//UCIENCIA/23
CONVENCIÓN CIENTÍFICA INTERNACIONAL

Temática: Pruebas de Software (V Taller Internacional de Ingeniería y Calidad de Software)

Técnicas de pruebas y herramientas en servicios de Machine Learning

Techniques for testing and tools in Machine Learning services

Amalia García Cardoso [1*], Héctor Raúl González Diez [2], José Antonio Castaño Guevara [3]

[1] Centro de Tecnologías para la Formación (FORTES). Universidad de las Ciencias Informáticas. acardoso@uci.cu
[2] Dirección de Ciencia, Tecnología e Innovación. Universidad de las Ciencias Informáticas. hglez@uci.cu
[3] Centro de Tecnologías para la Formación (FORTES). Universidad de las Ciencias Informáticas. joseantonio@uci.cu

[*] Autor para correspondencia: acardoso@uci.cu

Resumen
En el siguiente trabajo se realiza una revisión científica de las técnicas de pruebas y herramientas para garantizar la calidad y fiabilidad de los servicios de Machine Learning. Las técnicas de pruebas pueden ser unitarias, de integración, de sistema y de aceptación. Las pruebas unitarias se utilizan para probar componentes individuales del servicio de Machine Learning, como modelos y algoritmos de aprendizaje automático, para asegurarse de que funcionan correctamente. Las de integración se utilizan para probar cómo funcionan los componentes individuales del servicio cuando se integran. Las de sistema se utilizan para probar el sistema en su conjunto, asegurándose de que cumpla con los requisitos y expectativas del usuario. Las de aceptación se utilizan para asegurarse de que el servicio cumpla con los requisitos y expectativas del usuario antes de su lanzamiento. En el documento se muestran otras pruebas importantes, las de rendimiento y las de seguridad, que son esenciales para garantizar que el servicio de Machine Learning funcione de manera eficiente y segura. Otro aspecto abordado en el trabajo son las herramientas, las cuales permiten automatizar las pruebas de software ofreciendo una serie de ventajas tales como ejecutar pruebas más rápido, eficientemente y con menos esfuerzo, realizar pruebas más exhaustivas y repetitivas identificar errores y problemas en los modelos y algoritmos de aprendizaje automático antes de su implementación, entre otras. Se presentan múltiples herramientas, entre las que se destacan TensorFlow, PyTorch, Scikit-learn, Keras, Apache Spark MLlib, entre otras.

Palabras clave: machine learning, pruebas, calidad de software, herramientas.

Abstract
In this paper, a scientific review of testing techniques and tools to ensure the quality and reliability of Machine Learning services is conducted. Testing techniques can be unit testing, integration testing, system testing, and acceptance testing. Unit testing is used to test individual components of the Machine Learning service, such as models and machine learning algorithms, to ensure they function correctly. Integration testing is used to test how individual components of the service function when integrated. System testing is used to test the Machine Learning system as a whole, ensuring it meets user requirements and expectations. Acceptance testing is used to ensure the service meets user requirements

and expectations before its release. The paper also covers other important tests, such as performance testing and security testing, which are essential to ensure the Machine Learning service functions efficiently and securely. Another aspect addressed in the paper is the use of tools, which allow for the automation of software testing and offer a series of advantages, such as faster and more efficient testing, more exhaustive and repetitive testing, identification of errors and problems in Machine Learning models and algorithms before implementation, among others. Multiple tools are presented, including TensorFlow, PyTorch, Scikit-learn, Keras, Apache Spark MLlib, among others.

Keywords: *Machine learning, testing, software quality, tools.*

Introducción

Machine Learning (ML) es una rama de la inteligencia artificial (IA) que se enfoca en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender a partir de datos y mejorar su rendimiento en tareas específicas sin ser programadas explícitamente para ello (Rojas, s. f.).

En lugar de programar reglas específicas para realizar una tarea, como se hace en la programación tradicional, en el aprendizaje automático se proporciona un conjunto de datos de entrenamiento al modelo de ML, y este utiliza algoritmos para aprender patrones y relaciones en los datos y hacer predicciones o tomar decisiones sobre nuevos datos. Luego de la implementación del servicio de ML, es imprescindible realizarle Pruebas de Software (PSW), para garantizar que cumpla con los requisitos y expectativas definidas por el usuario (Rojas, s. f.).

