Máster en Ingeniería del Software: Cloud, Datos y Gestión TI

Data Science 2023

Análisis de variabilidad en distribuciones Linux

Proyecto Individual de José Antonio Zamudio Amaya

Participante del Grupo 2, formado por:

Diego Monsalves Vázquez
Carlos Núñez Arenas
José Antonio Zamudio Amaya

UNIVERSIDAD Đ SEVILLA

Índice del documento

1. Dominio y objetivo

Linux es un sistema operativo de código abierto que se basa en el núcleo de Linux. Fue creado en 1991 por Linus Torvalds y se ha convertido en uno de los sistemas operativos más populares del mundo, utilizado en servidores, dispositivos móviles y computadoras personales.

Las distribuciones de Linux, también conocidas como "distros", son variantes del sistema operativo Linux que están diseñadas para cumplir con diferentes necesidades y requisitos de los usuarios. Cada distribución tiene un conjunto único de características, paquetes de software preinstalados y herramientas de gestión de paquetes.

El ecosistema de Linux ofrece una alternativa gratuita y de código abierto a los sistemas operativos comerciales. Además, permite a los usuarios personalizar y adaptar su sistema operativo a sus necesidades específicas. También es conocido por su seguridad, estabilidad y flexibilidad. El uso de diferentes distribuciones de Linux y su ecosistema de gestión de paquetes permite a los usuarios tener un control total sobre su sistema operativo y su software.

Linux es un sistema operativo altamente variable debido a su naturaleza de código abierto y distribución en forma de distros. Los usuarios tienen acceso completo al código fuente del sistema operativo y de las aplicaciones que lo componen, lo que les permite modificar y adaptar el software según sus necesidades y preferencias. Además, la amplia variedad de distribuciones de Linux disponibles ofrece diferentes combinaciones de software y herramientas para satisfacer diferentes necesidades y preferencias de los usuarios.

El sistema de gestión de paquetes de Linux también contribuye a su alta variabilidad, ya que permite a los usuarios instalar, actualizar y eliminar paquetes de software según sea necesario. Esto les permite personalizar su sistema operativo para satisfacer sus necesidades específicas, ya sea para fines de desarrollo, de servidor, de escritorio, de multimedia, entre otros. Además, los paquetes de software pueden ser fácilmente intercambiados entre distribuciones de Linux, lo que permite a los usuarios mantener sus configuraciones y preferencias en diferentes sistemas. En general, la alta variabilidad de Linux es una de sus principales fortalezas y ha contribuido a su éxito como un sistema operativo de código abierto popular y altamente adaptable.

En este proyecto nos hemos propuesto analizar la variabilidad que tienen las distribuciones de Linux, fijándonos concretamente en Ubuntu, ya que es una de las distros más usadas en el mundo actualmente. Pero, ¿para qué nos sirve analizar la variabilidad?

1. Analizar la variabilidad del software puede ser importante para determinar si el software es capaz de cumplir con los requisitos y necesidades de los diferentes usuarios.
2. Analizar la variabilidad del software puede ayudar a identificar posibles problemas y defectos en el software antes de que sean lanzados al mercado.
3. Analizar la variabilidad del software puede ayudar a identificar los recursos necesarios para mantener el software a largo plazo.

2. Preguntas de interés

Para la realización del proyecto grupal hemos planteado dos preguntas distintas, que serán resueltas con técnicas distintas:

2.1 ¿Tiende a crecer el tamaño de las distribuciones a lo largo del tiempo?

Conforme el software se desarrolla y evoluciona, se espera que se agreguen nuevas funcionalidades para satisfacer las demandas de los usuarios y adaptarse a las nuevas tecnologías y tendencias del mercado. Sin embargo, a medida que se agregan nuevas funcionalidades al software, es probable que también aumente su complejidad y tamaño. Esto se debe a que cada nueva característica puede requerir la adición de nuevas líneas de código, lo que puede aumentar el tamaño del software.

