Avances Comparación entre artículos de Mercado Libre por medio del análisis, clasificación y ranqueo de comentarios

Jobson Agustín Facundo

Universidad Católica de Santiago del Estero, Departamento Académico Rafaela

El paso siguiente en lo que respecta a avances en esta iniciativa, se centra en su totalidad en la codificación y desarrollo de una web app que permita a los usuarios ingresar las URL's de los productos que desean comparar y el sistema realice dicha comparación, retornando de esta manera el producto mejor valorado por los comentarios.

Lenguaje y Framework de Desarrollo

Antes de comenzar a desarrollar la aplicación web, se debe tener muy en claro en que lenguaje va a ser desarrollado, y si se utilizará algún tipo de framework de desarrollo. Es claro que cuando se trata de machine learning, procesamiento de lenguaje natural o sentiment analysis, el lenguaje por excelencia es Python. Es por este motivo que se decidió realizar el desarrollo completo del sistema utilizando este lenguaje, puesto que todas las herramientas que facilitan el trabajo de recopilación de datos, análisis y demás son desarrolladas en base a este lenguaje. Una vez que se decidió el lenguaje, el siguiente paso es pensar que framework de desarrollo es el más adecuando teniendo en cuenta el alcance establecido. En este sentido, existen dos opciones, **Django o Flask**.

Django

Es lo que se conoce como un framework **"full stack"** para Python con el cual se pueden abordar todo tipo de proyectos en este lenguaje como el desarrollo web escalable y de alta calidad.

Se trata del entorno de desarrollo de Python más popular, ya que permite un desarrollo ágil al encargarse de manejar los controladores.

Fue diseñado para trabajar bajo un modelo MVT (modelo, vista, plantilla) porque el framework se encarga del manejo de controladores, razón por la cual se dice que es reusable y permite el desarrollo ágil.

Flask

Es un framework que permite desarrollar aplicaciones web de forma sencilla, está especialmente guiado para desarrollo web fácil y rápido con Python. Una de sus características a destacar es el potencial de instalar extensiones o complementos acorde al tipo de proyecto que se desarrolla. Con Flask se utilizan las líneas de código necesarias para realizar cualquier acción por lo que es mucho más sencillo comprender la estructura de cualquier aplicación o script, y saber qué es lo que realiza.

Diferencias entre ambos Frameworks

- Django cuenta con un sistema de autenticación de usuarios, mientras que Flask puede usar las opciones nativas para crear un sistema de autenticación básico
- Ambos frameworks permiten un backend limpio en manejo de usuarios, pudiendo limitar memoria y velocidad para evitar caídas en el servidor.

- Las API’s que genera Django son mejores
- La velocidad de Flask es un gran punto a su favor. El desempeño en velocidad que tiene es superior gracias a su diseño minimalista en su estructura.

En conclusión, los proyectos que se desarrollan empleando el framework Django, generalmente son proyectos más grandes y robustos, pensados para ser escalables y con una gran cantidad de funcionalidades en el mismo. En cambio, Flask, es un framework pensado para el rápido despliegue de aplicaciones sencillas y no tan grandes, priorizando la velocidad debido a que, a priori, se trata de aplicaciones minimalistas sin excesivas funcionalidades.

Es por este motivo, y teniendo en cuenta el alcance de esta iniciativa, que se decidió desarrollar la aplicación utilizando el framework Flask, ya que no se trata de una aplicación pensada para ser muy grande y además se trata de una aplicación simple (es solo una aplicación con pocos templates entre los cuales resaltan un template con un formulario donde el usuario ingresará los links de los productos que desea comparar, y un template de resultados), por lo que muchas de las funcionalidades que provee Django y que si serían útiles para proyectos un poco más complejos (como el sistema de autenticación de usuarios y demás), en este proyecto no serían de utilidad y contribuirían incluso a reducir el rendimiento del sitio.

Una vez determinados estos aspectos, es posible proseguir con el desarrollo de la aplicación per se.

