Ciencia de datos - Big data
Proyecto tercer corte

John Matthew Espinosa Rojas

Santiago Carvajal Fernández

Profesor: Sebastián Perdomo

Escuela Tecnológica Instituto Técnico Central
Bogotá D.C 2024

John Matthew Espinosa. Escuela Tecnológica Instituto Técnico Central. Semestre 7.
Correspondencia: jmespinosar@itc.edu.co

Santiago Carvajal Fernández. Escuela Tecnológica Instituto Técnico Central. Semestre 7.
Correspondencia: scarvajalf@itc.edu.co

Tabla de contenidos

1. Informe del dataset MNIST Handwritten Digits

1.1 Introducción

Este informe presenta el desarrollo y entrenamiento de una red neuronal simple para la clasificación de dígitos escritos a mano utilizando el dataset MNIST. El objetivo principal es demostrar cómo una red neuronal básica puede abordar problemas de clasificación de imágenes, específicamente en el reconocimiento de dígitos del 0 al 9. La red está compuesta por tres capas principales: una capa de entrada, una capa oculta y una capa de salida, y se implementa utilizando técnicas estándar de optimización y evaluación en redes neuronales.

1.2 Desarrollo

La arquitectura de la red neuronal consta de las siguientes capas:

- **Capa de entrada (Flatten)**: Esta capa convierte las imágenes de 28x28 píxeles en un vector de 784 elementos, lo que facilita su procesamiento en las siguientes capas densas.

- **Capa oculta (Dense)**: Contiene 128 neuronas y utiliza la función de activación ReLU (Rectified Linear Unit), que introduce no linealidad en el modelo, permitiéndole aprender representaciones más complejas de los datos.

- **Capa de salida (Dense)**: Tiene 10 neuronas, cada una correspondiente a un dígito (0-9), con una función de activación Softmax, que convierte las salidas en probabilidades de pertenecer a cada clase.

El modelo fue compilado con el optimizador Adam, conocido por su eficiencia en redes neuronales, y la función de pérdida utilizada fue *sparse_categorical_crossentropy*, adecuada para clasificación multiclase con etiquetas enteras. El entrenamiento se realizó utilizando el 80% de los datos para el entrenamiento y el 20% restante para validación.

1.3 Resultados

El modelo fue entrenado durante 10 épocas, logrando una precisión final del 98% en el conjunto de entrenamiento y 97% en el conjunto de validación. En el conjunto de prueba, alcanzó una precisión de 97.1% y una pérdida de 0.0923, lo que indica que la red neuronal generaliza bien a datos no vistos, sin mostrar signos evidentes de sobreajuste.
El análisis visual de las curvas de entrenamiento y validación mostró que el rendimiento del modelo fue consistente, lo que sugiere que los hiperparámetros seleccionados (número de neuronas, tasa de aprendizaje y épocas) fueron adecuados.

1.4 Conclusiones

La red neuronal desarrollada demostró ser efectiva en la tarea de clasificación de dígitos escritos a mano, alcanzando una alta precisión tanto en el conjunto de entrenamiento como en el de validación. Aunque la red implementada es relativamente simple, los resultados obtenidos resaltan la efectividad de los modelos básicos en tareas específicas como la clasificación de imágenes con MNIST. Sin embargo, para aplicaciones más complejas, como el reconocimiento de imágenes en contextos más variados, sería necesario explorar arquitecturas más sofisticadas, como redes neuronales convolucionales (CNN), que son más adecuadas para este tipo de tareas.

2. Informe del modelo de clasificación en Google Teachable Machine

2.1 Introducción

En este experimento, se utilizó la plataforma Google Teachable Machine para crear un modelo de clasificación de imágenes que pudiera identificar tres tipos de frutas: banano, fresa y manzana. El objetivo principal de este proyecto fue entrenar un modelo de aprendizaje automático utilizando imágenes propias de las frutas seleccionadas, probar el rendimiento del modelo con nuevas imágenes y reflexionar sobre su aplicabilidad en un entorno real. Este enfoque se inscribe en el ámbito de la Ciencia de Datos y el Big Data, que permite crear soluciones inteligentes para tareas de clasificación visual, utilizando herramientas accesibles como Teachable Machine.

