INTELIGENCIA DE NEGOCIOS

ISIS3301

"Etapa 2. Automatización de analítica de textos"

Profesor: *Fabián Peña*

Grupo 5

Wyo Hann Chu Mendez – 202015066

Diana Alejandra Silva Álvarez – 201815366

Sofía Velásquez Marín – 202113334

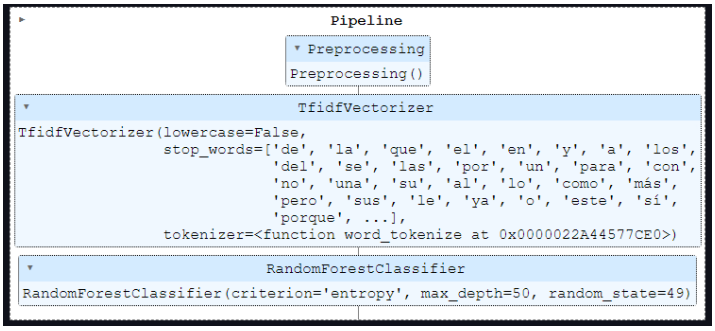
Tabla de contenido

Proceso de automatización del proceso de preparación de datos, construcción del modelo, persistencia del modelo y acceso por medio de API.

Para el proceso de automatización del proceso de preparación de datos se creó una clase “Preprocessing”. Para poder hacer la limpieza de los datos que nos llegan.

```python
from sklearn.base import BaseEstimator, TransformerMixin
import re
import ftfy

class Preprocessing (BaseEstimator, TransformerMixin):
  def fit(self, X, y=None):
      return self

  def remove_punctuation(self, text):
      # Use ftfy to fix text and remove punctuation points using regular expression
      cleaned_text = ftfy.fix_text(text)
      cleaned_text = re.sub(r'[^\w\s]', '', cleaned_text)
      return cleaned_text

  def transform (self, X):
      X['Textos_espanol'] = X['Textos_espanol'].apply(ftfy.fix_text)
      X['Textos_espanol'] = X['Textos_espanol'].apply(lambda x: x.lower())
      # Apply remove_punctuation function to 'Textos_espanol' column
      X['Textos_espanol'] = X['Textos_espanol'].apply(self.remove_punctuation)
      X["Textos_espanol"] = X["Textos_espanol"].astype(str)

      return X["Textos_espanol"]
```

Esta clase es la primera etapa por la que pasan los datos que le pasemos al pipeline. Después de procesado los datos, estos pasan por la vectorización de texto utilizando TF-IDF con tokenización y eliminación de las “stop words” definidas previamente y no se hace la conversión a minúsculas. Luego de esa etapa, son entregados a el “Random Forest Classifier” el cual combina la salida de múltiples árboles de decisión para alcanzar un solo resultado.



Este pipeline (modelo) es guardado en un archivo llamado “model.joblib” para poder utilizarlo en la API.

Para el acceso por medio de API, se implementa una API con el framework de FastAPI para que el usuario al ingresar un texto o un archivo, se le pueda entregar los resultados de la clasificación.

Para lograr esto, se crearon varios endpoints:

(/predict) Este endpoint recibe un JSON con el siguiente formato: { "Textos_espanol": “string” }. Donde “String” seria el texto a clasificar. Dicho texto se pasa por el modelo anterior mente exportado para poder clasificar a cuál ODS pertenece (3, 4 o 5).

```python
@app.post("/predict")
def make_predictions(dataModel: DataModel, ):
    df = pd.DataFrame(dataModel.dict(), columns=dataModel.dict().keys(), index=[0])
    df.columns = dataModel.columns()
    result = model.predict(df)
    return {
        "Textos_espanol": dataModel.dict()['Textos_espanol'],
        "sdg": int(result),
    }
```