El crecimiento en el tamaño del software puede tener varios efectos en su funcionalidad y eficiencia. En primer lugar, un software más grande puede requerir más recursos de hardware para su correcta ejecución, lo que puede afectar su rendimiento. Además, el aumento en la complejidad del software puede hacerlo más difícil de entender y modificar, lo que puede dificultar su mantenimiento y evolución. Por otro lado, un software más grande también puede ofrecer más opciones y características a los usuarios, lo que puede aumentar su valor y utilidad. Además, un software más grande también puede incluir mejoras en la seguridad y estabilidad del sistema, lo que puede aumentar su fiabilidad y evitar posibles vulnerabilidades.

2.2 ¿Cómo fluctúan los paquetes entre las distribuciones?

Cuando se lanza una nueva versión del software, es común que algunas características se retiren o se modifiquen para mejorar su eficiencia o adaptarse a las demandas del mercado. A su vez, se pueden agregar nuevas funcionalidades para mantener el software actualizado y competitivo.

En algunos casos, ciertas funcionalidades pueden desaparecer debido a su baja demanda o a que han sido reemplazadas por otras funciones más útiles. También puede haber cambios en la tecnología subyacente que hagan que algunas características sean obsoletas o innecesarias. Sin embargo, en otros casos, pueden agregarse nuevas funcionalidades que se ajusten a las necesidades de los usuarios o para competir con otros paquetes en el repositorio.

2.3 ¿Es normal que aumente el número de dependencias de un paquete respecto al tiempo?

Cuando un paquete de software se desarrolla y evoluciona, es común que se vaya ampliando su funcionalidad y que se agreguen nuevas características. Estas nuevas características pueden requerir el uso de otras herramientas o bibliotecas de software para poder funcionar correctamente, lo que a su vez aumenta el número de dependencias del paquete.

Además, a medida que el software se utiliza en diferentes entornos y situaciones, pueden surgir nuevos requisitos y necesidades que requieran la inclusión de nuevas dependencias en el paquete. Por ejemplo, si el paquete se utiliza en un entorno de servidor web, puede ser necesario agregar una biblioteca para manejar la comunicación con el servidor.

2.4 ¿Han desaparecido paquetes? ¿Han aparecido nuevos paquetes?

Es común que los paquetes desaparezcan o sean reemplazados por otros nuevos a medida que el repositorio evoluciona y las necesidades de los usuarios cambian. Por ejemplo, un paquete que antes era popular puede perder popularidad a medida que surjan alternativas más eficientes o fáciles de usar.

Por otro lado, también es común que aparezcan nuevos paquetes a medida que se desarrollan nuevas tecnologías o se identifican nuevas necesidades de los usuarios. Estos nuevos paquetes pueden ser una respuesta a los problemas que los paquetes existentes no pueden resolver o pueden ofrecer características únicas que los hacen atractivos para los desarrolladores y los usuarios.

3. Preprocesado

En primer lugar, se ha llevado a cabo un proceso de generación de Dockerfiles para las versiones 18.04, 20.04, 22.04 y 22.10 de Ubuntu. A partir de estos Dockerfiles, se han listado todos los paquetes del repositorio principal de cada versión en un archivo de texto, que ha sido procesado posteriormente para generar un archivo CSV con información detallada para cada paquete.

Este archivo CSV incluye las siguientes columnas para cada versión de Ubuntu: "Distro-Version, Distro-Year, Package, Description, Section, Version, Architecture, Priority, Essential, Build-Essential, Maintainer, Original-Maintainer, Size, Installed-Size, Depends, Pre-Depends, Recommends, Conflicts, Suggests, Replaces, Provides".

Una vez generado el archivo CSV para cada versión de Ubuntu, se ha llevado a cabo un proceso de concatenación para unir los cuatro CSVs generados en uno solo. Además, se ha aplicado un proceso adicional de preprocesamiento para transformar los valores nulos a NaN y eliminar aquellas columnas que no aportan información relevante.

Finalmente, se ha desarrollado un script para montar el archivo CSV en un datawarehouse, lo que permite realizar consultas SQL para analizar y procesar los datos de manera eficiente. Este proceso de preprocesamiento ha permitido obtener una fuente de datos sólida y estructurada para llevar a cabo análisis posteriores y tomar decisiones informadas basadas en datos precisos y confiables.

