PROYECTO ETAPA 2

AUTOMATIZACIÓN Y USO DE MODELOS DE ANALÍTICA DE TEXTOS

Elaborado por:

Andres Felipe Guerrero Sarmiento – 202015143
Julian Camilo Rivera — 202013338
Ricardo Andrés Sanchez Alvarez — 202014809

Institución:
Universidad De Los Andes

Curso:
Inteligencia de Negocios

Profesor:
Haydemar Núñez

TABLA DE CONTENIDO

- Descripción
- Objetivos
- Proceso de automatización del proceso de preparación de datos, construcción del modelo, persistencia del modelo y acceso por medio de API
- Desarrollo de la aplicación y justificación.
- Resultados
- Trabajo en equipo
- Encuentros

Descripción

Esta etapa se centra en el rol de ingeniero de datos. En ella,se automatizó el desarrollo del modelo de analítica de textos creado en la etapa 1 del proyecto y se desarrolló una aplicación orientada a un usuario donde, el uso del modelo tenga sentido y apoye alguna acción o decisión de dicho usuario. se interactuó con estudiantes del curso de estadística quienes jugaron el rol de usuarios del modelo analítico y de la aplicación. Con ellos trabajamos para identificar la aplicación desarrollada, el tipo de usuario que la usará, la forma de presentar los resultados obtenidos con el modelo de analítica construido en la etapa 1, en especial las métricas de desempeño del modelo y finalmente, se validó la aplicación desde el punto de vista de facilidad de uso y utilidad para el usuario para la cual se definió. Este trabajo de forma transdisciplinar pretende fortalecer las competencias de argumentación y mejorar la calidad de nuestros trabajos, gracias al análisis desde diferentes puntos de vista.

Objetivos

• Automatizar un proceso replicable para aplicar la metodología de analítica de textos en la construcción de modelos analíticos.
• Desarrollar una aplicación que utilice un modelo analítico basado en aprendizaje automático y sea de interés para una organización, empresa o institución y en particular para un rol existente en alguna de ellas.
• Interactuar con un grupo interdisciplinario para validar y mejorar la calidad de la solución analítica planteada y del producto de software construido.

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

Para el desarrollo de la pipeline se realizó lo siguiente en el Google Collaboratory:

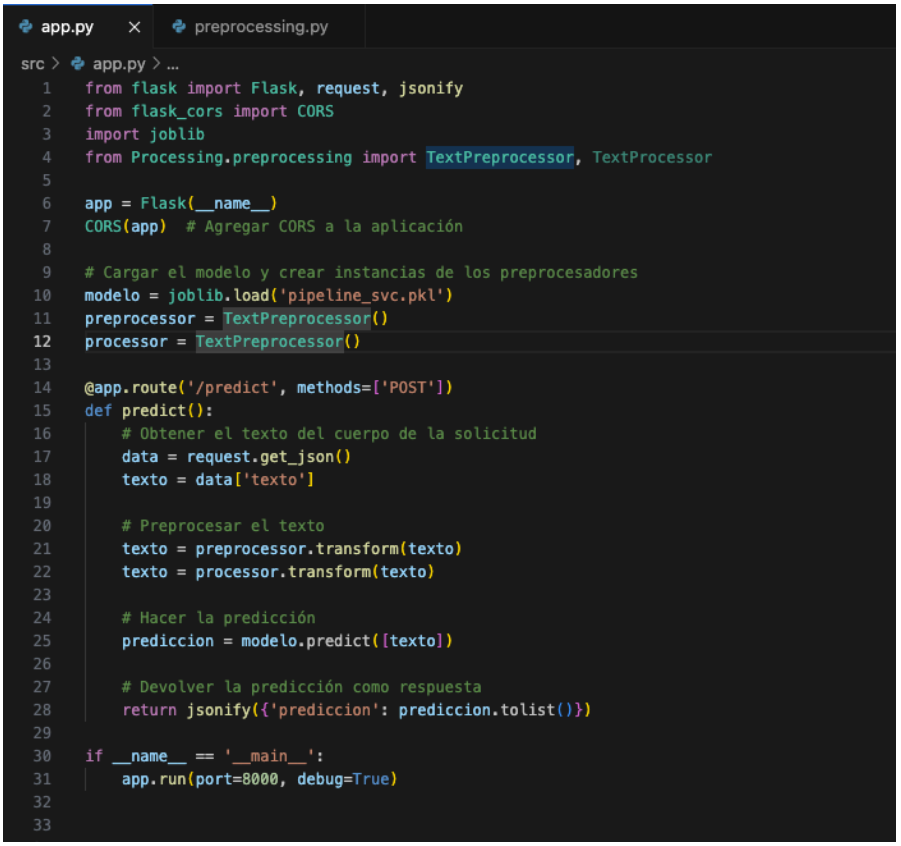
```
[60] joblib.dump(grid_search_svc, 'pipeline_svc.pkl')

     ['pipeline_svc.pkl']
```