Estructura de archivos de la Web App

Directorio Raíz de la web app

```
(venv-tesis) agu@agu-lenovo ~/r/t/app (develop)> ll
total 28K
-rw-rw-r-- 1 agu agu  239 nov 23 13:52 forms.py
drwxrwxr-x 2 agu agu 4,0K nov 25 16:09 __pycache__/
drwxrwxr-x 4 agu agu 4,0K nov 23 08:24 static/
drwxrwxr-x 2 agu agu 4,0K nov 25 11:22 templates/
-rw-rw-r-- 1 agu agu 4,2K nov 25 16:09 utils.py
-rw-rw-r-- 1 agu agu 1,5K nov 25 11:22 views.py
```

Imagen Nro 1: “Directorio raíz de la aplicación web”

En el directorio raíz de la app se pueden ver los siguientes archivos:

- Forms.py: en este archivo Python se encuentran todos los formularios necesarios para realizar el input de las URL’s de los productos a comparar.
- Utils.py: en este archivo Python se encuentran todas las funciones que son de utilidad para la obtención, preprocesado y demás de los comentarios de los productos.
- Views.py: se encuentran todas las funciones que redireccionan a los diferentes templetes de la web app
- Carpeta Static
- Carpeta Templates

Carpeta Templates

```
(venv-tesis) agu@agu-lenovo ~/r/t/app (develop)> ll templates/
total 16K
-rw-rw-r-- 1 agu agu 1,2K nov 25 11:22 base.html
-rw-rw-r-- 1 agu agu  457 nov 23 08:24 home.html
-rw-rw-r-- 1 agu agu 1,2K nov 23 13:58 products-compare.html
-rw-rw-r-- 1 agu agu  688 nov 25 11:22 results.html
```

Imagen Nro 2: "Contenido de la carpeta Templates"

En esta carpeta se encuentran todos los archivos HTML, que son los que se renderizarán cuando se accedan a determinadas URL's dentro de la web app.

Carpeta Static

```
(venv-tesis) agu@agu-lenovo ~/r/t/app (develop)> ll static/
total 8,0K
drwxrwxr-x 2 agu agu 4,0K nov 23 08:24 css/
drwxrwxr-x 2 agu agu 4,0K nov 23 08:24 images/
(venv-tesis) agu@agu-lenovo ~/r/t/app (develop)> ll static/css/
total 12K
-rw-rw-r-- 1 agu agu  371 nov 23 08:24 base.css
-rw-rw-r-- 1 agu agu 1,6K nov 23 08:24 form.css
-rw-rw-r-- 1 agu agu  261 nov 23 08:24 home.css
(venv-tesis) agu@agu-lenovo ~/r/t/app (develop)> ll static/images/
total 56K
-rw-rw-r-- 1 agu agu 32K nov 23 08:24 mercado-libre-logo.png
-rw-rw-r-- 1 agu agu 24K nov 23 08:24 star.png
```

Imagen Nro 3: "Contenido de la carpeta static"

Esta carpeta hace referencia a todos aquellos archivos que son estáticos, como lo son las imágenes o los archivos de estilos (css). Por lo tanto, se pueden encontrar dos carpetas distintas, la carpeta **css** que es la que contiene todos los archivos que darán estilo a los HTML de la carpeta **templates**; y también se puede encontrar la carpeta **images**, que es la que contiene las imágenes que aparecen en el template base.html.

Contenido de los archivos

Forms.py

```python
from flask_wtf import FlaskForm
from wtforms import StringField
from wtforms.validators import DataRequired

class getForm(FlaskForm):
    url1 = StringField(validators=[DataRequired()])
    url2 = StringField(validators=[DataRequired()])
```

Imagen Nro 4: "Archivo forms.py"

Este archivo forms.py se utiliza para crear el formulario (que solo tiene dos inputs de tipo string) que se utilizará para ingresar las URL's de los productos a comparar. En un principio se importan todos los elementos a utilizar, como el FlaskForm que se trata de

un formulario provisto por Flask, o el StringField, que se utiliza para inputs de tipo string y luego se crea la clase getForm con dichos elementos.

Utils.py

```python
from unicodedata import normalize
from nltk.corpus import stopwords

import re
import pandas as pd
import requests
import json
import nltk


#nltk.data.path.append("/home/agu/Desktop/tesis/utils/stopwords") # Ubuntu Virtualbox
nltk.data.path.append("/home/agu/repos/tesis/utils/stopwords") # Notebook
stop = stopwords.words('spanish')

def obtener_id_from_url(url):
    """Funcion para obtener el ID de productos a través de su URL"""
    id_prod = re.search(r'/p/(.+?)\?', url)
    if id_prod:
        return id_prod.group(1)
    url_split = url.split("/p/")
    if len(url_split) > 1:
        return url_split[1]
    return False


def obtener_info_producto(id_prod):
    """Funcion para llamar a la API de ML y obtener la informacion de un producto (nombre, imagenes, etc.)"""
    url_api = "https://api.mercadolibre.com/products/" + id_prod
    response = requests.get(url_api)
    if response.status_code != 200:
        return ""
    response_json = json.loads(response.text)
    return response_json


def convert_response_to_list(reviews):
    """Funcion para retornar algunas cosas de los comentarios"""
    list_reviews = []
    for review in reviews:
        new_json = {
            "titulo": review['title'].lower(),
            "comentario": review['content'].lower(),
            "valoracion": review['rate']
        }
        list_reviews.append(new_json)
    return list_reviews
```

