Universidad Nacional Autónoma de México

Facultad de Ingeniería

Proyecto Final - GUI de Minería de Datos

Minería de Datos

Profesor:
Dr. Guillermo Gilberto Molero Castillo

Alumno:
Jose Antonio Torres Verastegui

Fecha de entrega:
13/Diciembre/2021

Semestre 2022-1

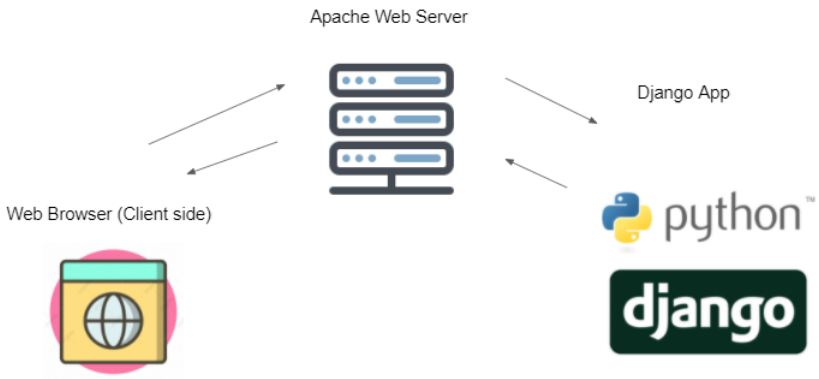
Índice

1. Objetivo

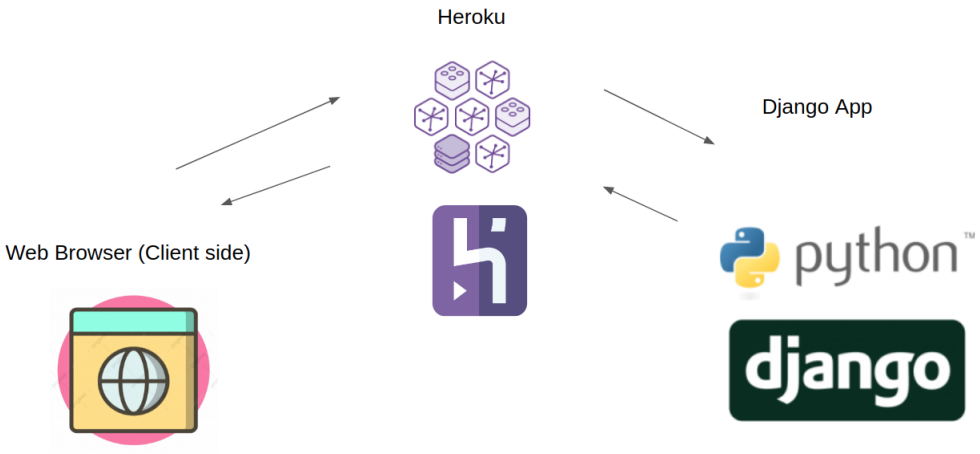
Crear una GUI que implemente algoritmos de Minería de Datos

2. Arquitectura

En el primer avance de proyecto se tenía la siguiente arquitectura.



Pero debido a que la configuración del servidor Apache para hacer un deploy de la aplicación Django era tardado, se optó por una opción con una configuración más sencilla, por lo que se utilizó Heroku y la arquitectura se modificó y la aplicación se ejecuta en los dynos de Heroku.



3. Metodología

Se utilizó una metodología tradicional de cascada con los siguientes pasos:

1. **Análisis**

   Lista de requerimientos:

   Funcionales

   - Recibe archivos csv
   - Implementa 8 componentes:
     - EDA
     - ACD
     - PCA
     - Clustering Jerárquico
     - K-means
     - árbol de decisión para pronóstico
     - árbol de decisión para clasificación
   - Capacidad de modificar parámetros de los algoritmos
   - Mostrar gráficos e información del desempeño de los algoritmos

   No Funcionales

   - Visualmente agradable
   - Interfaz gráfica fácil de usar

2. **Diseño**

   Lenguaje de programación

   La primer decisión tomada fue el lenguaje de programación a utilizar para implementar los componentes, el lenguaje seleccionado fue Python porque en las prácticas se observó como realizar los componentes deseados en Python.

   Tipo de aplicación

   Una vez decidido el lenguaje de programación, se procedió a decidir el tipo de aplicación, y se decidió por una aplicación web, esto debido a que las opciones para realizar aplicaciones standalone con GUI en Python no parecían tener elementos que son tan agradables visualmente.