(/predict_file) Este endpoint recibe un archivo, el cual debe ser un Excel o un csv, en caso contrario, se devuelve un código de error 400. Adicionalmente, se comprueba que tenga la columna “Textos_espanol”, debido a que de esa manera el API identifica que es dicha columna la que contiene los textos que debe clasificar. Al igual que en el endpoint anterior, se le pasan los datos a el modelo para ser procesados y para predecir a que ODS pertenece cada uno de los textos. Una vez se tiene el resultado de la predicción, se agrega una columna “sdg” con dichos resultados y se almacenan en un Excel y en un csv, que posteriormente pueden ser descargados.

```python
@app.post("/predict_file")
async def make_predictions_file(file: UploadFile=File(...)):

    try:
        #Si no termina en .xlsx  o .csv no se procesa
        if not file.filename.endswith((".xlsx", ".csv")):
            return JSONResponse(status_code=400, content={"message": "El archivo debe ser un excel o un csv"})

        #Se lee el archivo si es un excel
        if file.filename.endswith(".xlsx"):
            df = pd.read_excel(file.file)
        else:
            df = pd.read_csv(file.file)

        #Se verifica que el archivo tenga la columna Textos_espanol
        if not "Textos_espanol" in df.columns:
            return JSONResponse(status_code=400, content={"message": "El archivo no tiene la columna Textos_espanol"})

        result = model.predict(df)
        df['sdg'] = result

        # Crear una lista de diccionarios a partir del DataFrame
        data_list = df.to_dict(orient="records")

        # Convertir a csv y xlsx
        df.to_csv(predictions_csv, index=False)
        df.to_excel(predictions_xlsx, index=False)

        return data_list

    except Exception as e:
        return JSONResponse(content={"error": str(e)}, status_code=500)
```

(/download_predictions_csv o /download_predictions_xlsx) Estos endpoints permiten descargar el archivo correspondiente con la clasificación.

```python
# Descargar el archivo csv
@app.get("/download_predictions_csv")
def download_predictions():
    try:
        return FileResponse(predictions_csv, filename="predictions.csv")
    except Exception as e:
        print(e)
        return JSONResponse(status_code=500, content={"message": "Hubo un error descargando el archivo"})

# Descargar el archivo xlsx
@app.get("/download_predictions_xlsx")
def download_predictions():
    try:
        return FileResponse(predictions_xlsx, filename="predictions.xlsx")
    except Exception as e:
        print(e)
        return JSONResponse(status_code=500, content={"message": "Hubo un error descargando el archivo"})
```

Desarrollo de la aplicación y justificación.

Descripción del usuario/rol de la organización que va a utilizar la aplicación.

La aplicación fue diseñada con el objetivo de ayudar al Fondo de Población de las Naciones Unidas (UNFPA) mediante el desarrollo de un modelo de clasificación fundamentado en técnicas de aprendizaje automático. Este modelo tiene como objetivo clasificar, en base a un texto proporcionado, a que ODS pertenece. Sin embargo, el alcance del modelo está limitado a los ODS 3, 4 y 5, si se llegara a ingresar un texto que no está alineado con dichos ODS el modelo va a clasificarlo de manera errónea.

Aunque la aplicación fue diseñada para beneficiar a UNFRA, no se restringe únicamente a dicha organización. Cualquier persona / entidad interesada en los ODS puede verse beneficiado por la aplicación.

Importancia que tiene para ese rol la existencia de esta aplicación.

El Fondo de Población de las Naciones Unidas (UNFPA), al igual que otras organizaciones le es beneficiosa esta aplicación por las siguientes razones:

- Eficiencia: La aplicación automatiza la clasificación de texto de acuerdo con los Objetivos de Desarrollo Sostenible (ODS), lo que ahorra tiempo y recursos que, de lo contrario, se emplearían en laboriosas tareas manuales.
- Precisión: La aplicación proporciona resultados exactos y coherentes en la clasificación de texto, disminuyendo el riesgo de errores humanos y asegurando la calidad de la información evaluada. En este momento tenemos una precisión del 98,1111%.
- Facilita la toma de decisiones informadas: Al agilizar la identificación de cómo la información se relaciona con los ODS, la aplicación permite tomar decisiones más informadas en sus proyectos.

Opciones que tuvieron al momento de definir la aplicación.

Nos centramos en crear una herramienta que fuera fácil de usar para cualquier persona, independientemente de su nivel de experiencia. Para ello, diseñaron la aplicación con dos opciones simples: cargar un archivo .csv, .xlsx o escribir un texto

directamente. En todos casos, la aplicación proporciona resultados de la clasificación de los textos según los objetivos de Desarrollo Sostenible (3, 4 y 5).