Imagen Nro 5: "Inicio del archivo utils.py"

Realizando un breve repaso de lo que fue el avance anterior, se pueden mencionar dentro de las herramientas a utilizar, **requests**, que se trata de una librería basada en Python que facilita realizar peticiones HTTP; **nltk**, que es un conjunto de bibliotecas y herramientas para el procesamiento del lenguaje natural para el lenguaje de programación Python y posee un listado de "stopwords" en el idioma español; también podemos encontrar **pandas,** que es una herramienta de manipulación y análisis de datos de código abierto construida sobre el lenguaje de programación Python; entre otras herramientas que ayudan al pre-procesado de los datos.

A modo de resumen, a continuación, se detallan los métodos que se encuentran en este archivo y una breve descripción de los mismos:

- El método **"obtener_id_from_url"** lo que hace es tomar el URL pasado como parámetro y se encarga de extraer el ID del producto a evaluar.
- El método **"obtener_info_producto"** llama a una API (utilizando el id del producto pasado como parámetro) que brinda Mercado Libre (diferente a la que utilizo para obtener los comentarios de los productos), la cual utilizo para obtener todos los datos relacionados al producto y que muestro en la pantalla de resultados

(como el nombre completo del mismo, precio, familia, link para la compra y las imágenes del mismo). Aquí se hace uso de la herramienta requests.

- El método **"convert_response_to_list"** toma una lista de reviews, la cual tiene muchos datos que no serán útiles, y filtra por los datos que si lo son (como es el caso del título, el comentario per se y la valoración)

```
def obtener_df_reviews(id_prod):
    """Funcion para llamar a la API de ML y obtener los comentarios"""
    url_api = "https://api.mercadolibre.com/reviews/item/"
    url = url_api + id_prod
    args = {'limit': 200}

    response = requests.get(url, params=args)
    if response.status_code != 200:
        return ""
    response_json = json.loads(response.text)
    reviews = convert_response_to_list(response_json['reviews'])
    df_comentarios = pd.DataFrame.from_records(reviews)
    return df_comentarios
```

Imagen Nro 6: "Siguiente método del archivo utils.py"

- El método **"obtener_df_reviews"** llama a una API brindada por Mercado Libre la cual dado un determinado Id de producto, devuelve los comentarios correspondientes a dicho producto. Al igual que en una función anterior, se hace uso de la herramienta requests para realizar la petición HTTP. Además, se llama al método previamente mencionado **"convert_response_to_list"** para conservar sólo los datos útiles. En adición, se utiliza la herramienta **"Pandas"** (renombrada como pd) para convertir esa lista de objetos json en un objeto dataframe. Cabe aclarar que **Dataframe** es la estructura de datos fundamental de Pandas, representa una tabla de datos panel con indexación integrada. Cada columna contiene los valores de una variable y cada fila un conjunto de valores de cada columna. En este caso, cada fila hace referencia a un comentario, y se tendrían una serie de columnas que hacen referencia a ese comentario en particular, donde los datos que poseen dichas columnas son los que se especifican en el método **"convert_response_to_list"** (titulo, valoración y comentario).

```
palabras_indicadoras_comentario_positivo = "muy buen celular|el celular es excelente |muy bueno|todo bueno"
palabras_indicadoras_comentario_negativo = "malisimo|No recomandable|desastre|pobre|malisima|el teléfono es muy malo"

def analizar_comentarios_df(df):
    porc_comentarios_positivos = 0
    porc_comentarios_negativos = 0

    if not df.empty:

        df_neg = df[
            (df["comentario"].str.contains(palabras_indicadoras_comentario_negativo)) |
            (df["titulo"].str.contains(palabras_indicadoras_comentario_negativo))
        ]

        df_pos = df[
            (df["comentario"].str.contains(palabras_indicadoras_comentario_positivo)) |
            (df["titulo"].str.contains(palabras_indicadoras_comentario_positivo))
        ]

        porc_comentarios_positivos = (len(df_pos) / len(df)) * 100
        porc_comentarios_negativos = (len(df_neg) / len(df)) * 100

    return porc_comentarios_positivos, porc_comentarios_negativos

def obtener_mejor_producto(df1, df2):
    porc_com_pos_prod_1, porc_com_neg_prod_1 = analizar_comentarios_df(df1)
    porc_com_pos_prod_2, porc_com_neg_prod_2 = analizar_comentarios_df(df2)

    rdo_prod_1 = porc_com_pos_prod_1 - porc_com_neg_prod_1
    rdo_prod_2 = porc_com_pos_prod_2 - porc_com_neg_prod_2

    if rdo_prod_1 > rdo_prod_2:
        return 1
    return 2
```