Lo que nos permitió guardar el modelo entrenado en un archivo .pkl para usarlo luego en el back de la aplicación.

Para el back se utilizó un ambiente virtual y se definió el preprocesamiento de los datos en este.

```
src > Processing > preprocessing.py > TextProcessor
import pandas as pd
import numpy as np
import sys
import joblib
import inflect

import re, string, unicodedata
import spacy
import es_core_news_lg
from sklearn.linear_model import LinearRegression
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from sklearn.preprocessing import FunctionTransformer

from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer, HashingVectorizer
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier

from sklearn.base import BaseEstimator, ClassifierMixin, TransformerMixin

class TextPreprocessor(BaseEstimator, TransformerMixin):
    def __init__(self, language='es'):
        self.language = language
        self.nlp = es_core_news_lg.load()
        self.stemmer = SnowballStemmer('spanish')

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

    def transform(self, X):
        X_processed = self._preprocess_text(X)
        return X_processed

    def _preprocess_text(self, text):
        # Aplicar pasos de preprocesamiento
        text = self._remove_non_ascii(text)
        text = self._to_lowercase(text)
        text = self._replace_numbers(text)
        text = self._remove_punctuation(text)
        if self.language == 'es':
            text = self._remove_stopwords(text)
        return text

    def _remove_non_ascii(self, text):
        return unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8', 'ignore')

    def _to_lowercase(self, text):
        return text.lower()

    def _replace_numbers(self, text):
        p = inflect.engine()
        words = text.split()
        new_words = []
        for word in words:
            if word.isdigit():
```

Y se utilizó flask para crear la api que habilita el uso del pipeline del modelo



```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import joblib
from Processing.preprocessing import TextPreprocessor, TextProcessor

app = Flask(__name__)
CORS(app)  # Agregar CORS a la aplicación

# Cargar el modelo y crear instancias de los preprocesadores
modelo = joblib.load('pipeline_svc.pkl')
preprocessor = TextPreprocessor()
processor = TextPreprocessor()

@app.route('/predict', methods=['POST'])
def predict():
    # Obtener el texto del cuerpo de la solicitud
    data = request.get_json()
    texto = data['texto']

    # Preprocesar el texto
    texto = preprocessor.transform(texto)
    texto = processor.transform(texto)

    # Hacer la predicción
    prediccion = modelo.predict([texto])

    # Devolver la predicción como respuesta
    return jsonify({'prediccion': prediccion.tolist()})

if __name__ == '__main__':
    app.run(port=8000, debug=True)
```

Esto lo que nos permite es recibir un json del front, de este extraer el string del comentario y que automáticamente se le haga el preprocesado para póstumo a ello que el modelo realice la predicción sobre el texto.

Luego para el front de la aplicación se utilizó react con typescript:
Primero se generó una animación de carga para la aplicación:

```css
/* Estilos para el contenedor principal */
.init-loading-screen {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh; /* Ajusta según sea necesario */
}


/* Estilos para el contenido generado por el archivo CSS */
.edificio {
  width: 300px;
  height: 400px;
  background-color: beige; /* Establece el color de fondo como beige */
  animation: building 2s cubic-bezier(0.11, 0, 0.5, 0) 0s 1 normal forwards;
}

.cuadrado-pequeno {
  position: absolute;
}

@keyframes building {
    0% {
        animation-timing-function: ease-in;
        opacity: 1;
        transform: translateX(-48px);
    }

    24% {
        opacity: 1;
    }

    40% {
        animation-timing-function: ease-in;
        transform: translateX(-26px);
    }

    65% {
        animation-timing-function: ease-in;
        transform: translateX(-13px);
    }

    82% {
        animation-timing-function: ease-in;
        transform: translateX(-6.5px);
    }

    93% {
        animation-timing-function: ease-in;
        transform: translateX(-4px);
    }
```