El archivo generado a través del proceso de preprocesamiento cuenta con un total de 21 columnas y 299.733 filas, lo que supone un tamaño de 170 MB. Cabe destacar que el proceso de generación de este archivo llevó inicialmente alrededor de 12 horas para completarse, pero posteriormente fue optimizado para reducir el tiempo de ejecución a tan solo 24 minutos.

4. Análisis exploratorio

El objetivo de este apartado del trabajo es el de conocer más a fondo los datos con los cuales vamos a trabajar, visualizar de forma inmediata conclusiones rápidas e iniciales sobre el conjunto de datos e identificar posibles fallos y valores perdidos en el csv.

Lo primero que observamos es un reparto equitativo en los paquetes por cada versión de la distribución Ubuntu que va decreciendo con el paso del tiempo, así que a pesar de lo que se podría pensar de que un proyecto crece en tamaño con el tiempo, en el caso de Ubuntu vemos que la cantidad de paquetes se disminuye con el paso del tiempo. En el dataset tenemos un 26.9% de la versión Bionic, 26.2% de Focal, 23.9% de Jammy y 23.0% de Kinetic.

Posteriormente visualizamos los contribuyentes y mantenedores de paquetes más presentes por cada versión. Obtenemos que para todas las versiones, Ubuntu es el mantenedor más frecuente con diferencia, que contribuye en todas las versiones con más de 66.000 paquetes, seguido de Canonical y Kubuntu, que no llegan a las 10.000 aportaciones. El top de los mantenedores originales de los paquetes (columna “Original-Maintainer”, frente a “Maintainer”), sin embargo, está liderado por el equipo de Debian.

Observamos también la media del tamaño de los paquetes por cada versión, donde podemos ver también cómo esta media va disminuyendo con el paso del tiempo, con Bionic: 2996.0 Kb, Focal: 3519.39 Kb, Jammy: 2598.2 Kb y Kinetik: 1962.47 Kb.
En las tablas de contingencia propuestas, vemos cómo el csv corresponde con las versiones, y cada año está limitado a una versión, excepto en 2022 que contiene las versiones Jammy y Kinetic.

Visualizamos también con la ayuda de estas tablas que la mayoría de paquetes requeridos están únicamente soportados por la arquitectura amd64, y que el tamaño de los paquetes está altamente correlacionado con su prioridad y si están marcados como esenciales. Es decir, los paquetes esenciales tienden a ser más grandes que los no esenciales, y los que tienen la prioridad de opcionales tienden a ser los más grandes también. Se ve también que no hay paquetes requeridos que no estén marcados como esenciales.

Por último identificamos los valores perdidos en el dataset, los cuales son bastante pocos, ya que ninguna columna tiene más de un 0.1% de pérdida. “Pre-Depends” es la columna con más pérdidas, con tan solo 80 valores vacíos. Le sigue “Provides” con 31 valores. Con esto podemos afirmar que nuestro dataset es bastante completo y consistente.

5. Resultados

5.1 ¿Tiende a crecer el tamaño de las distribuciones a lo largo del tiempo?

Los pasos que hemos seguido para obtener la visualización de la Figura 1 han sido:

1. Preprocesamiento: nos quedamos con los datos donde la prioridad es 'important' o 'required'. (ya que son estos los paquetes que se instalan por defecto)
2. Creamos una columna con los tamaños en MB (la recibimos en bytes).
3. Creamos una serie temporal. Con la columna nueva creada y una periodicidad de 12 meses.