   Herramientas adicionales para el desarrollo (Framework, etc)

   Al decidir que se iba a realizar una aplicación web con Python en el backend, se tenían dos principales frameworks para trabajar: Flask y Django. Se optó por Django porque se tenía más conocimiento de este. Adicional a esto, debido a que es una aplicación web se utilizaron HTML, CSS, Javascript y Bootstrap para el desarrollo de la parte del cliente.

   Prototipos

   Se crearon prototipos con el diseño utilizando Adobe XD, estos prototipos se entregaron en el primer avance del proyecto.

3. **Desarrollo**

   Para el desarrollo de la aplicación se creó un entorno virtual en la computadora de desarrollo, este se creo utilizando pipenv. La pc de desarrollo utiliza Ubuntu 20.04. El desarrollo fue relativamente sencillo debido a que en las prácticas ya se tenía la implementación de los componentes del proyecto.

4. **Pruebas**

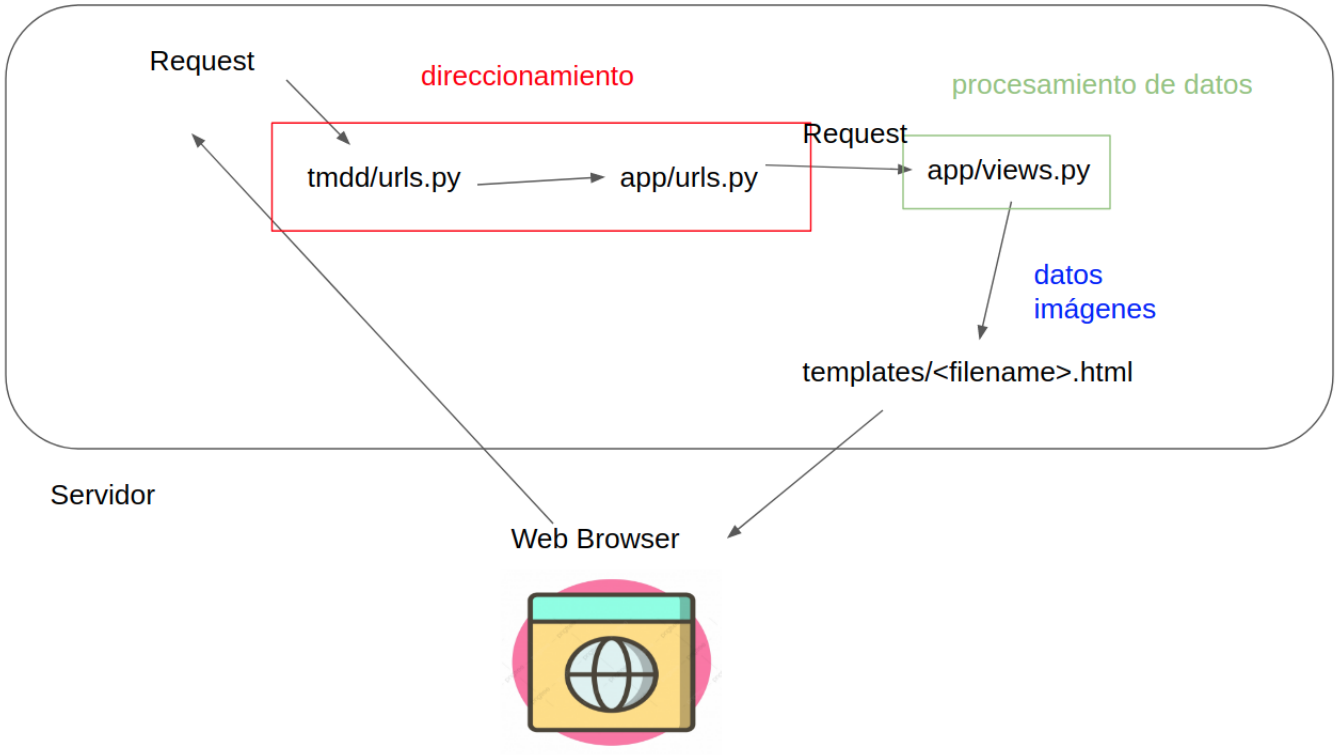
   Para realizar las pruebas se utilizó el servidor de desarrollo que incluye Django. Este servidor de desarrollo fue de gran utilidad debido a que así no es necesario instalar un servidor para ejecutar en este la aplicación y verificar que funciona. Pipenv fue de gran utilidad para instalar las bibliotecas necesarias y sus dependencias en el entorno virtual de desarrollo.