Las PSW son un proceso sistemático y controlado para evaluar la calidad de un software y asegurar que cumpla con los requisitos y expectativas del usuario. Son esenciales para garantizar que el software sea confiable, seguro, eficiente y fácil de usar y pueden ser realizadas en varias etapas del ciclo de vida del desarrollo de software, incluyendo (Serna M. et al., 2019):

1. Pruebas de unidad: se realizan en la etapa de desarrollo para probar componentes individuales del software, como funciones o módulos, para asegurarse de que funcionan correctamente.
2. Pruebas de integración: se realizan después de que se han probado los componentes individuales del software para asegurarse de que funcionan correctamente cuando se integran.
3. Pruebas de sistema: se realizan en todo el sistema para probar que cumple con los requisitos y expectativas del usuario.
4. Pruebas de aceptación: se realizan para asegurarse de que el software cumpla con los requisitos y expectativas del usuario antes de su lanzamiento.

Las PSW pueden ser realizadas manualmente o mediante el uso de herramientas automatizadas. Las manuales implican la revisión manual del software para identificar errores y problemas, mientras que las automatizadas utilizan

herramientas de software especializadas para ejecutar pruebas de manera automatizada y detectar errores y problemas de manera más rápida y precisa. Estas pruebas tienen múltiples ventajas, algunas de las cuales incluyen (Diaz et al., 2020):

1. Mejora de la calidad del software: esenciales para mejorar la calidad del software y asegurarse de que cumple con los requisitos y expectativas del usuario. Identifican errores y problemas en el software que pueden ser corregidos antes de que el software sea lanzado.
2. Reducción de costos: pueden reducir los costos al identificar errores y problemas en el software antes de que se lance. Esto evita el costo adicional de corregir errores y problemas después del lanzamiento y reduce el costo de reparación y soporte técnico.
3. Ahorro de tiempo: pueden ahorrar tiempo al identificar errores y problemas en el software de manera temprana. Esto evita la necesidad de corregir errores y problemas después del lanzamiento y reduce el tiempo necesario para reparar y mantener el software.
4. Mejora de la satisfacción del usuario: aseguran que el software cumpla con los requisitos y expectativas del usuario. Esto mejora la satisfacción del usuario y reduce la necesidad de soporte técnico y reparación.
5. Cumplimiento de los estándares de calidad: necesarias para cumplir con los estándares de calidad y seguridad en la industria del software.
6. Identificación de problemas de rendimiento: Las pruebas de software pueden identificar problemas de rendimiento en el software, como problemas de velocidad y capacidad. Esto permite a los desarrolladores corregir estos problemas antes del lanzamiento y asegurarse de que el software funcione de manera eficiente.
7. Identificación temprana de errores: permiten identificar errores y problemas en el software de manera temprana, cuando es más fácil y menos costoso corregirlos. Esto reduce el riesgo de errores y problemas en el software después del lanzamiento y evita la necesidad de correcciones costosas y difíciles de implementar.
8. Reducción del riesgo: reducen el riesgo de fracaso del software al identificar errores y problemas antes del lanzamiento, reduciendo la posibilidad de problemas de seguridad, pérdida de datos y otros riesgos para el usuario.
9. Mejora de la reputación de la empresa: mejoran la reputación de la empresa al entregar productos de alta calidad y confiables. Eleva la satisfacción del cliente y aumenta la lealtad y la confianza del cliente en la empresa.
10. Soporte para la toma de decisiones: proporcionan información valiosa sobre la calidad y el rendimiento del software que puede ser utilizada para tomar decisiones informadas. Esto puede incluir decisiones sobre el lanzamiento del software, la corrección de errores y la mejora continua del software.

11. Mejora de la eficiencia: pueden mejorar la eficiencia del proceso de desarrollo de software al identificar errores y problemas de manera temprana y reducir la necesidad de correcciones fuera de tiempo, que atenten contra el cronograma de entrega del producto.