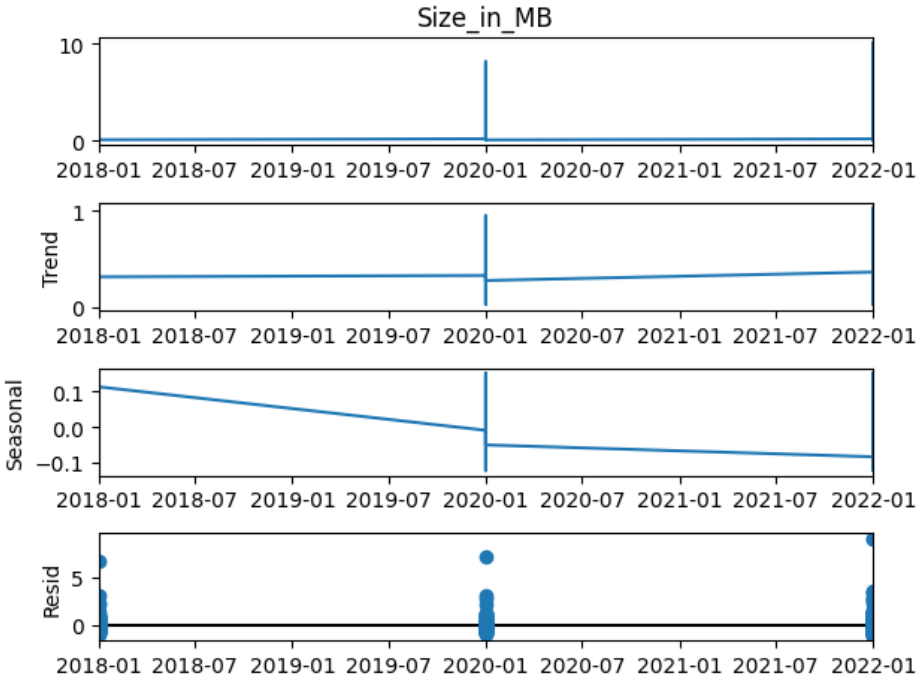

Figura 1. Análisis de series temporales

Viendo la figura, podemos destacar que, de manera ordenada, vemos la serie de tiempo original de los tamaños de las distribuciones en megabytes, la tendencia de la serie de tiempo, la estacionalidad de la serie de tiempo, y el residuo de la serie de tiempo.
Por lo que, la descomposición muestra que hay una tendencia ascendente en el tamaño de las distribuciones y una estacionalidad anual que se repite cada 12 meses.

Tras ver esto, creemos que no es muy concreto ya que solo tenemos info de los paquetes en 4 instantes de tiempo en nuestro csv. Por ello, decidimos añadir una regresión lineal para modelar la relación entre el tamaño de la distribución y el año.

En dicha regresión lineal, observamos que el coeficiente de regresión indica que hay un aumento promedio de 19.64 KB en el tamaño de los paquetes, algo bastante significativo. El valor de p-value de 0.26 indica que la relación entre el tamaño de los paquetes y el tiempo no es estadísticamente significativa. Esto significa que no podemos rechazar la hipótesis nula de que el coeficiente de regresión es igual a cero y que la variación en el tamaño de los paquetes no está relacionada con el tiempo.

5.2 ¿Cómo fluctúan los paquetes entre las distribuciones?

Los pasos que hemos seguido para obtener la visualización de la Figura 2 han sido:

1. Identificación de los paquetes de cada distro: En este paso, se identificaron los paquetes que se incluyen en cada distribución.
2. Análisis de aparición para cada par de distros: En este paso, se analizó la frecuencia con la que aparece cada par de distribuciones en los registros de instalación de paquetes.
3. Análisis de desaparición para cada par de distros: En este paso, se analizó la frecuencia con la que se desinstalan paquetes de cada par de distribuciones.
4. Visualización de las variables finales: En este paso, se utilizó la técnica de visualización de datos barplot para presentar los resultados del análisis de una manera clara y comprensible.