Imagen Nro 7: "Últimos métodos del archivo utils.py"

- El método **“analizar_comentarios_df”** se trata de una primera aproximación base en el análisis de cada comentario, únicamente con fines ilustrativos con el objetivo de proseguir con el desarrollo del sistema. Recibe como parámetro un Dataframe y analiza cada comentario (según las palabras indicadas) con el objetivo de determinar que porcentaje de comentarios es positivo y que porcentaje es negativo. Este método es el que, en futuros avances, incorporará modelos y algoritmos de machine learning y sentiment análisis en pos de realizar un análisis mas preciso de cada comentario.
- Para finalizar, se desarrolló el método **“obtener_mejor_producto”,** el cual llama al método “analizar_comentarios_df” previamente desarrollado con ambos dataframes de datos y retorna el producto que dio mejores resultados.

Views.py

```python
from flask import Flask, Blueprint, render_template, session, redirect, url_for
from forms import getForm
from utils import obtener_info_producto, obtener_id_from_url, obtener_df_reviews, obtener_mejor_producto

app = Flask(__name__)
app.config['SECRET_KEY'] = '1234'

@app.route("/")
@app.route("/home")
def home():
    """vista de la home"""
    return render_template("home.html")


@app.route("/products-compare", methods=["GET","POST"])
def products_compare():
    """Vista del template de comparacion de productos"""
    form = getForm()
    if form.validate_on_submit():
        id_prod1 = obtener_id_from_url(form.url1.data)
        id_prod2 = obtener_id_from_url(form.url2.data)

        df_prod1 = obtener_df_reviews(id_prod1)
        df_prod2 = obtener_df_reviews(id_prod2)

        mejor_producto = obtener_mejor_producto(df_prod1, df_prod2)
        if mejor_producto == 1:
            return redirect(url_for("results", prod = id_prod1))
        return redirect(url_for("results", prod = id_prod2))
    return  render_template("products-compare.html", form=form)

@app.route("/results/<prod>")
def results(prod):
    """Vista del template de resultados"""
    info_prod = obtener_info_producto(prod)
    return render_template("results.html", prod = info_prod)


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

Imagen Nro 8: “Contenido del archivo views.py”

Al igual que el caso anterior, se comienzan importando todos los elementos, funciones y clases que se utilizarán, como la clase **getForm** creada anteriormente en el archivo forms.py o las funciones desarrolladas en el archivo utils.py.

Lo que se realiza en este archivo es el mapeo de cada URL de la aplicación web con sus correspondientes funciones y su correspondiente template. De esta manera, por ejemplo, cuando se entre a la url “/home” o simplemente “/” se renderizará el template “home.html”, cuando se ingrese a la url “/products-compare” se utiliza el formulario creado en forms.py, y si se trata de una petición GET, simplemente se renderiza el template “products-compare.html”, pero si, por el contrario, se trata de una petición POST (lo que quiere decir que el usuario ya ha ingresado las URL de los productos que desea

comparar y le ha dado al botón de comparar) lo que se hace es extraer el ID de los productos con el método desarrollado en utils.py y posteriormente se obtienen todos los comentarios de esos IDs, para finalmente utilizar la función “obtener_mejor_proucto” para saber cuál fue el mejor de los dos ingresados y redireccionar a la url “/results/<producto>” con el ID del mejor producto.

Para finalizar, cuando se ingresa a la URL “/results/<prod>” lo que se hace es tomar ese ID pasado como parámetro al método y utilizar el método importado de utils.py “obtener_info_producto” con dicho ID, con el objetivo de obtener todos los datos necesarios que serán mostrados al usuario. Y finalmente se renderiza “results.html” con la información obtenida de dicho método

Templates

A continuación, se expone el código correspondiente a algunos de los templates más importantes de la aplicación.

Products-compare.html