Para el framework del front, teníamos 2 opciones en mente, React o Flutter. Nos decidimos por React, debido a que 2 de las 3 personas del grupo ya tenían conocimiento y experiencia sobre dicho framework, pero sobre Flutter solo lo tenia 1 de las 3 personas.

Para el framework del back, nos decidimos por FastAPI, principalmente porque en internet había muchos tutoriales de como desplegar un modelo de machine learning con dicho framework, y también debido a que 1 de las 3 personas del grupo ya había trabajado con FastAPI.

Aporte del equipo de estadística.

Este fue el comentario realizado por el equipo de estadística:

“Primero, quiero decir que es de gran importancia el análisis de textos y la preparación de datos en la creación del proyecto. Por esto, quiero resaltar que es una medida adecuada tener en cuenta los errores en la conversión de los caracteres en el texto. Esta problemática es importante y aún más al tratar diferentes formatos, ya que de lo contrario el análisis de estos se podría complicar y causar problemas. Aparte de esto, también me parece personalmente relevante el tema de los idiomas al manejar datos en español. Es algo adecuado siempre y cuando el modelo se limite a este lenguaje. Quitar los textos por fuera de este idioma ayuda a evitar confusiones o errores, lo cual mejora los resultados del modelo, mejorando la precisión y la eficacia.

Simplificar los textos me parecería algo correcto, pero también me pareció un punto bien tratado. En algunos casos, la puntuación en algunos textos podría tener información relevante para la clasificación de estos textos. Desde el análisis de datos, es algo importante y un punto positivo considerar qué tan necesario es simplificar la puntuación de los textos y hacerlos más simplificados o no.

Finalmente, y, en conclusión, el enfoque que tiene el equipo frente a la preparación y el manejo de los datos lo veo sólido y con bastante coherencia con buenas prácticas. Al manejar y categorizar o estandarizar los datos, podría garantizar el buen funcionamiento del modelo para que sea efectivo y con resultados precisos en su tarea. Además, la parte de documentar y de justificar el proceso es vital en el proyecto para el análisis de este, ya sea en la calidad de los datos o en la interpretación de resultados. Validar o verificar estos últimos es de mucha ayuda.”

Se tomaron en cuenta los comentarios realizados, se modificó el notebook para que la limpieza de los datos se quitaran las palabras en inglés. Sin embargo, los resultados obtenidos con dicho cambio fueron peores de los que teníamos. Se bajo la métrica F1 de un 98,1111% a un 96,8292%, por lo que se revirtieron los cambios.

Adicionalmente, presentaron comentarios sobre el UI/UX de la aplicación, principalmente dijeron que podríamos adicionar una sección para poner de que trata la aplicación y dar más contexto a los usuarios que la fueran a utilizar.

Resultados.

El desarrollo front de la aplicación se realizó por medio de FastAPI y React, para su desarrollo se hace el llamado del modelo desde el back (que debe ser iniciado con anterioridad antes de probar el modelo en el front), y por medio de la API, se crea la clase Texto para recibir y verificar la entrada de datos, en caso de que entre en el formato incorrecto, se le avisa por medio de un pop-up al usuario que ingrese un archivo en formato válido, se manda el texto o documento para que el modelo procese los datos.

```
function Texto() {
  const [formValues, setFormValues] = useState({
    texto_espanol: "",
    file: null,
  });
  const [validationState, setValidationState] = useState({
    texto_espanol: false,
    sdg: false,
    file: false,
    addFile: false,
  });
  const [prediction, setPrediction] = useState({});
  const [pfile, setPFile] = useState([]);

  const handleTextChange = (e) => {
    const placeholder = e.target.placeholder;
    const texto_espanol_e = e.target.value;
    setFormValues({ ...formValues, texto_espanol: texto_espanol_e });

    setValidationState((prevState) => ({
      ...prevState,
      texto_espanol:
        texto_espanol_e !== placeholder &&
        texto_espanol_e !== null &&
        texto_espanol_e !== undefined &&
        texto_espanol_e.length > 0,
    }));
  };
```

```
const clickSubmitFile = async () => {
  if (!validationState.addFile) {
    return;
  }
  const URL = "http://localhost:8000/predict_file";
  console.log(formValues.file);
  const formData = new FormData();
  formData.append("file", formValues.file);
  try {
    const requestOptions = {
      method: "POST",
      body: formData,
    };
    const result = await fetch(URL, requestOptions);
    if (result.status === 200) {
      console.log("OK");
      const data = await result.json();
      setPFile(data);
      setValidationState((prevState) => ({
        ...prevState,
        file: true,
        addFile: true,
      }));
    } else {
      alert("Suba un archivo valido");
    }
  } catch (error) {
    console.error(error);
  }
  console.log(validationState.file);
};
```

```
const clickSubmit = () => {
  if (!validationState.texto_espanol) {
    return;
  }

  const URL = "http://localhost:8000/predict";

  const data = {
    Textos_espanol: formValues.texto_espanol,
  };
  const requestOptions = {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(data),
  };

  fetch(URL, requestOptions)
    .then((response) => response.json())
    .then((data) => {
      console.log(data);
      setPrediction(data);
      setValidationState((prevState) => ({
        ...prevState,
        sdg: true,
      }));
    })
    .catch((error) => {
      console.error("Error fetching prediction:", error);
    });
};
```

