utilizó GradientBoostingClassifier con 300 estimadores, una máxima profundidad de 5 y un estado aleatorio fijo. Esta fase fue añadida en el pipeline con el nombre de **classification,** luego del preprocesamiento anterior

```python
# Define the pipeline
pipeline = Pipeline([
    ('text_preprocessing', TextPreprocessor()),
    ('vectorization', TextVectorizer()),
    ('scaling', FeatureScaler()),
    ('classification', GradientBoostingClassifier(n_estimators=300, max_depth=5, random_state=1234))
])
```

1.3. Persistencia del modelo

Se guardó el modelo de pipeline descrito anteriormente, pero entrenado con los datos suministrados, con el fin de tener un punto de inicio. La aplicación permite cargar (load) y sobrescribir (dump) este archivo .joblib para las actividades de clasificación y reentrenamiento.

El método load se encuentra al inicio del archivo:

```
# Cargar el pipeline previamente entrenado
pipeline = load('pipeline.joblib')
```

Por otro lado, el dumb se realiza en el endpoint de /retrain, que se enseñará más adelante en la sección correspondiente:

```
# Guardar el modelo actualizado
dump(pipeline, 'pipeline.joblib')
```

1.4. Acceso por medio de la API REST

El backend de la aplicación fue implementado con Python. En particular, Uvicorn fue utilizado como un ASGI (Asynchronous Server Gateway Interface) para conformar una interfaz entre servidores asíncronos, frameworks y la aplicación. FastAPI fue el framework utilizado para construir la API en el entorno web. La comunicación se realiza a través de objetos JSON.

En la siguiente imagen se evidencia la creación básica de la API, donde se realizaron algunos ajustes de CORS para poder ser llamada desde el navegador. Asimismo, se crean dos clases para facilitar la gestión de peticiones.

```
# Inicializar la aplicación FastAPI
app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Modelo para la solicitud de predicción
class PredictionRequest(BaseModel):
    Textos_espanol: list


# Modelo para la solicitud de reentrenamiento
class RetrainingRequest(BaseModel):
    Textos_espanol: list
    sdg: list
```

En el siguiente fragmento de código se ejecuta al correr la aplicación, y permite que el backend corra en el localhost en el puesto 8000.

```
# Para correr la aplicación
if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

Puede encontrarse documentación más ejemplificada en el README.md de la carpeta backend del repositorio

1.4.1. Endpoint POST /predict

Este endpoint retorna las predicciones del ODS para un conjunto de textos, junto a las probabilidades de pertenencia asociadas a las clases 3, 4 y 5 para cada una de estas predicciones

Entradas del body: Una lista de textos en español, en particular comentarios relacionados a los ODS, se envía en la llave "Textos_espanol".

Salidas: Una lista de predicciones del ODS en la llave "predicciones". También retorna en la llave "probabilidades" una lista de listas para mostrar las probabilidades de pertenencia a las clases de ODS pertenecientes a cada texto recibido

```
# Endpoint 1: Predicción
@app.post("/predict")
async def predict(request: PredictionRequest):

    # Extraer los datos de la solicitud
    textos = pd.Series(request.Textos_espanol).astype(str)

    # Usar el pipeline para hacer predicciones
    predicciones = pipeline.predict(textos)
    probabilidades = pipeline.predict_proba(textos)