Las diferentes técnicas de pruebas, a partir de sus conceptos y ventajas, enfocadas en los servicios de Machine Learning se utilizan para evaluar el desempeño de los modelos en desarrollo y asegurar que el sistema funcione correctamente en el entorno de producción. Algunas de las técnicas de pruebas en servicio de ML más comunes incluyen (Serna M. et al., 2019):

1. Prueba de extremo a extremo: implica el monitoreo de todo el flujo de trabajo del modelo de ML, desde la entrada de datos hasta la salida del modelo, para verificar que el modelo esté produciendo los resultados correctos.
2. Prueba de regresión: se utiliza para verificar que los cambios realizados en el modelo no hayan introducido errores o problemas en otras partes del sistema.
3. Prueba de carga: utilizada para evaluar el rendimiento del modelo de ML bajo cargas de trabajo pesadas y para identificar cuellos de botella en el sistema.
4. Prueba de estabilidad: realiza la ejecución continua del modelo de ML durante un período prolongado para identificar problemas de estabilidad, como la degradación del rendimiento o la inestabilidad del modelo.
5. Prueba de seguridad: permite identificar posibles vulnerabilidades de seguridad en el sistema de ML, como ataques de adversarios o fugas de información.
6. Prueba de robustez: evalúa la capacidad del modelo de ML para manejar datos atípicos o inesperados y para identificar posibles puntos débiles en el modelo.
7. Prueba de variación de datos: se utiliza para evaluar cómo el modelo de ML responde a diferentes tipos de datos y para identificar posibles problemas de generalización.
8. Prueba de regresión de datos: verifica que los cambios en los datos de entrada no hayan introducido errores o problemas en el sistema.
9. Prueba de disponibilidad: permite evaluar la capacidad del sistema de ML para manejar cargas de trabajo inesperadas y para identificar posibles puntos débiles en la infraestructura del sistema.
10. Prueba de cumplimiento normativo: se utiliza para comprobar si el sistema de ML cumple con los requisitos legales y regulatorios.

Es importante tener en cuenta que las técnicas de pruebas en servicios de ML pueden ser complejas y requieren un enfoque sistemático y riguroso. Además, deben considerar las características específicas del sistema de ML en cuestión, como el tipo de modelo, la infraestructura utilizada, la cantidad y el tipo de datos de entrenamiento, entre otros factores.

Método de investigación

Para realizar este trabajo, se utilizó el método de investigación científica analítico sintético. Este, fue utilizado para el análisis documental de la literatura que referencia el estado del arte de las técnicas de pruebas de software en servicios de Machine Learning , así como para las herramientas que actualmente se utilizan para tal fin.

Técnicas de pruebas y herramientas en servicios de Machine Learning

Las pruebas en servicios de ML pueden ser automatizadas utilizando herramientas y técnicas específicas, algunas de ellas muy comunes en el desarrollo tradicional de software. A continuación, se muestran algunas de las formas en que se pueden automatizar estas pruebas (Diaz et al., 2020):

1. Integración continua: implica la automatización de pruebas en cada etapa del ciclo de vida del software y la ejecución de pruebas de manera continua a medida que se realizan cambios en el código del modelo de ML.
2. Pruebas basadas en reglas: verifican si el modelo de ML cumple con ciertas reglas o criterios predefinidos. Pueden ser diseñadas para identificar problemas como la precisión insuficiente, la detección de datos atípicos o el incumplimiento de las normativas.
3. Pruebas basadas en datos: utilizan conjuntos de datos específicos para verificar el rendimiento del modelo de ML en diferentes escenarios. La automatización de estas pruebas puede implicar la selección automática de conjuntos de datos y la ejecución automática de pruebas en diferentes escenarios.
4. Pruebas de regresión: se utilizan para identificar cualquier regresión en el rendimiento del modelo de ML después de realizar cambios en el código o en los datos de entrenamiento. Se pueden automatizar para ejecutarse de forma continua y para notificar de inmediato a los desarrolladores si se identifican problemas.
5. Pruebas de carga: realizan la simulación de cargas de trabajo pesadas y la medición del rendimiento del modelo de ML en diferentes niveles de carga. Pueden repetirse, de forma automatizada, en intervalos regulares y para proporcionar informes automatizados sobre el rendimiento del modelo de ML.
6. Pruebas de integración: utilizadas para verificar que el modelo de ML se integra correctamente con otros componentes del sistema y pueden ejecutarse en cada iteración del desarrollo.
7. Pruebas de monitoreo: contemplan la medición continua del rendimiento del modelo de ML en tiempo real y se pueden automatizar para detectar problemas de rendimiento y notificar a los desarrolladores de inmediato.
8. Pruebas de seguridad: se utilizan para identificar posibles vulnerabilidades de seguridad en el sistema de ML y se deben ejecutar de forma regular y para identificar posibles riesgos de seguridad.