5. **Despliegue**

   Para el despligue realizado en Heroku, las configuraciones a la aplicación fueron mínimas, sólo se añadieron los archivos Procfile, requirements, y runtime para indicarle a Heroku las bibliotecas y sus versiones utilizadas en la aplicación. También se desactivó el modo DEBUG porque sólo es para poder ver los errores con mayor detalle en el desarrollo y también se agregó el middleware whitenoise para poder acceder a los elementos estáticos en un entorno distinto al de desarrollo.

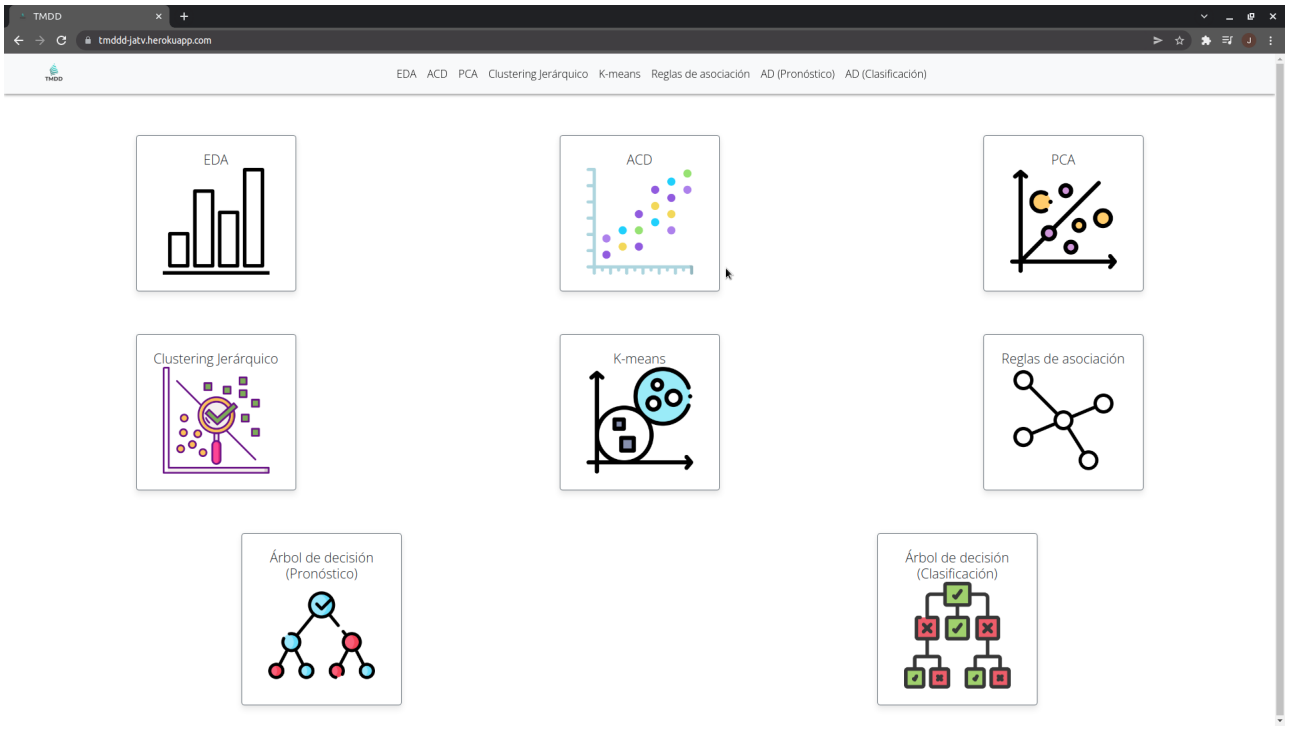
4. Arquitectura interna de la aplicación

En la aplicación de Django, los archivos llamados como urls.py se encargan de direccionar los requests que llegan a la aplicación, estos requests son direccionados a su correspondiente función en el archivo views.py, donde para cada elemento al que apunta la dirección, hay una función que se encarga de realizar el procesamiento requerido. En views.py es donde se ejecutan los algoritmos. Después del procesamiento, views.py envía los datos y el archivo html obtenido de templates hacia el web browser, donde este renderiza el html y muestra la página. Otro archivo muy importante del proyecto es settings.py, en el cual se realiza la configuración de los elementos del proyecto.