    # Devolver las predicciones y probabilidades
    return {"predicciones": predicciones.tolist(), "probabilidades":
probabilidades.tolist()}
```

1.4.2. Endpoint POST /retrain

Este endpoint permite el reentrenamiento del modelo a partir de un conjunto de textos ya clasificados o etiquetados, es decir, cuentan con la variable objetivo.

Como hay **tres formas** de efectuar el reentrenamiento bajo este aprendizaje supervisado basado en un modelo (en vez de instancias) que utiliza un algoritmo de clasificación, discutiremos cada una para elegir la metodología a implementar:

Juntar los datos iniciales y los nuevos datos (los de todos los reentrenamientos): Esta metodología se alinea mayoritariamente con el aprendizaje en batch. Batch learning imposibilita el aprendizaje incremental directamente, por lo que siempre usa todos los datos disponibles. Por esto, cada reentrenamiento tomaría tanto los nuevos datos como los viejos para procesarlos como un todo, sobrescribiendo el modelo anterior. Como requiere una alta cantidad de tiempo y recursos computacionales, normalmente se ejecutaría de forma offline, cuando no se están realizando predicciones (Géron, 2022).

Esta metodología es menos eficiente y requiere almacenar todos los datos de entrenamiento iniciales y todos los de reentrenamiento. Es decir que, por cada reentrenamiento, estos nuevos datos deben ser persistidos. No obstante, esto puede tener sentido cuando los datos brindados para reentrenamiento son mucho más grandes que el conjunto original.

Nótese que no solo se debería guardar los datos iniciales y los últimos datos de rentrenamiento, ya que no se tendría en cuenta el impacto de los reentrenamientos anteriores, por lo que esencialmente todo trabajo de reentrenamiento preliminar sería en vano. Esto lo haría un modelo no incremental en toda norma, lo cual dificulta el cumplimiento de los objetivos del negocio.

Tomar solo los nuevos datos: Esta metodología también se alinea principalmente con el aprendizaje en batch. Batch learning imposibilita el aprendizaje incremental directamente, por lo que siempre usa los datos disponibles. Por esto, cada reentrenamiento tomaría solamente los nuevos datos, sobrescribiendo el modelo anterior. Como requiere una alta cantidad de tiempo y recursos computacionales, normalmente se ejecutaría de forma offline, cuando no se están realizando predicciones (Géron, 2022).

Esta metodología es mucho más eficiente que la anterior, ya que ni siquiera requiere almacenar los datos iniciales, al solo necesitar el primer modelo entrenado para su primera ejecución de predicciones en el otro endpoint. No obstante, los reentrenamientos no serían incrementales, por lo que se perdería el progreso del anterior modelo y los resultados predictivos o métricas podrían cambiar drásticamente entre reentrenamientos. Esto podría llevar a una pérdida de información valiosa que se vislumbraría en la afectación de los parámetros internos del algoritmo, disminuyendo la capacidad de generalización de igual forma.

Transferir el aprendizaje (partial fit): Esta metodología se alinea más con el aprendizaje en línea. Online learning permite el aprendizaje incremental efectuado sobre la marcha. Por esto, cada reentrenamiento tomaría solamente los nuevos datos, sobrescribiendo el modelo anterior, pero teniendo en cuenta su información valiosa. Este tipo de aprendizaje se caracteriza por su rápida adaptación a nuevas entradas y el buen rendimiento al poder operar sobre menos recursos computacionales. Por ello, puede reentrenarse usando datos que puedan exceder la memoria principal del servidor. De esta forma, este procedimiento puede realizarse cuando la aplicación está en línea en otros endpoints (Géron, 2022). Tampoco habría grandes cambios entre entrenamientos si los conjuntos de datos de reentrenamiento son pequeños o moderados.

Esta metodología es eficiente tanto en tiempo como en recursos computacionales, sumado al beneficio de no olvidar el impacto de la información brindada desde la creación del modelo. Por ello, sería el modelo más incremental y capaz de generalizar a mediano y largo plazo. Sin embargo, su implementación es más demandante, al requerir transferencia de conocimiento y no simplemente un reemplazo del modelo o una persistencia de datos. Por último, su efectividad puede decaer cuando el nuevo conjunto de datos tiene un tamaño mucho mayor al inicial.

La alternativa escogida e implementada es la primera, dado que se desea mantener flexibilidad en el tamaño del conjunto de reentrenamiento, favoreciendo grandes conjuntos de datos. También se escogió por su facilidad de implementación a costa de un ligero aumento del tiempo de ejecución y uso de recursos computacionales

Por ello, esta es la información del endpoint:

Entradas del body: Una lista de textos en español, en particular comentarios relacionados a los ODS, se envía en la llave "Textos_espanol". También se envía una lista de etiquetas o valores objetivo que indican el ODS al que pertenece cada texto en la llave “sdg”. Ambas listas son de la misma cardinalidad.

Salidas: Las 4 métricas del modelo reentrenado **(**exactitud, precisión, recuerdo, y puntaje f1) se presentan en las llaves “accuracy”, “precision”, “recall”, y “f1_score” respectivamente. También se retorna la nube de palabras en la llave “words”, como tres listas para las tres clasificaciones respectivamente. La ejecución del endpoint sobrescribe el modelo. joblib anterior