Que se utilizó en una página de carga:

```tsx
src > pages > init_load > init_load.tsx > InitLoadingScreen
import React, {useState, useEffect}  from 'react';
import { useNavigate } from 'react-router-dom';
import '../../building.css'
import {CuadradoProps} from '../../interfaces'

const CuadradoPequeno: React.FC<CuadradoProps> = ({ left, top, width, height, backgroundColor }) => {
  return <div className="cuadrado-pequeno" style={{ left, top, width, height, backgroundColor}}></div>;
};


const InitLoadingScreen: React.FC = () => {

  const history = useNavigate();
  const [showContent, setShowcontent] = useState(true);

  useEffect(()=>{
    const timer = setTimeout(()=>{
      setShowcontent(false);
      history('/home');
    },2000);

    return()=> clearTimeout(timer);

  },[history]);

    return (
      <div className="init-loading-screen">
        <div className="edificio">
          <CuadradoPequeno top='50%' left='60%' width='90px' height='70px' backgroundColor='rgb(174, 214, 241)'/>
          <CuadradoPequeno top='50%' left='10%' width='90px' height='70px' backgroundColor='rgb(174, 214, 241)'/>
          <CuadradoPequeno top='20%' left='60%' width='90px' height='70px' backgroundColor='rgb(174, 214, 241)'/>
          <CuadradoPequeno top='20%' left='10%' width='90px' height='70px' backgroundColor='rgb(174, 214, 241)'/>
          <CuadradoPequeno top='82%' left='35%' width='90px' height='70px' backgroundColor='rgb(135, 54, 0 )'/>
        </div>
      <h1 className="title">ReviewAndes</h1>
      </div>
    );
}
export default InitLoadingScreen;
```

Para la página home o principal se utilizó la librería de MaterialUI para crear un cuadro de texto:

src > component > home > textField.tsx > FullWidthTextField

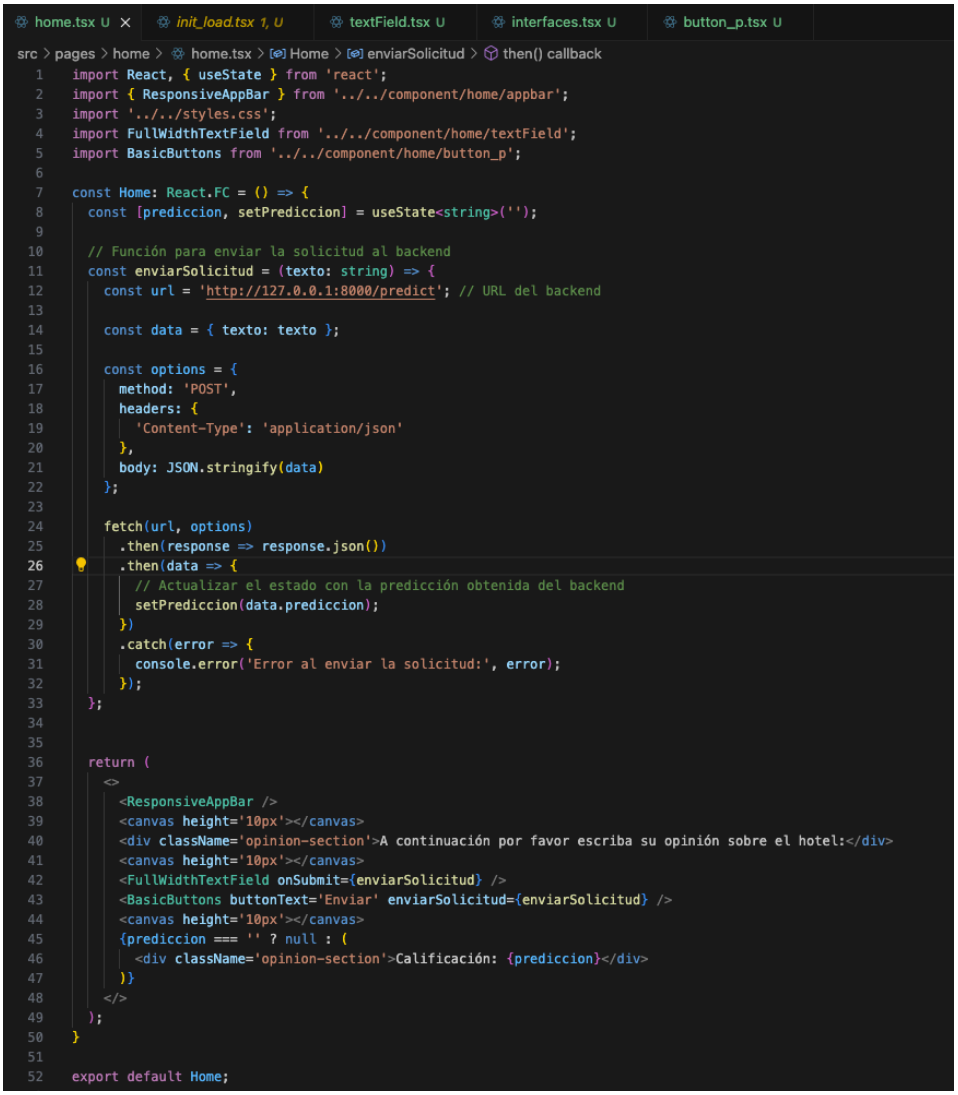
```tsx
// En FullWidthTextField.js
import React, { useState } from 'react';
import Box from '@mui/material/Box';
import TextField from '@mui/material/TextField';

export default function FullWidthTextField({ onSubmit }:any) {
  const [texto, setTexto] = useState('');

  const handleChange = (event:any) => {
    setTexto(event.target.value);
  };

  const handleKeyPress = (event:any) => {
    if (event.key === 'Enter') {
      onSubmit(texto);
    }
  };

  return (
    <Box
      sx={{
        width: 700,
        maxWidth: '100%',
        marginLeft: '20px',
        '& .MuiTextField-root': { m: 1, width: '700' },
      }}
    >
      <TextField
        fullWidth
        multiline
        label="Tu Review"
        id="fullWidth"
        value={texto}
        onChange={handleChange}
        onKeyPress={handleKeyPress}
      />
    </Box>
  );
}

