UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERIA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Universidad de los Andes
Facultad de Ingeniería

PROYECTO ENTREGA GRUPAL

Presentado a
Haydemar Nuñez

Presentado por
Tomas Acosta Bernal - 202011237
Diego Andres Rubio - 201816492
David Alvarez Polo - 201911318

BOGOTÁ DC - COLOMBIA
29 de Octubre 2023

Índice general

Capítulo 1

Segunda Parte

1.1. Descripción del proceso e implementación realizados por el ingeniero de datos

El ingeniero de datos llevó a cabo el siguiente procedimiento para la parte de automatización-preparación de datos y modelamiento para la aplicación:

1. **Carga de datos:**
   Se cargaron los datos desde un archivo de Excel en un DataFrame de Pandas.
2. **Preparación de datos:**
   Se realizaron las siguientes operaciones de limpieza y preparación de los datos:
   - Limpieza de caracteres no ASCII.
   - Conversión de caracteres a minúsculas.
   - Eliminación de signos de puntuación.
   - Eliminación de stop words.
   - Reemplazo de special coders por caracteres con tildes.
   - Reemplazo de números por su representación textual.
   - Tokenización.
   - Normalización.
3. **Entrenamiento de modelos:**
   Se entrenaron tres modelos de clasificación de textos:
   - Regresión logística.
   - Naive Bayes.
   - Random Forest.

1.1.1. Implementación

Para esta parte se realizaron las siguientes tareas:

1. **Pipeline**: Se implementó un pipeline que integraba las funciones necesarias para llevar a cabo las transformaciones esenciales de los datos. Este pipeline permitió gestionar eficazmente el flujo de datos y aplicar las preparaciones requeridas de manera sistemática.

```python
from sklearn.pipeline import Pipeline
from sklearn.base import BaseEstimator, TransformerMixin

class TextPreprocessor2(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

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

    def transform(self, X):
        p = engine()
        preprocessed_data = []

        for sentence in X:
            words = sentence.split()
            words = to_lowercase(words)
            words = replace_numbers(words, p)
            words = remove_punctuation(words)
            words = remove_non_ascii(words)
            words = remove_stopwords(words)
            words = remove_specialCoders(words)
            preprocessed_sentence = ' '.join(words)
            preprocessed_data.append(preprocessed_sentence)

        return pd.Series(preprocessed_data)

# Crear un pipeline
preprocessing_pipeline = Pipeline([
    ('text_preprocessor', TextPreprocessor2()),
    ('vectorizer',CountVectorizer(max_features=3000))
])
```

Figura 1 Pipeline implementado

2. **Framework para desarrollo de API**: Se utilizó Flask para desarrollar una API REST que permitiera acceder a los resultados de los modelos.

1.1.2. Código

Para la creación de nuestra aplicación es relevente destacar algunos fragemntos de código que se presentan a continuación:

1. **Importación de las librerias:** Realizamos la importación de todas las librerías necesarias para desplegar nuestra aplicación. Entre estas se destaca **Streamlit**. Esta nos permite hacer nuestra aplicación web y conectarla con nuestro notebook de manera sencilla.

```python
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import proyecto_grupo13
from pandas_profiling import ProfileReport
import plotly.express as px
```

Figura 2 Librerias importadas

2. **Carga de datos y creación de gráficos:** Se realizar la respectiva carga de datos y luego se hace lo siguiente:
   - Se crea un diccionario llamado dataGraph que contiene los datos del gráfico de barras. La clave Modelo contiene los nombres de los modelos de clasificación. La clave Exactitud contiene las exactitudes de los modelos de clasificación.
   - Se crea un DataFrame de Pandas llamado df a partir del diccionario dataGraph
   - Crea un gráfico de barras con la biblioteca plotly.express. El gráfico muestra la exactitud de cada modelo de clasificación.

- Se utiliza la función `st.plotly_chart()` para mostrar el gráfico de barras en la aplicación web.

```
data = pd.read_excel('cat_345.xlsx')

dataGraph = {
    'Modelo' : ['Regresión Logística', 'Naive Bayes', 'Random Forest'],
    'Exactitud' : [proyecto_grupo13.accuracy2, proyecto_grupo13.accuracy3, proyecto_grupo13.accuracy4]
}

df = pd.DataFrame(dataGraph)
```

Figura 3 Carga y gráficos

3. **Modelado en la aplicación:** A continuación se muestra el codigo para modelar uno de los algoritmos usados en la anterior etapa en la aplicación web:

```
## Describir brevemente cada modelo y razones de elección
## Mostrar matrices de confusión creadas
st.title('Resultados')
st.write('_Nota: Las exactitudes son valores entre 0 y 1_')

## Gráficos sobre resultados
st.title('Gráfico comparativo sobre modelos')
fig = px.bar(df, x='Modelo', y='Exactitud', title='Gráfico de Barras')
st.plotly_chart(fig)

st.header('Regresión Logística Multivariada')
st.markdown('<div style="text-align: justify;">Este modelo nos permite analizar opiniones y relacionarlas con l
st.text('')
st.write('Exactitud del modelo de Regresión Logística:', proyecto_grupo13.accuracy2)
st.write('**Razones de elección:**')
st.markdown("""
    - Capacidad para manejar problemas de clasificación multiclase
    - Interpretabilidad
    - Suposiciones
    """)
st.subheader('Matriz de confusión - Regresión Logística')
st.image('./confusion_matrix_logist.png')
st.subheader('Predicciones Modelo Test Regresión Logística')
st.write(proyecto_grupo13.dataFramePredictedLogit)
```