```
{% extends "base.html" %}

{% block content %}
    <link type="text/css" rel="stylesheet" href="{{url_for('static', filename='css/form.css')}}"/>
    <div style="margin-left:50px;">
        <h2>Ingrese las URL's de los productos que desea comparar</h2>
        <form id="compare_form" name="compareUrl" action="" method="post" enctype="multipart/form-data" novalidate>
            {{ form.hidden_tag() }}
            <div style="margin-top:4%">
                <h4>URL del Producto 1</h4>
                {{form.url1(class_="select-css")}}

                <h4>URL del Producto 2</h4>
                {{form.url2(class_="select-css")}}
            </div>
            <div style="margin-top:3%">
                <a
                    id="compareButton"
                    onclick="document.getElementById('compare_form').submit();"
                    class="button1">Comparar
                </a>
                <a
                    type="button"
                    class="button1"
                    href="/"
                    style="background-color:#CCCCCC; color:#000000">Volver
                </a>
            </div>
        </form>
    </div>
{% endblock %}
```

Imagen Nro 9: “Contenido del archivo products-compare.html”

Results.html

```
{% extends "base.html" %}

{% block content %}
    <div style="margin-left:50px;">
        <h1>MEJOR PRODUCTO:</h1>
        <center>
            <h3>{{ prod["name"] }}</h3>
            <p><b>Precio:</b> ${{ prod["buy_box_winner_price_range"]["min"]["price"] }} - ${{ prod["buy_box_winner_price_range"]["max"]["price"] }}</p>
            <p><b>Familia:</b> {{ prod["family_name"] }}</p>
            <p><b>Compra el producto:</b> <a href='{{ prod["permalink"] }}' target="_blank">{{ prod["permalink"] }}</a></p>
            {% for imagen in prod["pictures"] %}
                <img src='{{ imagen["url"] }}' height="60%" />
            {% endfor %}
        </center>
    </div>
{% endblock %}
```

Imagen Nro 10: “Contenido del archivo results.html”

Aplicación Final

Pagina Inicio (Home)

Imagen Nro 11: "Muestra de la página de inicio"

Algunos ejemplos de Comparación de productos y Resultados

Ejemplo 1: "Samsung Galaxy 01 VS Samsung Galaxy A32"

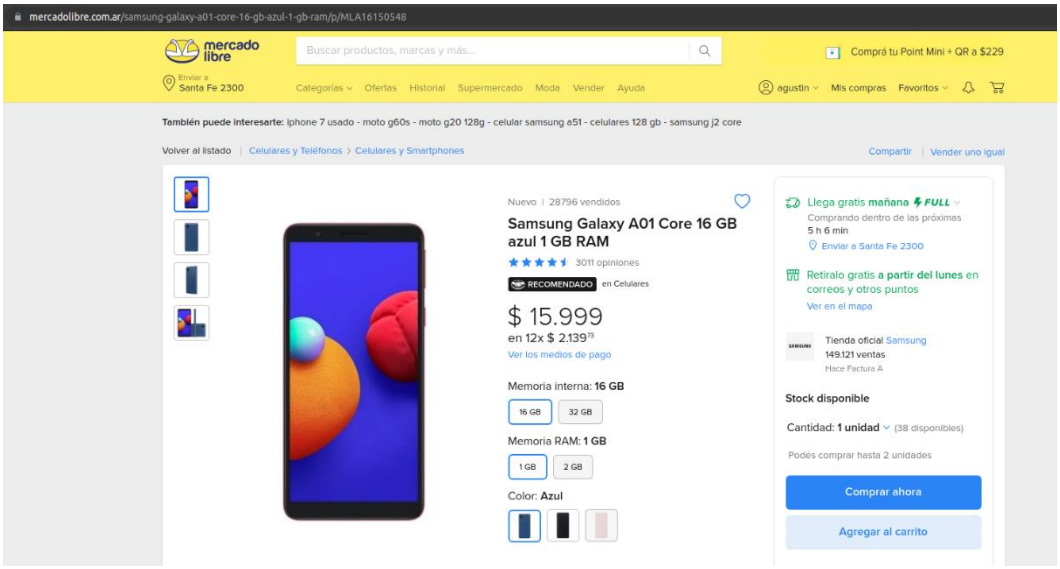

Imagen Nro 12: "Producto Samsung Galaxy A01 en la página Mercado Libre"

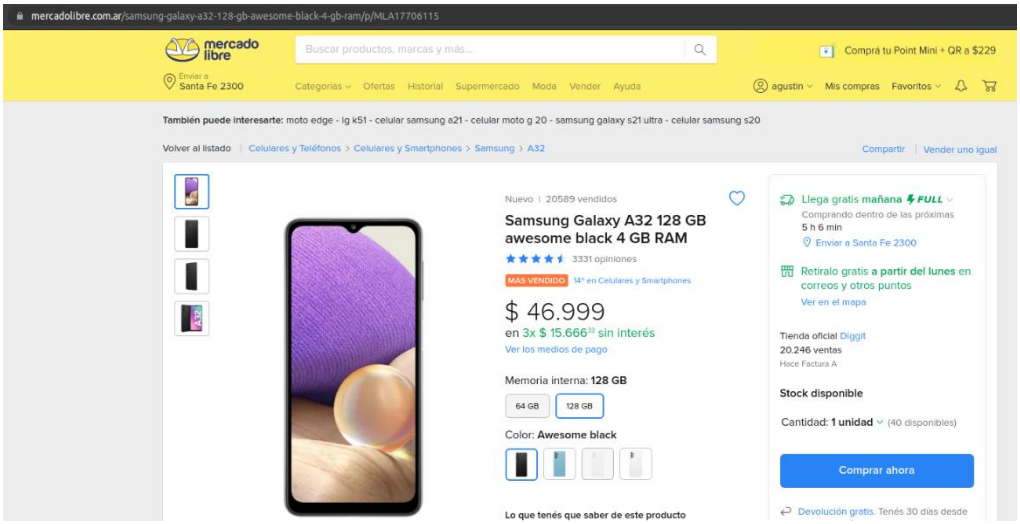

Imagen Nro 13: ""Producto Samsung Galaxy A32 en la página Mercado Libre"

Imagen Nro 14: "Ejemplo de comparación con los productos mencionados"