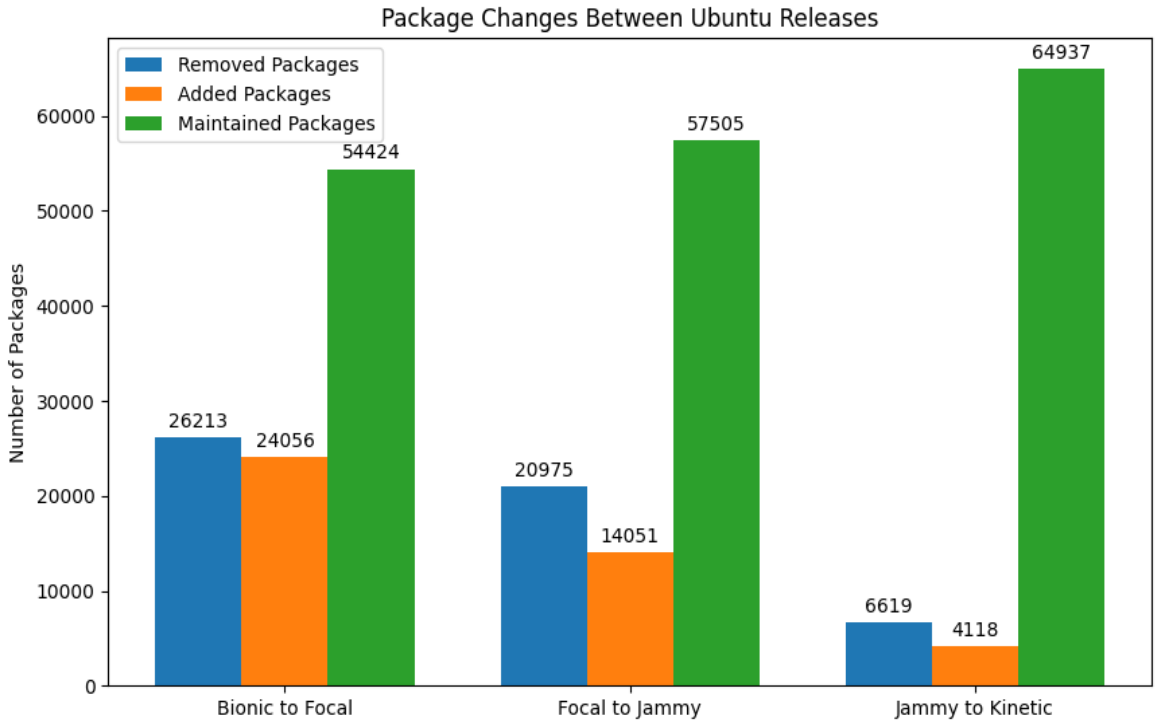

Figura 2. Resultados del análisis de paquetes

La visualización de la Figura 2 sugiere que a medida que pasa el tiempo, las distribuciones tienden a estabilizarse y converger en términos de los paquetes disponibles. Es decir, las distribuciones se vuelven más similares a medida que evolucionan. Esta convergencia puede ser causada por una mayor colaboración y una mejor comunicación entre los desarrolladores de diferentes distribuciones, lo que les permite compartir paquetes comunes. También se puede concluir que la aparición de paquetes nuevos es mayor que la desaparición de paquetes. Esto sugiere que hay una tendencia hacia el crecimiento en el número de paquetes disponibles en las distribuciones a lo largo del tiempo.

5.3 ¿Es normal que aumente el número de dependencias de un paquete respecto al tiempo?

Los pasos que hemos seguido para obtener el resultado de este test estadístico han sido:

1. Preprocesado de los datos: En este paso, se eliminaron las columnas innecesarias y se convirtió la columna de Dependencias en una lista. También se calculó el número de dependencias de cada paquete y se agregó como una nueva columna en el dataframe.
2. Análisis de correlación: En este paso, se agruparon los datos por versión y se calculó la media del número de dependencias para cada versión. Se asignó un número a cada versión y se creó una nueva serie de valores numéricos para las versiones. Se calculó la correlación entre la serie de valores numéricos de las versiones y la serie de valores numéricos del número de dependencias.
3. Prueba ANOVA: En este paso, se obtuvieron los datos de cada grupo y se realizó el test ANOVA para determinar si existían diferencias significativas en el número de dependencias entre las diferentes versiones.
4. Prueba de Tukey: En este paso, se realizó la prueba de Tukey para identificar qué pares de versiones presentaban diferencias significativas en el número de dependencias.