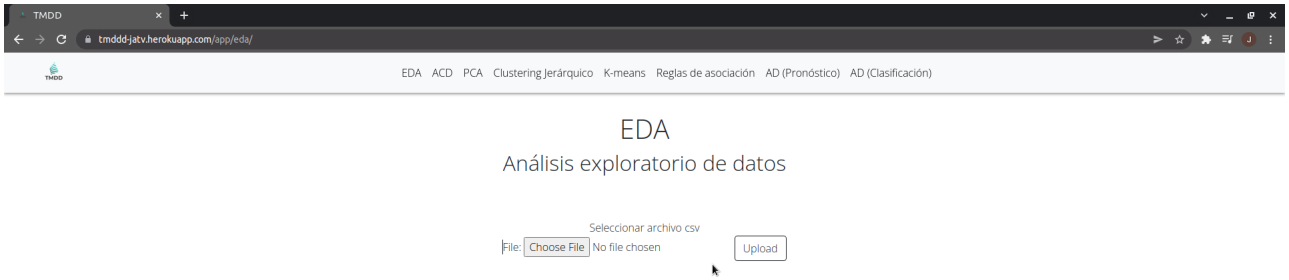


5. Resultados

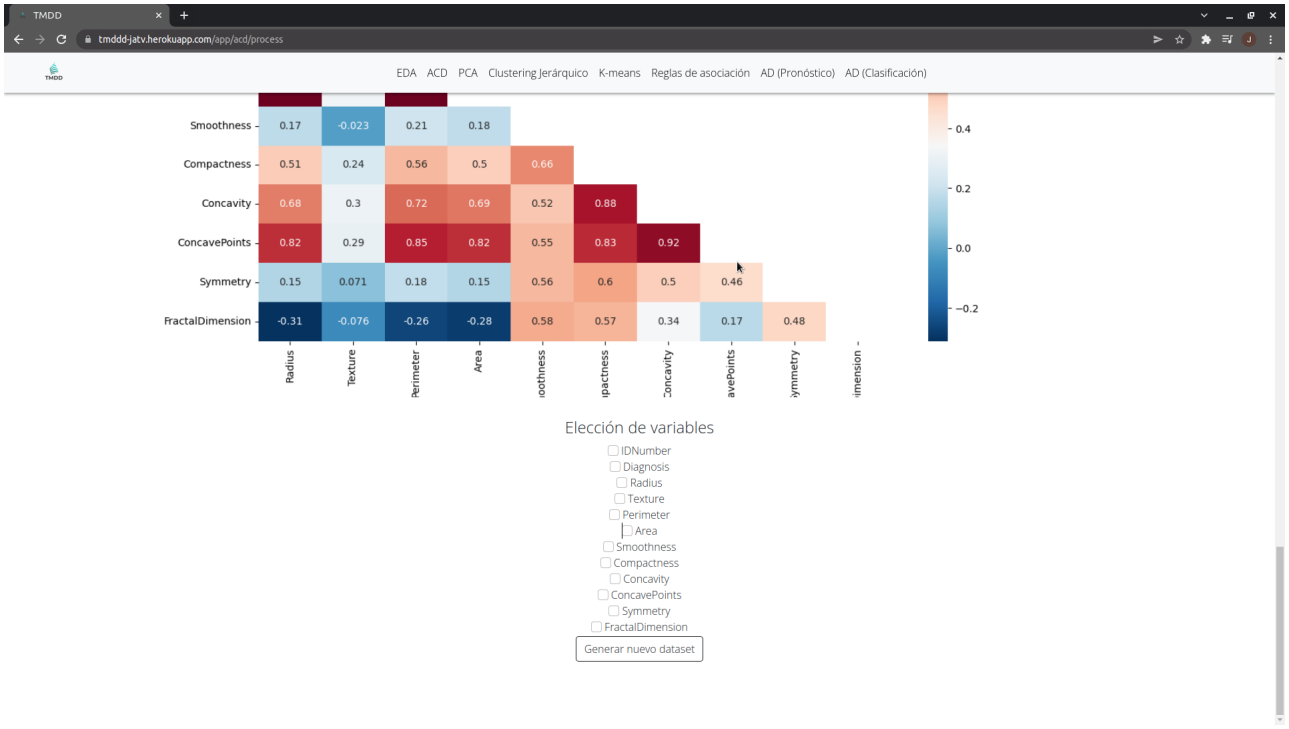
Para acceder a la aplicación se entra al siguiente link: `https://tmddd-jatv.herokuapp.com/`. Al entrar se mostrará la pantalla del menú. Cuando se entra a la aplicación en un principio, puede tardar en cargar, esto porque se tiene el plan gratuito de Heroku, por lo que cuando pasa un tiempo sin que la aplicación sea utilizada, Heroku la quita de la memoria del servidor y lo vuelve a cargar cuando se recibe una petición.