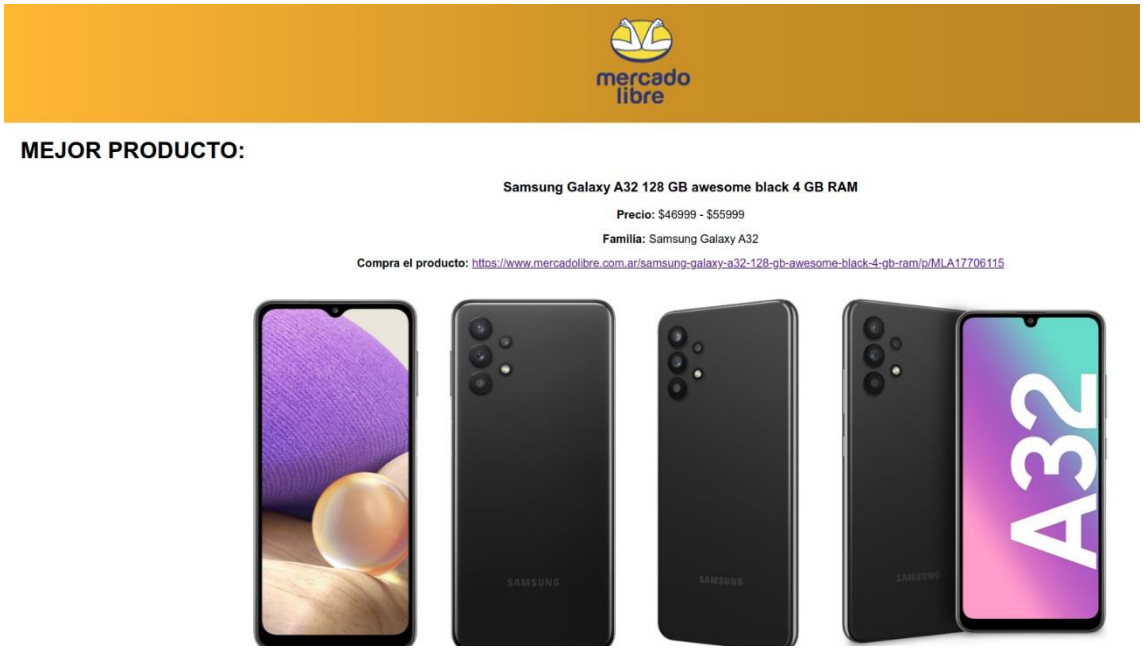

Imagen Nro 15: "Resultados obtenidos con los productos mencionados"

Ejemplo 2: “LG K50 VS Xiaomi Note 8”

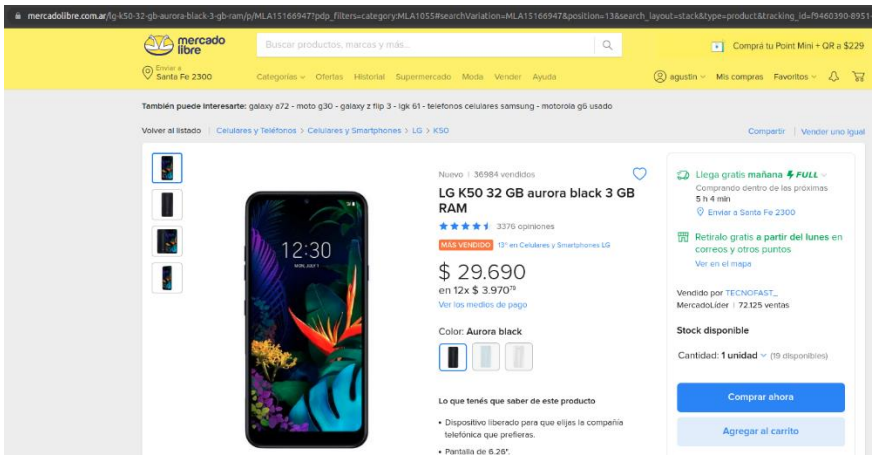

Imagen Nro 16: “Producto LG K50 en la página Mercado Libre”

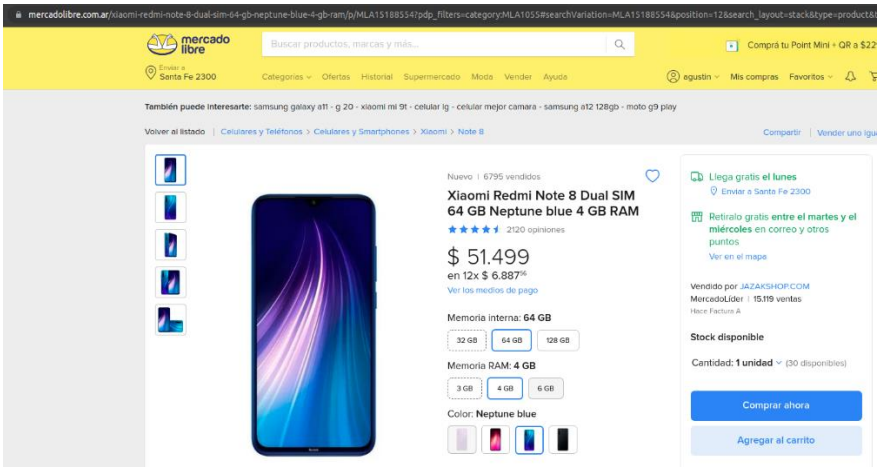

Imagen Nro 17: “Producto Xiaomi Redmi Note 8 en la página Mercado Libre”

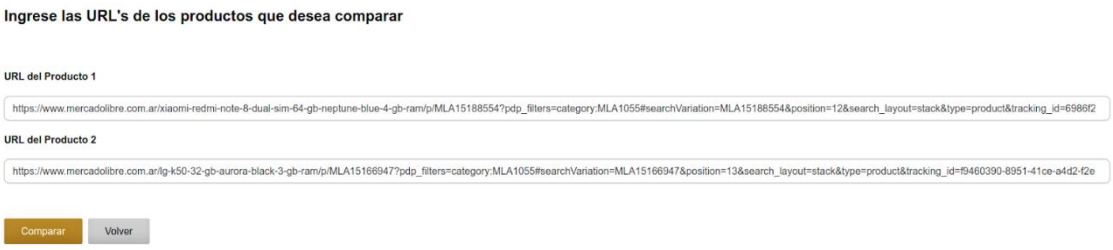

Imagen Nro 18: “Ejemplo de comparación con los productos mencionados”

MEJOR PRODUCTO:

LG K50 32 GB aurora black 3 GB RAM

Precio: $29690 - $29690

Familia: LG K Series K50

Compra el producto: https://www.mercadolibre.com.ar/lg-k50-32-gb-aurora-black-3-gb-ram/p/MLA15166947

Imagen Nro 19: "Resultados obtenidos con los productos mencionados"

Ejemplo 3: "Samsung Galaxy A52 VS Motorola Moto G20"

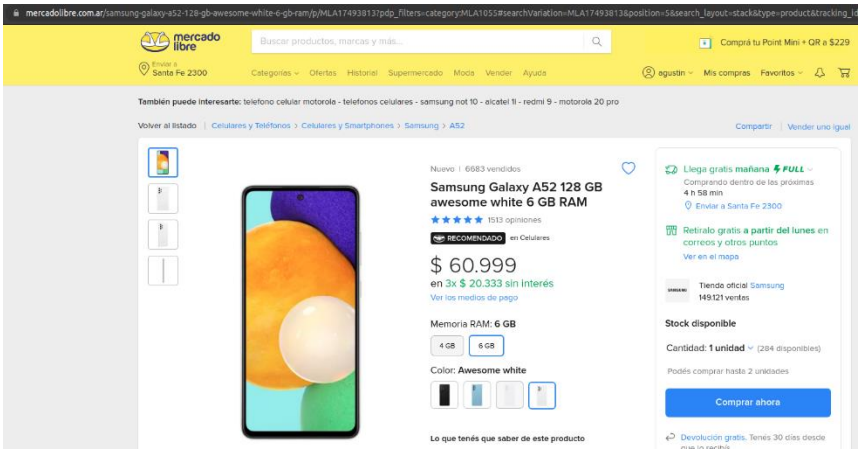

Imagen Nro 20: "Producto Samsung Galaxy A52 en la página Mercado Libre"

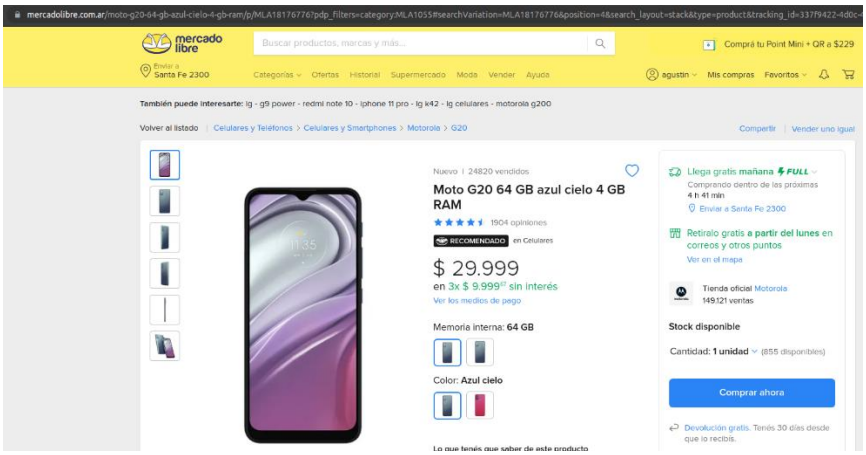

Imagen Nro 21: "Producto Motorola Moto G20 en la página Mercado Libre"

Ingrese las URL's de los productos que desea comparar

URL del Producto 1

https://www.mercadolibre.com.ar/moto-g20-64-gb-azul-cielo-4-gb-ram/p/MLA18176776?pdp_filters=category:MLA1055#searchVariation=MLA18176776&position=4&search_layout=stack&type=product&tracking_id=337f9422-4d0c-46c1-8857-f6e6