```

Que permite mandar la información al darle enter y cada vez que registra un cambio en lo que se escribe lo registra en el campo text que es el que se envía a al back para hacer la predicción.
Luego está el botón:

src > component > home > button_p.tsx > BasicButtons

```tsx
import * as React from 'react';
import Stack from '@mui/material/Stack';
import Button from '@mui/material/Button';
import Box from '@mui/material/Box';
import {theme} from './appbar'; // Importa el tema desde appbar.js
import { ThemeProvider } from '@mui/material/styles';
import { BasicButtonProps } from '../../interfaces';

export default function BasicButtons({buttonText,enviarSolicitud}:BasicButtonProps) {
  const handleClick = () => {
    // Capturar el texto del campo de texto y enviarlo al backend
    const texto = document.getElementById('fullWidth')?.textContent || '';
    enviarSolicitud(texto);
  };

  return (
    <ThemeProvider theme={theme}>
      <Box sx={{ marginLeft: '25px', maxWidth: '100%', width: 100 }}>
        <Stack spacing={2} direction="row">
          <Button variant="contained" onClick={handleClick}>{buttonText}</Button>
        </Stack>
      </Box>
    </ThemeProvider>
  );
}

```

Que es el encargado de hacer el trigger del envío de la solicitud del texto que pertenece al cuadro de texto anterior, esto se puede ver en la función flecha de handleClick().
Luego tenemos la página de home:



En esta página definimos la función que se encarga de enviar el texto a la url adecuada en formato json y con el método adecuado, que en este caso es post. Luego usamos fetch para recibir la respuesta que viene del back y guardarla en una variable. Y por último renderizamos los componentes mencionados a continuación, y luego si se recibe la respuesta del back se muestra en la página.

Desarrollo de la aplicación y justificación.

- **Descripción del Usuario/Rol de la Organización:**

El usuario principal de esta aplicación sería el personal relacionado con la gestión y promoción turística, incluyendo representantes del Ministerio de Comercio, Industria y Turismo de Colombia, miembros de la Asociación Hotelera y Turística de Colombia (COTELCO), y personal de diversas cadenas hoteleras como Hilton, Hoteles Estelar y Holiday Inn, así como propietarios de hoteles más pequeños en diferentes municipios de Colombia. Estos usuarios están involucrados en la toma de decisiones estratégicas para promover destinos turísticos y mejorar la experiencia del turista.