Para todos los algoritmos, se muestra una pantalla como la siguiente para que el usuario suba su archivo.



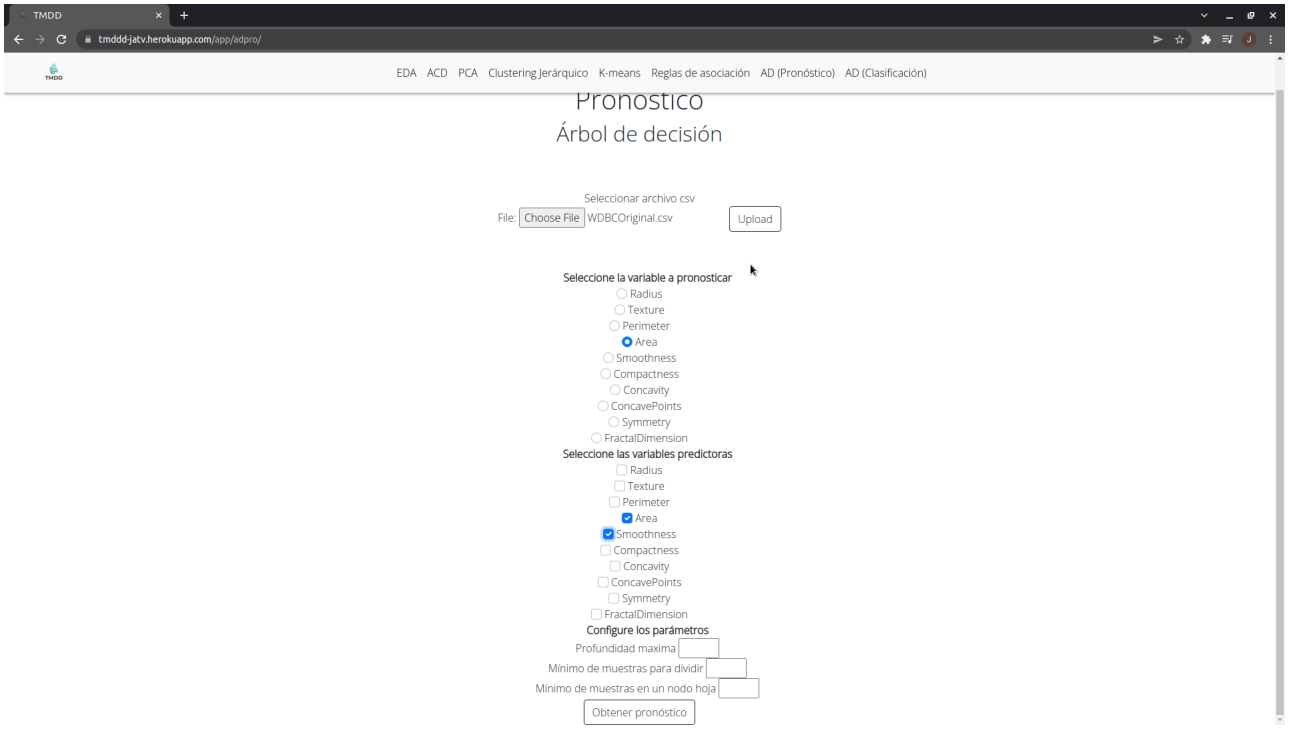
Para los algoritmos de selección de características al final de mostrar todas las imágenes y estadísticas se le da la opción al usuario de descargar el dataset con las variables que el seleccione.