Los resultados obtenidos indican que no hay una correlación significativa entre el número de versiones y el número de dependencias. Esto se evidencia por el valor de PearsonRResult, que muestra una correlación positiva moderada pero no significativa (estadístico de 0.876 y un p-valor de 0.123). Además, el análisis de varianza ANOVA muestra un alto valor de F-estadístico y un p-valor muy bajo, lo que indica que hay una diferencia significativa entre las medias de los grupos comparados. Sin embargo, al aplicar la prueba de comparación múltiple Tukey, se concluye que no hay diferencias significativas entre los grupos, reforzando la idea de que no hay una correlación significativa entre el número de versiones y el número de dependencias.

5.4 ¿Han desaparecido paquetes? ¿Han aparecido nuevos paquetes?

Los pasos que hemos seguido para obtener la visualización de la Figura 3 han sido:

1. En primer lugar, procedemos a leer el conjunto de datos, para así poder seleccionar los paquetes correspondientes a cada distribución Ubuntu, incluyendo Bionic, Focal, Jammy y Kinetic.
2. A continuación, llevamos a cabo un análisis de las versiones correspondientes a cada una de las distribuciones mencionadas. Este análisis incluye la comparación de las versiones n y n+1, así como de las transiciones entre Bionic-Focal, Focal-Jammy y Jammy-Kinetic.
3. A partir de este análisis, determinamos los paquetes que desaparecen, los que se mantienen y los que aparecen en cada uno de los pares de distribuciones.
4. Finalmente, representamos visualmente la distribución de peso de los paquetes con un diagrama de cajas.

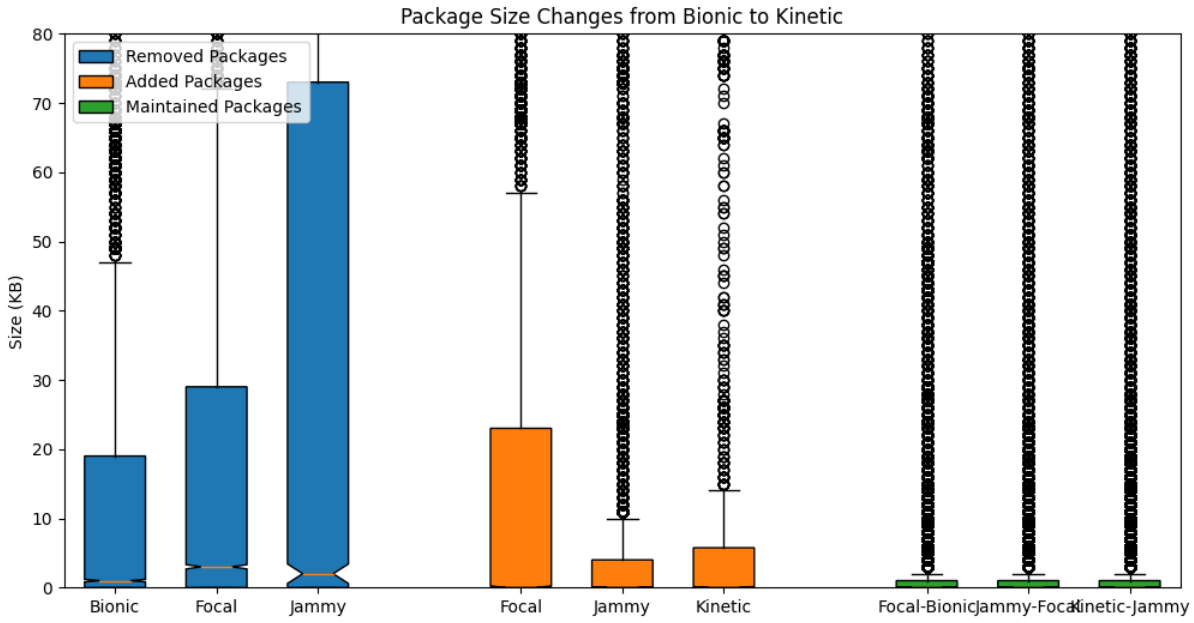

Figura 3. Diagrama de cajas para la distribución del peso de los paquetes

En resumen, se observa que en cada evolución entre distribuciones Ubuntu (Bionic, Focal, Jammy y Kinetic), el peso de los paquetes que son eliminados es significativamente mayor que el peso de los paquetes que son añadidos. Sin embargo, el peso de los paquetes que se mantienen permanece constante a lo largo de las distintas transiciones.