9. Pruebas de rendimiento: evalúan el rendimiento del modelo de ML en diferentes escenarios y para identificar posibles puntos débiles en el sistema. Una vez automatizadas, pueden proporcionar informes automatizados sobre el rendimiento del modelo de ML.
10. Pruebas de reglas de negocio: verifican que el modelo de ML cumple con las reglas de negocio y los requisitos del cliente.

En general, la automatización de las pruebas en servicios de ML puede ayudar a aumentar la eficiencia y la eficacia de estas, reducir los errores humanos y mejorar la calidad del modelo desarrollado, además de garantizar que el modelo funcione de manera correcta y confiable. Se deben tener en cuenta también, sus múltiples ventajas: validación de modelos, identificación de errores, mejora de la calidad del modelo, reducción del tiempo de inactividad, cumplimiento normativo, aumento de la confianza, optimización del rendimiento, mejora de la escalabilidad, reducción de costos y mejora de la innovación.

Como resultado de esta investigación, a continuación, se presentan algunas de las herramientas encargadas de la automatización de pruebas de software para modelos de ML.

1. TensorFlow versión 2.7.0: es una biblioteca de aprendizaje automático de código abierto desarrollada por Google en 2015, que se utiliza para la creación y entrenamiento de modelos de aprendizaje profundo. Proporciona una plataforma flexible para la creación de modelos de aprendizaje profundo utilizando redes neuronales, incluyendo redes convolucionales, redes recurrentes y redes generativas adversarias Generative Adversarial Network (GAN). También incluye herramientas para la implementación de modelos en diferentes plataformas de hardware, como CPU, GPU y TPU. Además de la creación y entrenamiento de modelos, proporciona herramientas para la evaluación y el despliegue de modelos, así como para la preparación y procesamiento de datos. Es una de las bibliotecas de aprendizaje automático más populares en la actualidad y se utiliza en una amplia variedad de aplicaciones, incluyendo el procesamiento de imágenes, el procesamiento de lenguaje natural, la detección de fraudes y la robótica, entre muchas otras. (TensorFlow, s. f.).
2. PyTorch versión 1.9.0: biblioteca de software de código abierto para el aprendizaje automático que se centra en la facilidad de uso y la flexibilidad, y ofrece una amplia gama de herramientas para la construcción y entrenamiento de modelos de ML. Desarrollado por el Laboratorio de Investigación de Inteligencia Artificial de Facebook (FAIR) y programado en Python, C++ y CUDA y corre para los sistemas operativos Linux, macOS y Microsoft Windows.(PyTorch, s. f.)