URL del Producto 2

https://www.mercadolibre.com.ar/samsung-galaxy-a52-128-gb-awesome-white-6-gb-ram/p/MLA17493813?pdp_filters=category:MLA1055#searchVariation=MLA17493813&position=5&search_layout=stack&type=product&tracking_id=da0846ae-0

Comparar | Volver

Imagen Nro 22: "Ejemplo de comparación con los productos mencionados"

MEJOR PRODUCTO:

Moto G20 64 GB azul cielo 4 GB RAM

Precio: $29999 - $39999

Familia: Motorola Moto G20

Compra el producto: https://www.mercadolibre.com.ar/moto-g20-64-gb-azul-cielo-4-gb-ram/p/MLA18176776

Imagen Nro 23: "Resultados obtenidos con los productos mencionados"

Bibliografía

- **Comenzando con datos – Análisis y visualización de datos usando Python** [https://datacarpentry.org/python-ecology-lesson-es/02-starting-with-data/]
- **Comparativa de Flask vs Django** [https://www.ilimit.com/blog/flask-vs-django/]
- **Qué es Flask y por qué usarlo en lugar de Django**
- [https://platzi.com/blog/flask-django-desarrollo-web-python/?gclid=Cj0KCQiA15yNBhDTARIsAGnwe0VYib6eBfAxfVnFEinW514W_hCm4ybo814oOpLdy3amgRpxabU7UcoaAr7sEALw_wcB&gclsrc=aw.ds]
- **Django vs Flask** [https://openwebinars.net/blog/django-vs-flask/]