```python
# Endpoint 2: Reentrenamiento del modelo
@app.post("/retrain")
async def retrain(request: RetrainingRequest):

    # Extraer los datos de la solicitud
    textos = pd.Series(request.Textos_espanol).astype(str)
    etiquetas = pd.Series(request.sdg).astype(int)

    # Entrenar un nuevo modelo con los datos proporcionados
    pipeline, metrics = train_model(textos, etiquetas)

    # Guardar el modelo actualizado
    dump(pipeline, 'pipeline.joblib')

    return metrics
```

2. Desarrollo de la aplicación y justificación

2.1. Usuarios y roles

Esta aplicación puede ser utilizada por trabajadores de mediano y alto rango de la UNFPA y de ciertas entidades públicas. En especial, los roles de quienes utilizarían la aplicación en ambos tipos de organizaciones serían el CEO o representante de la entidad, las mesas directivas y las áreas involucradas con datos (recolectores de datos, científicos de datos, ingenieros de datos).

2.2. Conexión de la aplicación y el proceso de negocio apoyado

La aplicación se relaciona directamente con la automatización del proceso de negocio de análisis de información textual recopilada, que actualmente consume muchos recursos humanos y monetarios. Esto se logra a través de un modelo analítico subyacente que puede ser utilizado y entrenado por quienes deseen visualizar el proceso de clasificación para grandes volúmenes de datos

2.3. Importancia de la existencia de la aplicación

La aplicación es crucial para automatizar el análisis de datos, disminuir la carga sobre los expertos y mejorar la toma de decisiones dentro de la UNFPA respecto a los Objetivos de Desarrollo Sostenible, los cuales son centrales en su actividad social. Esto se debe a que para esta organización los datos sociodemográficos sirven de base para la formulación de políticas públicas de diversos alcances, por lo que en

este caso relacionar las opiniones ciudadanas con los ODS de forma automática favorece la utilización de estas grandes cantidades de información Adicionalmente, las entidades públicas pueden analizar sus datos directamente recolectados sobre esta materia para adecuar la prestación de sus servicios de acuerdo con la participación ciudadana recibida, con el fin de hacerlos más sostenibles y benéficos.

2.4. Funcionamiento de la aplicación

El frontend de la aplicación fue desarrollado con Next.js y TypeScript, utilizando Tailwind CSS para los estilos. Las visualizaciones se realizaron empleando las librerías react-chartjs-2, chart.js, y react-wordcloud. Estas herramientas de visualización se pueden instalar fácilmente mediante un gestor de paquetes como npm, que también fue utilizado durante el desarrollo del proyecto.

Se pueden visualizar los diseños o mockups de Figma en el repositorio dentro de la carpeta frontend y en el siguiente enlace https://www.figma.com/design/6oswJKPowOwiytbUl4OryE/BI-1---2?t=vO8rrv46Yx9K3Ode-1

En la página web, se encuentran las siguientes ventanas:

Acerca de nosotros: Muestra el propósito de la aplicación, al igual que información de la organización y el grupo del proyecto

Entrenar modelo: Permite reentrenar el modelo con un nuevo conjunto de datos en Excel. Utiliza el endpoint de POST /retrain. Por ello, muestra las cuatro métricas de clasificación mencionadas a forma de diagramas de torta cuando termina el proceso. También muestra las nubes de las palabras más significativas para los 3 ODS.

Clasificar archivo: Clasifica un archivo Excel con opiniones. Utiliza el endpoint de POST /predict. Por ello, muestra las proporciones de clasificación de los textos enviados con un diagrama de torta, al igual que los promedios de probabilidades por ODS en diagramas de barras

Clasificar opinión: Clasifica una opinión recibida como texto. Utiliza el endpoint de POST /predict. Por ello, muestra la clase a la que pertenece el texto junto a su probabilidad de pertenencia a las clases. También muestra las proporciones de clasificación de los textos anteriormente enviados con un diagrama de torta, al igual que los promedios de probabilidades por ODS en diagramas de barras para los textos anteriormente clasificados