3. Scikit-learn versión 1.0: es una biblioteca de aprendizaje automático de código abierto escrita en Python que se utiliza para la implementación de algoritmos de aprendizaje supervisado y no supervisado, así como para la evaluación de modelos y la selección de características. Fue creado en 2007 por David Cournapeau como un proyecto de Google Summer of Code y ahora es mantenido por la comunidad de código abierto. Es una herramienta muy popular en la comunidad de aprendizaje automático debido a su facilidad de uso y a la gran cantidad de algoritmos de aprendizaje automático que incluye, como regresión lineal, Support Vector Machine (SVM), árboles de decisión, clustering y muchos más. También proporciona herramientas para la preparación de datos, como la normalización y la selección de características, y para la evaluación de modelos, como la validación cruzada y la curva de aprendizaje. Se utiliza en una amplia variedad de aplicaciones de aprendizaje automático, incluyendo análisis de imágenes, procesamiento de lenguaje natural, detección de fraudes y análisis de datos financieros (scikit-learn, s. f.).
4. Keras versión 2.6.0: es una biblioteca de redes neuronales de código abierto escrita en Python que se utiliza para el desarrollo de modelos de aprendizaje profundo. Fue desarrollada originalmente por Francois Chollet en 2015 y ahora es mantenido por la comunidad de código abierto. Es una API de alto nivel que permite a los desarrolladores crear y entrenar modelos de aprendizaje profundo con una sintaxis simple y fácil de usar. Una de las características más atractivas de Keras es su capacidad para ejecutar en diferentes plataformas de procesamiento, como CPU, GPU y TPU. Se utiliza en una variedad de aplicaciones de aprendizaje profundo, incluyendo análisis de imágenes, procesamiento de lenguaje natural y reconocimiento de voz. (Keras, s. f.).
5. Apache Spark MLlib versión 3.2.0: es una biblioteca de aprendizaje automático de código abierto desarrollada por Apache Spark. Proporciona una amplia variedad de algoritmos de aprendizaje automático escalables y distribuidos que se pueden utilizar para la creación y entrenamiento de modelos de aprendizaje automático en grandes conjuntos de datos. Incluye algoritmos para la clasificación, la regresión, el clustering, la reducción de dimensionalidad y la recomendación, entre otros. También proporciona herramientas para la preparación de datos, como la normalización y la selección de características, y para la evaluación de modelos, como la validación cruzada. Se utiliza comúnmente en aplicaciones de big data, donde se necesita procesar grandes conjuntos de datos de manera eficiente y escalable. Se integra con Apache Spark, lo que permite el procesamiento distribuido y paralelo de grandes conjuntos de datos (MLlib | Apache Spark, s. f.).
6. Kubeflow versión 1.4: es una plataforma de código abierto para el aprendizaje automático en Kubernetes, que proporciona una manera fácil de implementar, administrar y escalar pipelines de machine learning en sus entornos. Proporciona una amplia variedad de herramientas y componentes para ayudar a los desarrolladores y equipos de machine learning a crear, entrenar y desplegar modelos de manera eficiente. Algunos de los componentes más

destacados incluyen Jupyter Notebooks, Tensorflow, PyTorch, KFServing, Katib, y muchas otras herramientas de software libre. También ofrece características de automatización, como la automatización de la creación de pipelines de machine learning, la gestión de recursos y la escalabilidad (Kubeflow, s. f.).

7. MLflow versión 1.20.0: es una plataforma de código abierto para la gestión del ciclo de vida de los modelos de machine learning. Permite a los desarrolladores organizar, realizar un seguimiento y gestionar los experimentos de entrenamiento de modelos, los artefactos de modelo y los despliegues de modelos de manera eficiente en una variedad de entornos de ejecución. Se compone de varios componentes, incluyendo MLflow Tracking, que permite a los desarrolladores realizar un seguimiento de los experimentos y comparar los resultados de distintas ejecuciones de los modelos, y MLflow Models, que proporciona una forma de gestionar y hacer un seguimiento de los artefactos de modelo. Además, incluye MLflow Projects, que permite a los desarrolladores definir, empaquetar y compartir fácilmente sus proyectos. La plataforma admite múltiples lenguajes de programación, incluyendo Python, Java y R, y es compatible con una variedad de frameworks de machine learning, como TensorFlow, PyTorch, Scikit-learn y XGBoost (MLflow, s. f.).
8. Hugging Face Transformers versión 4.12.2: es una biblioteca de código abierto que proporciona una interfaz fácil de usar para utilizar modelos de lenguaje natural pre-entrenados de última generación. Permite a los desarrolladores utilizar estos modelos para tareas de procesamiento del lenguaje natural (NLP) como la clasificación de texto, la generación de texto, la traducción de idiomas, el análisis de sentimiento y muchas otras. Proporciona herramientas para ajustar los modelos pre-entrenados a conjuntos de datos específicos, lo que se conoce como ajuste fino (fine-tuning).Además de la biblioteca de modelos pre-entrenados, Hugging Face proporciona una plataforma de intercambio de modelos, llamada Hugging Face Hub, que permite a los desarrolladores compartir y descargar modelos pre-entrenados y ajustados en una variedad de lenguajes y dominios (Transformers, s. f.).

Conclusiones

El aprendizaje automático es una técnica en la que se utilizan algoritmos para permitir que las máquinas aprendan y mejoren su rendimiento en una tarea específica a medida que se les proporcionan más datos. Las ventajas del aprendizaje automático incluyen la toma de decisiones más informadas, la flexibilidad, la mejora continua, el descubrimiento de nuevos conocimientos, la reducción de errores y la adaptabilidad.