Figura 4 Modelamiento en la aplicación

1.2. Desarrollo de la aplicación y justificación

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

:
El rol que va a utilizar la aplicación es el analista de datos. Este va a ser el responsable de realizar análisis de datos para la organización y para ello necesitará de la aplicación web para visualizar los resultados y así realizar dicho análisis de manera exitosa.

1.2.2. Conexión entre aplicación y el proceso de negocio

La aplicación web desarrollada en este proyecto apoyará el proceso de análisis de datos de la organización. El analista utilizará la aplicación para mostrar los resultados de sus análisis, lo que le permitirá tomar mejores decisiones a futuro y que le beneficien a la empresa. En este caso, sobre los mejores modelos o las situaciones en las que es más indicado usar uno en específico.

1.2.3. Importancia de la aplicación para el rol

Esta aplicación es de gran importancia para el analista por los siguientes motivos:

- Le permitirá compartir sus análisis con otros usuarios de la organización.
- Podrá obtener retroalimentación de otros usuarios sobre sus análisis.
- Logrará identificar tendencias y oportunidades en los datos.

1.3. Resultados-Aplicación

A continuación se presenta nuestra aplicación implementada:

Figura 5 Modelamiento en la aplicación

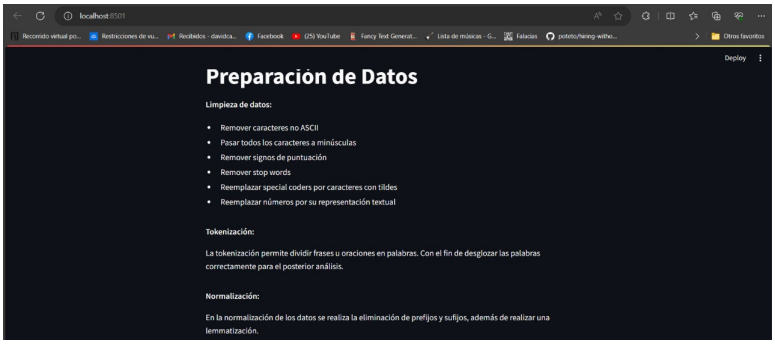

Figura 6 Visualización de la aplicación

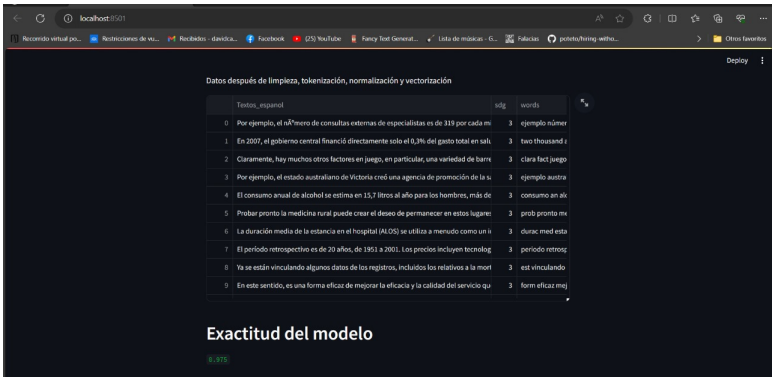

Figura 7 Visualización de la aplicación

Figura 8 Visualización de la aplicación

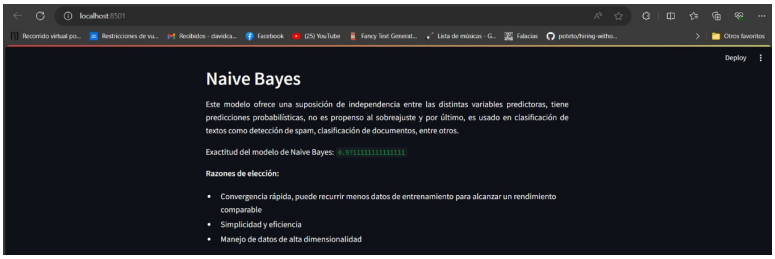

Figura 9 Visualización de la aplicación

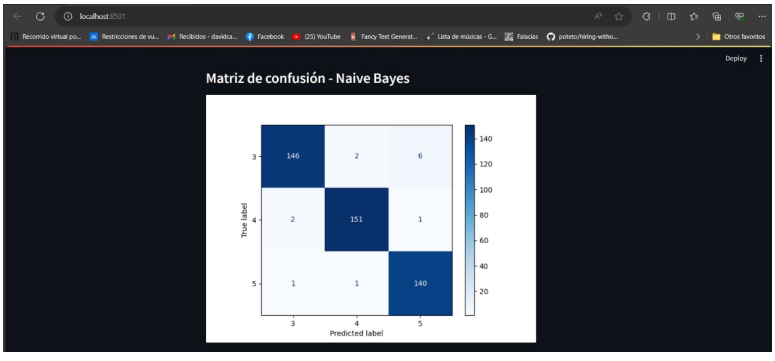

Figura 10 Visualización de la aplicación

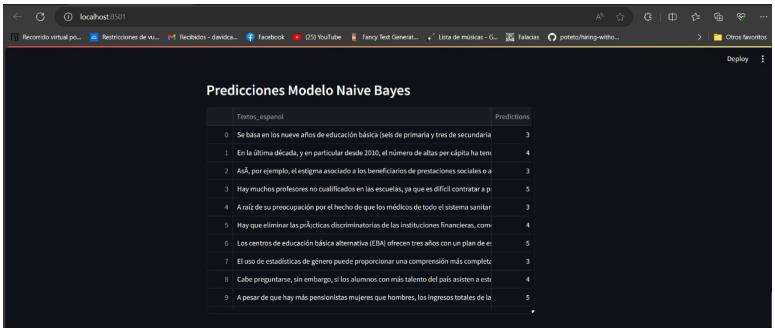

Figura 11 Visualización de la aplicación

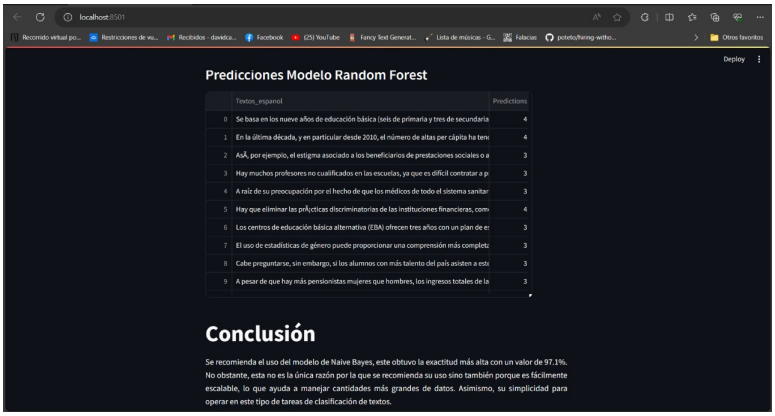

Figura 12 Visualización de la aplicación

1.4. Conclusiones

A partir de esto podemos concluir que con la aplicación realizada el analista de datos podrá observar los resultados que arrojó cada algoritmo y a partir de esto escoger alguno dependiendo el caso o problema que se le presente a la hora de clasificar textos. Encontramos que si se desea mejorar la clasificacion de la categoría 4 es mejor usar el algoritmo de regresión logistica multivariada, mientras que para la categoria 5 es mejor Naive