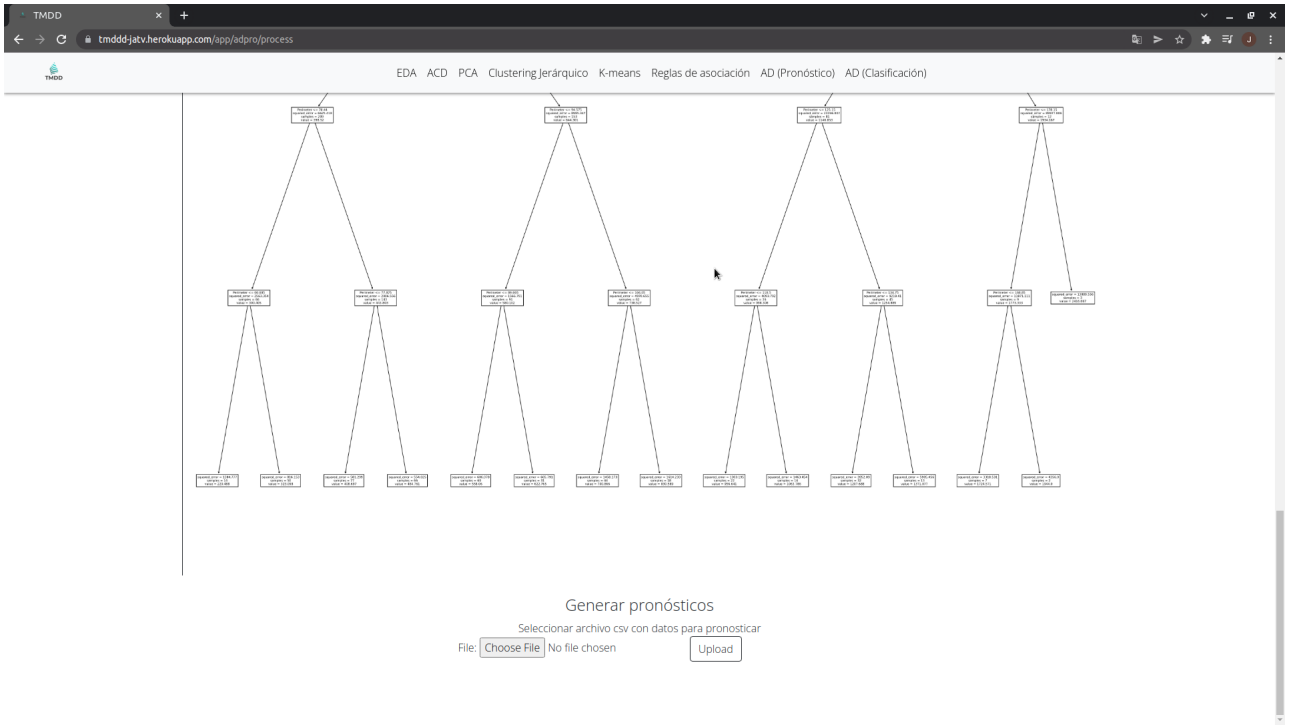
Para los algoritmos de clustering, se le pide al usuario que seleccione las variables a utilizar. Para el caso del clustering jerárquico también se le pide el número de clusters.

Para el algoritmo apriori se le pide al usuario el soporte, la confianza y la elevación mínima para usar como parámetro y generar las reglas.

En los algoritmos de árbol de decisión se le pide al usuario ingresar la variable a pronosticar o clasificar y las variables predictoras. Además el usuario puede ingresar la profundidad máxima, el mínimo de muestras para dividir un nod y el mínimo de muestras por hoja.



Al final de los algoritmos de clustering, el usuario puede subir un conjunto de datos para que sean pronosticados o clasificados, los resultados los puede descargar el usuario. En los resultados el pronóstico o clasificación se agrega como una columna a los datos que ingresa el usuario.



6. Trabajo a futuro

A futuro creo que la parte que se puede mejorar es la del manejo de los archivos que sube el usuario, sería bueno implementar que se borren cuando expire la sesión del usuario que subió el

archivo. Porque actualmente se borran cuando el usuario actual se va a una página donde ya no usará el archivo que subió, pero si el usuario recibe los resultados y cierra el navegador, no se borra el archivo subido. También al estar usando Heroku gratuito, si se sube un archivo muy grande y tarda mucho en procesarse, Heroku envía errores porque es demasiado el procesamiento que se necesitaría en esos casos.

7. Conclusiones

Se cumplió el objetivo de crear una GUI que implemente algoritmos de Minería de Datos, esto con la aplicación realizada en Django. La realización no fue tan complicada, pero si fue tardada, no fue complicada porque ya se tenía el código para todos los componentes debido a que estos fueron realizados en las prácticas a lo largo del semestre. Esto proyecto me pareció de gran utilidad para implementar los algoritmos vistos durante el semestre de una forma más sencilla de usar para cualquier usuario. Si bien no era necesario hacer deploy en un la nube para el proyecto, se optó por hacerlo para observar el proceso y las configuraciones necesarias, pero como se usó Heroku gratis, el rendmiento incluso es mejor cuando se ejecuta localmente el proyecto, por lo que para el video optó grabarse en la ejecución de forma local.