3. Resultados

3.1. Video

Este es el link del video

3.2. Resumen de resultados

Se logró implementar todas las funcionalidades del frontend y el backend previstas, de acuerdo con los diseños y la elección de tecnologías.

Por ello, se pueden realizar clasificaciones de un texto a la vez, al igual que de archivos de Excel con texto y sin ODS asignado. Estas clasificaciones también otorgan la probabilidad de pertenencia de cada texto a los ODS en cuestión.

También se puede reentrenar el modelo sobre información existente, obteniendo las métricas de clasificación para conocer la bondad de dicho modelo. Este reentrenamiento también sobrescribe el archivo del modelo como es esperado.

3.2.1. Impacto de la aplicación al negocio y a Colombia

La aplicación suministrada ayuda a analizar grandes volúmenes de datos textuales, apoyando el proceso de negocio anteriormente identificado de forma eficaz. De esta forma, su utilización ayuda a disminuir el tiempo de análisis y favorece la generación de resultados accionables que favorecen la toma de decisiones en políticas públicas en Colombia, las cuales pueden tener un enfoque diferencial por territorios o por proveedor de información. Entonces, podemos decir que este producto servirá para priorizar recursos y planes de acción en la transformación hacia un futuro más sostenible y equitativo. En pocas palabras, puede haber una mejora de la eficiencia en la identificación de problemas sociales clave a través del análisis automatizado de opiniones

3.2.2. Pruebas de uso

Hay pruebas de los endpoints del backend en el README.md de la carpeta del backend.

Por parte del frontend, se realizaron pruebas de uso durante el video explicativo realizado por Sara Cárdenas. De igual forma, durante las pre-entregas los integrantes del grupo validaron la usabilidad e interactividad de la interfaz gráfica, tanto en los diseños como en la implementación. Adicionalmente, dos compañeros del curso confirmaron lo intuitivo de la interfaz gráfica y la utilidad de los resultados en ella. Ellos mencionaron que los gráficos son dicientes y correctamente diseñados, al igual que toda la parte funcional del modelo de clasificación se ve reflejada en la web y en los archivos del repositorio.

4. Trabajo en equipo

4.1. Roles y tareas realizadas por cada integrante

Integrante 1: Juan Sebastián Urrea López

Rol(es): Ingeniero de datos e Ingeniero de software responsable de desarrollar la aplicación final

Contribución: 33.3 / 100. **Horas dedicadas:** 15

Tareas:

- Velar por la calidad del proceso de automatización relacionado con la construcción del modelo analítico
- Adecuar la preparación de datos de la iteración anterior para que pueda ser utilizada en la aplicación
- Adecuar el pipeline de la iteración anterior para que pueda ser utilizado en la aplicación
- Gestionar el proceso de construcción de la aplicación
- Implementar el backend de la aplicación
- Realizar pruebas del backend de la aplicación
- Documentar las pruebas del backend de la aplicación

Integrante 2: Daniel Felipe Diaz Moreno

Rol(es): Ingeniero de software responsable del diseño de la aplicación y de desarrollar la aplicación final

Contribución: 33.3 / 100. **Horas dedicadas:** 14

Tareas:

- Liderar el diseño de la aplicación
- Realizar los mockups de la interfaz gráfica de la aplicación
- Gestionar el proceso de construcción de la aplicación
- Implementar el frontend de la aplicación
- Realizar pruebas del frontend de la aplicación
- Documentar las pruebas del frontend de la aplicación

Integrante 3: Sara Sofía Cárdenas Rodríguez

Rol(es): Líder de proyecto, Especialista estadístico e Ingeniera de software responsable de los resultados

Contribución: 33.3 / 100. **Horas dedicadas:** 14

Tareas:

- Estar a cargo de la gestión del proyecto
- Definir las fechas de reuniones, pre-entregables del grupo y verificar las asignaciones de tareas para que la carga sea equitativa
- Subir la entrega del grupo
- Reunir los resultados obtenidos
- Generar el video con los resultados obtenidos
- Encargarse de las rpeuabs estadísticas