Bayes. En este caso no es recomendable usar Random Forest ya que mostró un mayor margen de error de predicción en comparación con los otros algoritmos. La implementacion de Naive Bayes en particular podría brindar ventajas sustanciales a la UNFPA, puede automatizar la clasificación de opiniones de los ciudadanos sobre temas sociales y económicos. Este enfoque respalda a la organizacion de dos maneras:Permite identificar de manera rápida la categoría de ODS y reduce la necesidad de clasificar manualmente los comentarios

1.5. Trabajo en equipo

Tomás Acosta

Roles: Líder de proyecto, líder de datos
Tareas realizadas: Entendimiento de los datos, preparación de los datos, creación de modelo, elaboración del informe
Tiempo: 20 horas
Retos: Preparar datos para mantenerlo en lenguaje natural, aprender herramientas y conceptos nuevos para la preparación de los datos y obtención de palabras más útiles
Soluciones: Definición de funciones dentro de pipeline para remover caracteres que no sean ASCII, pasar todas la palabras a minúsculas, remover signos de puntuación, convertir caracteres especiales, implementar código para eliminar prefijos y sufijos, reemplazar números por palabras, realizar tokenización y lemmatización de los datos (normalización de los datos mediante el uso de lems y stems)
Puntaje: 33.3

Diego Rubio

Rol: Líder de negocio
Tareas realizadas: Entendimiento del negocio, entendimiento de los datos, preparación de los datos, documento/informe
Tiempo: 20 horas
Retos: Exploración de temáticas correspondientes para cada ODS, identificar el rol beneficiado y su conexión con la aplicación.
Soluciones: Investigar sobre el negocio a manera mas detallada y sobre los roles beneficiados por la aplicación.
Puntaje: 33.3

David Polo

Rol: Líder de analítica
Tareas realizadas: Implementación de la aplicación y wiki
Tiempo: 20 horas
Retos: Encontrar soluciones para implementar la aplicaicón y desplegarla
Soluciones: Usar librerias de python para poder crear la aplicación web
Puntaje: 33.3

1.6. Referencias

1. Scikit-learn developers. (2021). Scikit-learn: Machine Learning in Python. Recuperado de: `https://scikit-learn.org/stable/index.html`
2. SciELO. (2014). Clasificación automática de textos usando redes de palabras. Recuperado de: `https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-09342014000300001`.