- **Conexión con el Proceso de Negocio:**

La aplicación desarrollada se integra directamente con el proceso de toma de decisiones relacionadas con la gestión turística y la promoción de destinos. Al automatizar el proceso de análisis de reseñas turísticas, la aplicación permite a los usuarios obtener de manera rápida y eficiente una evaluación de la opinión asociada con cada reseña. Esto les proporciona información valiosa sobre la percepción de los turistas respecto a los destinos turísticos específicos y ayuda en la identificación de áreas de mejora y oportunidades para aumentar la popularidad y atractivo de los destinos.

- **Importancia para el Rol/Usuario:**

Para los usuarios mencionados anteriormente, la existencia de esta aplicación es de vital importancia. Les permite tomar decisiones informadas basadas en datos concretos sobre la percepción de los turistas, en lugar de depender únicamente de intuiciones o suposiciones. Al tener acceso instantáneo a la calificación asignada a cada reseña, los usuarios pueden priorizar acciones y asignar recursos de manera más efectiva para mejorar la calidad de los destinos turísticos y, en última instancia, aumentar la afluencia de turistas. Además, al facilitar el análisis de grandes volúmenes de reseñas de manera automatizada, la aplicación ahorra tiempo y recursos, permitiendo a los usuarios concentrarse en la implementación de estrategias específicas para mejorar la experiencia turística.

La aplicación desarrollada no solo agiliza el proceso de análisis de reseñas turísticas, sino que también mejora la capacidad de los usuarios para tomar decisiones estratégicas que impactan directamente en la industria turística de Colombia. Lo que finalmente conlleva a una optimización de recursos y en la promoción de destinos turísticos de manera más efectiva y centrada en el cliente.

Resultados

Las dos acciones específicas que el usuario puede realizar como resultado de su interacción con la aplicación son:

1. Identificación de los mejores sitios turísticos:

Los usuarios pueden utilizar la aplicación para tener una idea sobre la opinión sobre diferentes sitios turísticos de parte de otras personas. Por ejemplo, si un usuario prefiere destinos tranquilos y naturales con buenas calificaciones, a través de la aplicación puede conocer o obtener información sobre sitios turísticos que se acoplen a sus gustos.

2. Definición de estrategias de mejora:

Basado en los resultados obtenidos del modelo analítico, los usuarios pueden desarrollar estrategias específicas para mejorar la calidad y la popularidad de los distintos sitios turísticos basados en las reseñas dadas. Por ejemplo, si descubren que un destino particular recibe críticas con valoraciones negativas consistentes sobre la falta de higiene, pueden implementar estrategias con el objetivo de mejorar la higiene del sitio.

Trabajo en equipo

Estudiante	Rol	Tareas	Tiempo	Retos	Soluciones
Julian Rivera	Líder de proyecto | Ingeniero de software responsable de desarrollar la aplicación final	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. Tambien se encarga de gestionar el proceso de construcción de la aplicación.	7 horas	La implementacion e integracion de el modelo predictivo con el aplicativo realizado	busqueda de informacion en distintos lugares y recibir apoyo de los integrantes del grupo
Andres Guerrero	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.	7 horas	el analisis y el entendimiento del problema	la discusión en las clases con los profesores y la resolucion de dudas con monitores
Ricardo Sánchez	Ingeniero de datos	Es responsable de velar por la calidad del proceso de automatización relacionado con la construcción del modelo analítico.	7 horas	Hacer el estudio de calidad de los datos, buscar y entender las tecnologias necesarias para el correcto uso de la informacion	utilizar los recursos brindados en clase IA e informacion en linea para realizar un correcto analisis de datos

Encuentros

Reunion	
Fecha	Descripción de la reunión
8/04/2024	Reunión de lanzamiento y planeación, Para definir roles y forma de trabajo del grupo. Se genera lluvia de ideas sobre la forma de resolver el proyecto. Se realizo la reunion con el grupo de estadística que nos fue asignado.
15/04/2024	Reuniones de seguimiento se realizo para ver si existian avances de parte de cada uno del proyecto sobre todo lo que se habia definido
20/04/2024	Reunión de finalización: Para consolidar el trabajo final, verificamos el trabajo del grupo y analizamos el funcionamiento de la aplicación. interactuamos con el grupo de estadística para validar que se lograron los objetivos.

evidencia reunión 20 de abril: