Luis Felipe Torres – 202123815

David Pérez Cárdenas – 202123314

Las instrucciones para correrlo se encuentran en el readme

Proyecto 1 Etapa 2

Inteligencia de Negocios

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

Antes de todo realizamos la creación del joblib/pipeline:

Esto como tal se encuentra en “proyecto2.ipynb”. El primer elemento necesario para poder hacer esto era tener los elementos de la etapa pasada, tanto el modelo (el escogido en la etapa anterior) como también tener la limpieza de datos.

Teniendo esto en cuenta, en el inicio de este archivo tenemos lo clase Limpieza que realiza todo el proceso de limpieza de datos (más adelante en el documento hablamos de ella así que por ahora la obviaré).

Luego tenemos la creación del pipeline como tal:

```
pipeline = make_pipeline(
    TfidfVectorizer(sublinear_tf=True, max_df=0.5, min_df=5, ngram_range=(1, 3)),
    LogisticRegression(max_iter=1000)
)
```

El modelo utilizado es el “LogisticRegression”, y su razón fue explicada en la etapa anterior, dado que nos dimos cuenta que este era el más adecuado.

Posteriormente tenemos la creación del joblib que es lo que necesitamos para poder hacer uso del pipeline utilizando el API:

```
from joblib import dump, load

dump(pipeline, 'reviewModel.joblib')
```

Ahora viene la creación del API. En primera instancia lo que realizamos fue generar un archivo “limpieza_module.py” el cual contiene la clase Limpieza:

```python
class limpieza:

    def solo_letras(self, texto):
        abc = set(list("abcdefghijklmnñopqrstuvwxyz"))
        new_texto = ""
        for i in range(len(texto)):
            if texto[i] not in abc:
                new_texto += " "
            else:
                new_texto += texto[i]
        return new_texto

    def texto(self):
        self.data = self.data.lower()
        self.data = self.data.replace("á", "a").replace("é", "e").replace("í", "i").replace("ó", "o").replace("ú", "u")
        self.data = self.solo_letras(self.data)
        return self.data
```

Esta clase es la encargada de realizar todo el proceso de preparación de datos, en este caso específico está hecha como una única clase, con el propósito de automatizar este trabajo, debido a que en la clase principal (que mencionaremos más adelante), al momento de la entrada de datos, se llama esta clase para realizar su limpieza. Algunas de las cosas que realiza son: Eliminar todos los caracteres que no sean letras, eliminar tildes, eliminar las palabras vacías y reescribir las palabras en sus raíces. Todo esto con el propósito de que nuestro modelo sea más correcto.

Luego tenemos la otra clase “transformData.py”:

```python
data = pd.read_csv('tipo2_entrenamiento_estudiantes.csv')

data["Review"] = data["Review"].apply(limpieza().limpiar)

data_train, data_test = train_test_split(data, test_size=0.2)
x_train, x_test = data_train["Review"], data_test["Review"]
y_train, y_test = data_train["Class"], data_test["Class"]
```

Esta clase es la encargada de definir nuestros valores de entreno y testeo de datos, esto lo realizamos en una clase aparte, para que todo este proceso ocurra al momento de montar el API y no cada vez que se realice una petición.

Finalmente tenemos la clase “main.py”:

```python
@app.get("/")
def read_root():
    return {"status": "Running"}

@app.post("/predict")
def make_predictions(dataModel: DataModel):
    df = pd.DataFrame(dataModel.dict(), columns=dataModel.dict().keys(), index=[0])
    df.columns = dataModel.columns()

    df["review"] = df["review"].apply(limpieza().limpiar)

    result = model.predict(df["review"]).tolist()[0]
    prob = model.predict_proba(df["review"]).tolist()[0]

    return {"result": result, "prob": prob}
```

Esta clase es lo más fundamental del API teniendo en cuenta que aquí es donde se construye como tal.

Inicialmente realizamos unos elementos fundamentales como cargar el joblib, definir la aplicación y retornar el estado para la correcta ejecución.

El primer elemento importante es el “post(“/predict”)”, este es el elemento que nos permite realizar la predicción de cuál será la clase (la calificación) y la probabilidad de dicha reseña/texto. Esto utiliza el joblib que creamos previamente (que ya mencionamos como funciona) y con esto logra predecir esta información para posteriormente enviarla mediante un JSON para su vista.

La otra función primordial es:

```python
@app.get("/metrics")
def get_metrics():
    y_pred = model.predict(x_test)

    report = classification_report(y_test, y_pred, target_names=model.named_steps['logisticregression'].classes_, output_dict=True)
    weighted = report.pop("weighted avg")
    del weighted["support"]
    keys = list(report.keys())
    for i in keys:
        try:
            del report[i]["support"]
            report[str(i)] = report[i]
            del report[i]
        except:
            continue
    report["weighted avg"] = weighted

    words = {}

    feature_names = model.named_steps['tfidfvectorizer'].get_feature_names_out()
    coefficients = model.named_steps['logisticregression'].coef_
    for i, class_label in enumerate(model.named_steps['logisticregression'].classes_):
        top10 = np.argsort(coefficients[i])[-10:][::-1]
        words[str(class_label)] = [feature_names[j] for j in top10]

    return {"report": report, "words": words}
```

La función “get("/metrics")” nos permite encontrar aquellas palabras más relevantes para cada clasificación, así como también unas métricas específicas para cada clase. En este orden de ideas, con esto podemos encontrar aquellas características que le interesan al negocio, así como también podemos determinar la calidad del modelo.

Cabe aclarar que cuando mostramos las palabras, mostramos las mejores 5 de cada clase. Aquellas que mejor se acomodan a dicha calificación.

2) Desarrollo de la aplicación y justificación

El usuario que más se beneficiaría de nuestra página web serían aquellos que trabajen dentro de aquellas empresas hoteleras que tengan el rol de mejorar la experiencia/el establecimiento en sí. Esto es gracias a que nuestra página web permite visualizar aquellas características relevantes que presenta una reseña, así como también, tendrá la posibilidad de estimar que elementos tienen que importancia para el usuario, esto mediante la generación de promts que podría escribir un usuario, y pasarlos mediante el método (dentro de la página) de predicción, para determinar que tanta calificación tendría de forma aproximada.

También cabe destacar que la relación entre esta aplicación y el negocio es muy estrecha, ya que esto cumpliría ambas necesidades del negocio, por una parte, la capacidad de predecir la clasificación de una reseña y, por otro lado, definir características que determinan cierta calificación.