Las pruebas de software son un proceso sistemático y controlado para evaluar la calidad del software y asegurarse de que cumpla con los requisitos y expectativas del usuario. Las ventajas de las pruebas de software incluyen la mejora de

la calidad del software, la reducción de costos, el ahorro de tiempo, la mejora de la satisfacción del usuario, el cumplimiento de los estándares de calidad y seguridad, la identificación temprana de errores, la reducción del riesgo, la mejora de la reputación de la empresa, el soporte para la toma de decisiones y la mejora de la eficiencia.

En el contexto de los servicios de Machine Learning, las técnicas de pruebas son esenciales para garantizar la calidad y fiabilidad del servicio. Las pruebas pueden incluir pruebas unitarias, pruebas de integración, pruebas de sistema, pruebas de aceptación, pruebas de rendimiento y pruebas de seguridad. Cada tipo de prueba tiene su propio propósito y objetivo, pero todas son esenciales para garantizar que el servicio de Machine Learning funcione correctamente y cumpla con los requisitos y expectativas del usuario.

En la actualidad existen herramientas para realizar pruebas automatizadas en servicios de Machine Learning, algunas de ellas han sido referenciadas en este trabajo, que debe servir como punto de partida para futuras investigaciones sobre el tema.

Referencias

1. Transformers. (s. f.). Recuperado 15 de mayo de 2023, de https://huggingface.co/docs/transformers/index
2. Diaz, A. M., Casañola, Y. T., & Hidalgo, D. B. (2020). Estrategia de pruebas para organizaciones desarrolladoras de software. Testing strategy for software development organizations. Revista Cubana de Ciencias Informáticas (RCCI). Vol.14 (No.3), p83-104.
3. Keras. (s. f.). Recuperado 15 de mayo de 2023, de https://keras.io/
4. Kubeflow. (s. f.). Recuperado 15 de mayo de 2023, de https://www.kubeflow.org/
5. MLflow. (s. f.). Recuperado 15 de mayo de 2023, de https://mlflow.org/
6. MLlib | Apache Spark. (s. f.). Recuperado 15 de mayo de 2023, de https://spark.apache.org/mllib/
7. PyTorch. (s. f.). Recuperado 15 de mayo de 2023, de https://www.pytorch.org
8. Rojas, E. M. (s. f.). Machine Learning: Análisis de lenguajes de programación y herramientas para desarrollo. Revista Ibérica de Tecnologías de Informacao (RISTI). Machine Learning, Vol. E28, p 586 - 599.
9. Scikit-learn. (s. f.). Recuperado 15 de mayo de 2023, de https://scikit-learn.org/stable/
10. Serna M., E., Martínez M., R., & Tamayo O., P. (2019). A Review to Reality of Software Test Automation. Computación y Sistemas, 23(1), 169. https://doi.org/10.13053/cys-23-1-2782
11. TensorFlow. (s. f.). Recuperado 15 de mayo de 2023, de https://www.tensorflow.org
12. Test Automation University. Recuperado el 13 de mayo, 2023, de https://testautomationu.applitools.com/.
13. Huang, C. H., Li, Y., & Li, X. (2020). A survey of testing techniques for machine learning systems. Journal of Systems and Software, 166, 110550. https://doi.org/10.1016/j.jss.2020.110550

14. Leitner, A., Bayrak, A. E., Cito, J., Gall, H. C., & Avgeriou, P. (2020). Testing machine learning applications: A survey. IEEE Transactions on Software Engineering, 46(11), 1314-1340. https://doi.org/10.1109/TSE.2019.2933618
15. Zhang, X., Li, Z., & Xie, T. (2020). A survey of software testing for machine learning systems. ACM Computing Surveys, 53(6), 1-38. https://doi.org/10.1145/3417370
16. Chen, T., Gao, J., & Liu, Y. (2021). A survey on testing and debugging techniques for machine learning systems. IEEE Access, 9, 110222-110237. https://doi.org/10.1109/ACCESS.2021.3090892
17. Kaur, J., & Singh, S. (2021). An overview of testing techniques in machine learning. Journal of Intelligent & Fuzzy Systems, 41(3), 3541-3554. https://doi.org/10.3233/JIFS-201234
18. Test Automation University. Recuperado el 13 de mayo, 2023, de https://testautomationu.applitools.com/.