Se realiza también el procesamiento para la descarga de los datos procesados tanto en formato CSV como en formato Excel (.xlsx), para que el usuario pueda obtener los resultados que necesite en el formato de su preferencia.

```
const downloadCSV = () => {
  const API_URL = "http://localhost:8000/download_predictions_csv";
  fetch(API_URL)
    .then((response) => {
      if (response.ok) {
        return response.blob();
      } else {
        throw new Error("Failed to download XLSX");
      }
    })
    .then((blob) => {
      const url = window.URL.createObjectURL(blob);
      const a = document.createElement("a");
      a.href = url;
      a.download = "predictions.csv";
      document.body.appendChild(a);
      a.click();
      window.URL.revokeObjectURL(url);
    })
    .catch((error) => {
      console.error("Error downloading CSV:", error);
    });
};
```

```
const downloadXLSX = () => {
  const API_URL = "http://localhost:8000/download_predictions_xlsx";
  fetch(API_URL)
    .then((response) => {
      if (response.ok) {
        return response.blob();
      } else {
        throw new Error("Failed to download XLSX");
      }
    })
    .then((blob) => {
      const url = window.URL.createObjectURL(blob);
      const a = document.createElement("a");
      a.href = url;
      a.download = "predictions.xlsx";
      document.body.appendChild(a);
      a.click();
      window.URL.revokeObjectURL(url);
    })
    .catch((error) => {
      console.error("Error downloading XLSX:", error);
    });
};
```

Trabajo en equipo.

- Líder de proyecto: Sofía Velásquez Marín – 202113334, estuvo a cargo de la gestión del proyecto, definición de tareas, reparto de responsabilidades, chequeo de trabajo realizado, bloqueantes y tiempo de gestión de la entrega. Si no hay consenso sobre algunas decisiones, tiene la última palabra.
- Ingeniero de datos: Wyo Hann Chu Mendez – 202015066, encargado de gestionar los datos que se van a usar en el proyecto y de las asignaciones de tareas sobre datos, el funcionamiento del joblib para las entradas nuevas de datos y su retorno hacia la API, y desarrollo del video.
- Ingeniero de software responsable del diseño de la aplicación y resultados: Sofía Velásquez Marín – 202113334, se encargó de liderar el diseño de la aplicación y de la generación del desarrollo del front de la App con React, el cómo recibe y retorna los datos para el usuario.
- Ingeniero de software responsable de desarrollar la aplicación final: Diana Alejandra Silva Álvarez – 201815366, Se encarga de gestionar el proceso de construcción a nivel de diseño de la aplicación, interfaz que se le muestra al usuario y la UI/UX.

Sobre el trabajo realizado el porcentaje de tiempo dedicado, de 100 puntos se otorgan 34 puntos a Wyo Hann Chu, 34 a Sofía Velásquez y 32 a Diana Silva. Se definieron reuniones semanales para chequear avances.