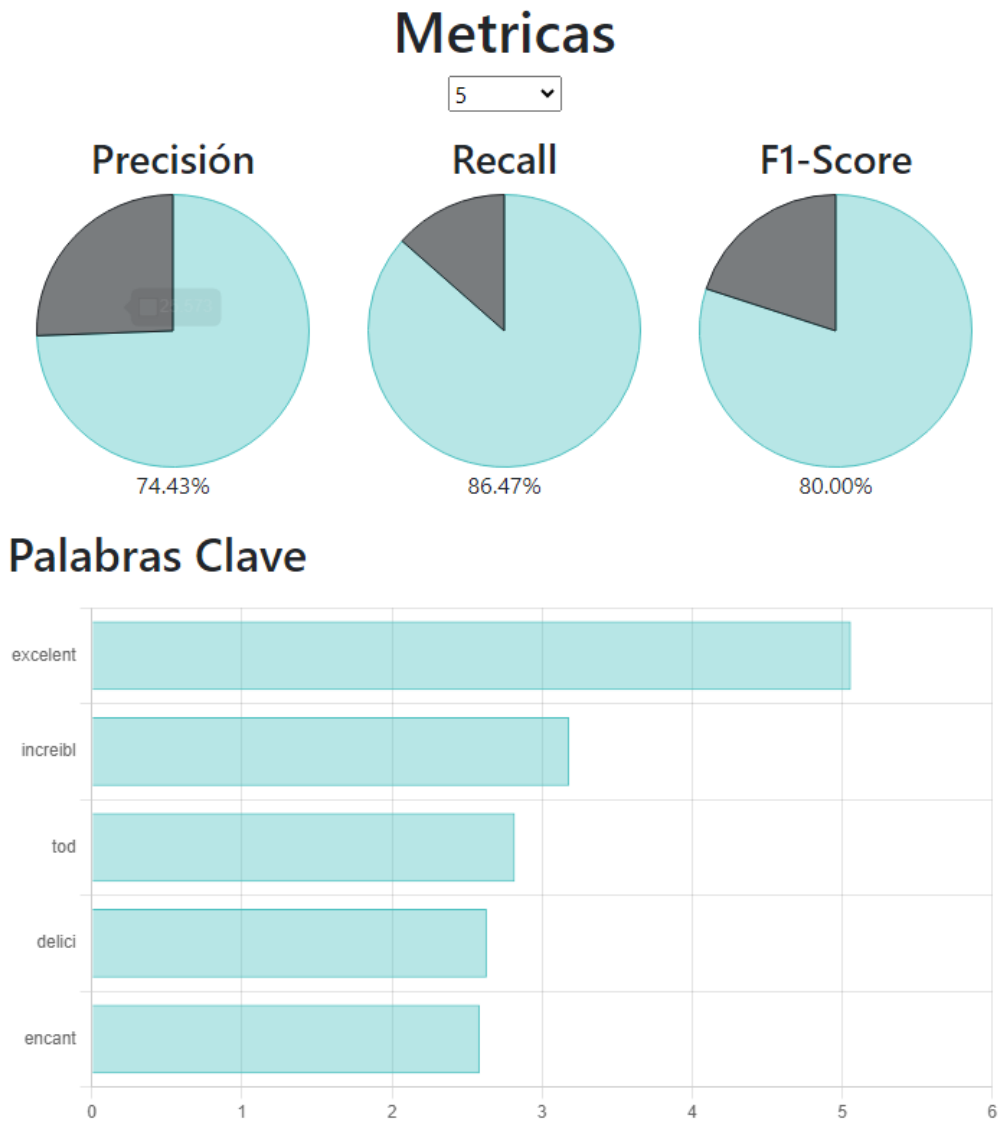
Ahora hablaremos sobre la aplicación/página web como tal:

Esta es la pestaña de inicio donde pueden ver nuestros nombres y navegar a las diferentes pestañas:

En la pestaña de predecir, el usuario puede escribir aquellas palabras/reseñas que quiere predecir su clasificación y ver dicho número:

Si desea hacer cambios a tiempo real y ver como influye, es necesario activar la opción de “predicción en tiempo real”.

En la pestaña de métricas, el usuario puede navegar por las distintas clases para poder ver las métricas de cada una, y las palabras más influyentes.



3) Resultados

El video se encuentra en el padlet.

4) Trabajo en equipo

Este trabajo realizado fue labor de Luis F. Torres y David Pérez. Ambos integrantes trabajaron (cada uno) alrededor de 8-10 horas en el proyecto, aunque hayan tenido roles diferentes, hubo muchas instancias donde el trabajo en equipo era fundamental, por esta razón cada integrante debería de tener 50 puntos (que suman 100 teniendo en cuenta que fueron 2 integrantes).

El estudiante Luis F. Torres tenía el rol de líder de modelado y líder de análisis, este estudiante se encargaba del desarrollo principal del API, así como también de gran parte de la elaboración del front de la aplicación.

El estudiante David Pérez tenía el rol de líder de planeación y líder de negocio, este estudiante se encargaba de realizar labores de complemento del código estructuralmente, desarrollo del pipeline y construcción del documento.

Esta etapa del proyecto constó principalmente de 3 reuniones, en la primer hubo un entendimiento de que era el paso a seguir y se realizaron las primeras labores de creación del API y lo suficiente de front como para poder realizar la siguiente reunión. Posteriormente nos reunimos con los compañeros de estadística para poder escuchar sus insumos y recomendaciones de como seguir, y finalmente hubo una reunión donde se finalizó el front, se realizó el video y se terminó el documento.

Aquellos elementos que se resaltaron para mejorar son: Realizar una comunicación más efectiva; esto no implica que haya habido mala comunicación, sino que debería priorizarse su efectividad, y también se encontró que otro aspecto a mejorar es la definición de prioridades; no fue del todo correcta la decisión de prioridades en el proyecto, eso es un aspecto para mejorar.