La tendencia general es reducir el número de paquetes en cada nueva versión, eliminando aquellos que ya no son necesarios o que pueden ser reemplazados por otras soluciones más eficientes. Además, esto reduce el peso general del repositorio, ya que se eliminan paquetes pesados y aparecen paquetes más livianos.

6. Conclusiones

Para finalizar este documento, queremos acabar aportando algunas conclusiones que hemos obtenido gracias a la realización de este proyecto:

6.1 Coste de generar datasets propios:

La generación de nuestro propio dataset ha requerido un gran esfuerzo en términos de tiempo. Aunque tenemos un dataset exclusivo que ofrece una gran cantidad de información, el costo de su generación ha sido considerable y, posiblemente, desproporcionado en relación con los requisitos de la asignatura. Por tanto, es fundamental evaluar minuciosamente si la creación de un dataset propio es la mejor opción en función de los recursos disponibles y los objetivos del proyecto.

6.2 Importancia de preprocesar la información:

El procesamiento de datos es una parte fundamental del proceso de aprendizaje automático. Los datos sin procesar pueden contener errores, ruido y redundancias que pueden afectar negativamente el rendimiento del modelo. El preprocesamiento de datos implica la limpieza de datos, la selección de características relevantes y la transformación de datos para adaptarse mejor al modelo de aprendizaje automático que se utilizará. Un buen preprocesamiento de datos puede mejorar significativamente la precisión y eficiencia del modelo.

6.3 La visualización no es suficiente:

Aunque la visualización de datos puede ser una herramienta poderosa para explorar y comprender los datos, no es suficiente para realizar un análisis completo de los datos. La visualización puede ser útil para detectar patrones y tendencias, pero no puede proporcionar información detallada sobre la distribución de los datos y las relaciones entre las variables. Además, algunos patrones pueden no ser visibles en una visualización y solo pueden ser descubiertos mediante técnicas de análisis más avanzadas. Por lo tanto, es importante utilizar una variedad de técnicas de análisis de datos en conjunto con la visualización.

6.4 Ajustarse bien a la evaluación:

En nuestra experiencia personal en el proyecto, hemos invertido un considerable esfuerzo en tareas que podrían considerarse menos relevantes para el alcance de esta asignatura. Por ejemplo, dedicamos una cantidad significativa de tiempo al preprocesamiento del dataset, la creación de dockerfiles y la optimización del parseo de archivos TXT a CSV para reducir el tiempo de ejecución a 20 minutos. Es posible que estas tareas no tengan un peso significativo en la evaluación final del proyecto. En contraste, otros grupos que han optado por utilizar datasets existentes en Kaggle han ahorrado tiempo y han podido enfocarse más en la resolución de preguntas relevantes para el proyecto.

7. Trabajo futuro

A continuación, se presentan algunas sugerencias para el trabajo futuro en relación con el proyecto:

7.1 Aumentar el número de versiones para Ubuntu:

Actualmente, el dataset contiene información sobre varias versiones de Ubuntu, pero se podría considerar la inclusión de más versiones para obtener una imagen más completa de su evolución. Sería necesario investigar las versiones más relevantes y obtener información sobre ellas para incluirlas en el dataset.

7.2 Añadir nuevas distribuciones:

Además de Ubuntu, se podrían incluir otras distribuciones populares, como Debian o Fedora, para ampliar el alcance del dataset y comparar diferentes sistemas operativos. Esto requeriría investigar las distribuciones más relevantes y obtener información sobre ellas para incluirlas en el dataset.

7.3 Redactar un artículo:

Una vez completado el análisis global, se podría considerar la redacción de un artículo que resuma los hallazgos más relevantes del proyecto. Este artículo podría ser publicado en una revista científica relacionada con la gestión de la variabilidad software o el mining variability, como SPLC, ConfWS o JISBD.