- Realizar el documento de informe

4.2. Reuniones de grupo

Reunión de lanzamiento y planeación

Fecha: 20 de septiembre del 2024. **Integrantes:** Todos

Resumen: Se definieron los roles y la forma de trabajo del grupo. También se realizó una lluvia de ideas sobre la forma de resolver el proyecto. En especial, hubo un enfoque en la utilización de tecnologías, frameworks y herramientas. Finalmente, se establecieron las primeras acciones necesarias para el desarrollo del proyecto y se acordó la fecha de todas las reuniones posteriores, incluyendo las de seguimiento y finalización. Adicionalmente, se definió la utilización de notificaciones de avance por parte de los integrantes en fechas subsiguientes a las reuniones de seguimiento.

Reuniones de seguimiento

Fecha: 23 de septiembre del 2024. **Integrantes:** Todos

Resumen: En la semana 8, se presentaron los diseños para el frontend y el backend de la aplicación, en concordancia con las tecnologías seleccionadas. Se discutió la viabilidad de estos diseños y se propusieron cambios para antes de la implementación

Fecha: 7 de octubre del 2024. **Integrantes:** Todos

Resumen: En la semana 9, se presentaron los avances en la implementación para el frontend y el backend de la aplicación, en concordancia con los diseños finales. Se discutió la viabilidad de esta implementación y se propusieron cambios para antes de la entrega final

Notificaciones de avance

Fecha: 27 de septiembre del 2024. **Integrantes:** Todos

Resumen: En la semana 8, cada integrante reveló al grupo sus avances en los diseños y en la ejecución de sus tareas. Los demás integrantes avalaron los cambios

realizados luego de inspeccionar esta información. Así, cada integrante supo cómo empezar a implementar esta planeación

Fecha: 11 de octubre del 2024. **Integrantes:** Todos

Resumen: En la semana 9, cada integrante reveló al grupo sus avances en la implementación y en la ejecución de sus tareas. Los demás integrantes avalaron los cambios realizados luego de inspeccionar esta información. Así, cada integrante supo cómo terminar su implementación y casos de prueba.

Reunión de finalización

Fecha: 12 de octubre del 2024. **Integrantes:** Todos

Resumen: Se revisó el cumplimiento de las tareas asignadas y se reflexionó sobre la efectividad en la distribución del trabajo y la comunicación. No se identificaron mayores inconvenientes, por lo que se propusieron pocas acciones adicionales que se profundizarán a continuación. El grupo dio el aval para la entrega de esta iteración.

4.3. Retos enfrentados en el proyecto y formas de resolución

Los principales retos fueron conocer la forma de adecuación de los resultados de la entrega pasada, en especial la preparación de datos y la construcción de un pipeline que pudiera ser usado por una aplicación. Adicionalmente, la integración del backend y el frontend fue uno de los procesos más cruciales y cuidadosos del proyecto. Por esto, la cohesión de las tareas y las tecnologías usadas fue un desafío de un tamaño moderado, pero de gran importancia. Otro reto fue la selección de la metodología para reentrenar el modelo, ponderando tiempo de ejecución, costo computacional y facilidad de implementación. Por último, los plazos de entrega del proyecto y sus pre-entregables estuvieron delimitados por fechas límite muy cercanas, por lo que fue importante la comunicación y el trabajo efectivo.

4.4. Puntos para mejorar

El principal punto para mejorar es el inicio temprano de las actividades de un proyecto de este tipo o escala, ya que la limitante de tiempo es un problema conocido en este tipo de entregas, ya sea en un ambiente académico o profesional. Por parte del modelo y la aplicación, deben reducirse los tiempos de ejecución y reentrenamiento, ya sea usando otros mecanismos u optimizando el código. También se recomienda continuar con las buenas prácticas de comunicación y el trabajo efectivo en equipo

5. REFERENCIAS

Géron, A. (2022). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow : concepts, tools, and techniques to build intelligent systems (Third edition). O'Reilly Media, Inc.