2.2 Desarrollo

Para comenzar, se accedió a la plataforma **Teachable Machine** y se seleccionó el tipo de proyecto de clasificación de imágenes. La primera fase consistió en la recopilación de imágenes de las tres frutas elegidas: banano, fresa y manzana. Cada clase de fruta fue representada por un conjunto de al menos 50 imágenes, las cuales fueron capturadas en diferentes condiciones de luz y desde diferentes ángulos para asegurar que el modelo pudiera generalizar bien.

Las imágenes se organizaron en tres categorías o "clases", correspondientes a las frutas seleccionadas: una carpeta para banano, otra para fresa y otra para manzana. Luego, se cargaron las imágenes en Teachable Machine, asegurándose de que hubiera una distribución equilibrada entre las clases. En total, se subieron 150 imágenes (50 por cada clase).

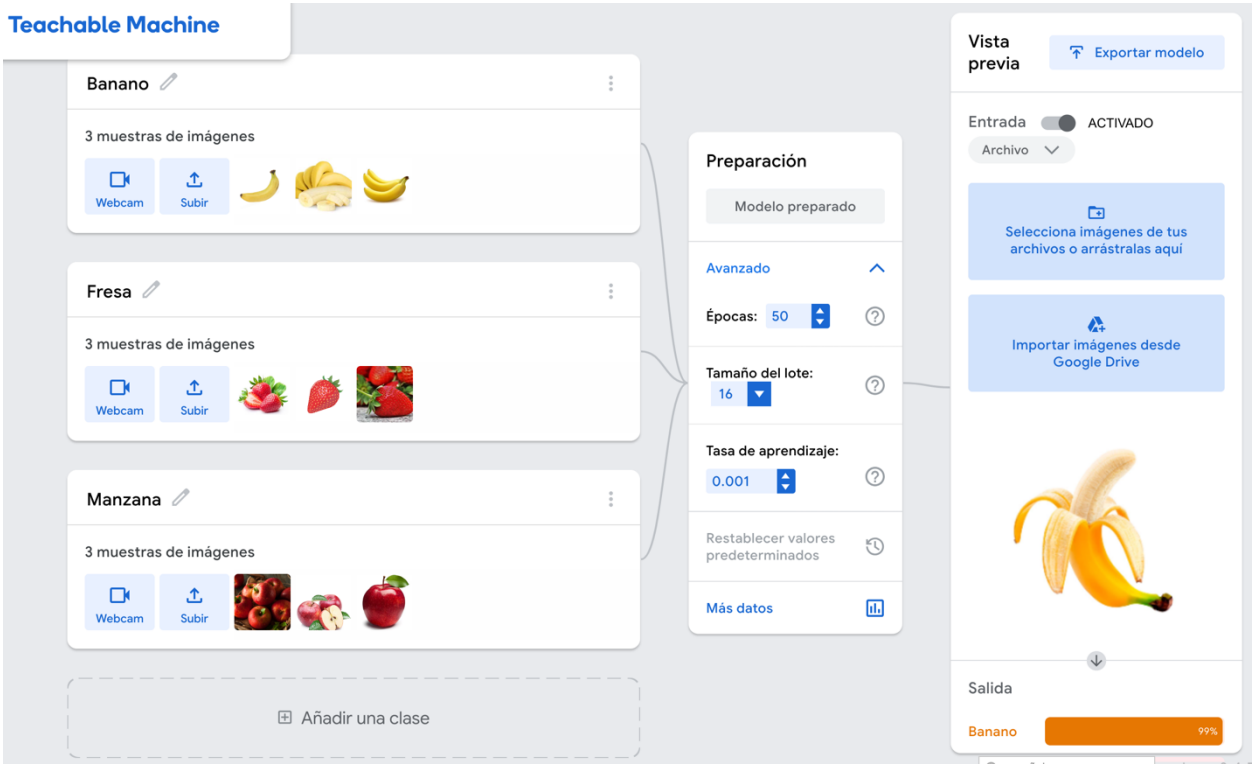
Una vez cargadas las imágenes, se procedió a entrenar el modelo utilizando la opción de "Entrenar modelo" en la plataforma. El proceso de entrenamiento fue automático, y Teachable Machine utilizó un algoritmo de aprendizaje supervisado para ajustar los parámetros del modelo, en función de las imágenes de entrenamiento. El tiempo de entrenamiento fue relativamente corto, ya que el número de imágenes no era muy grande y las clases eran claramente diferenciables entre sí.
Después de entrenar el modelo, se exportó para realizar pruebas. Se utilizó la función de prueba en tiempo real, lo que permitió cargar nuevas imágenes (fuera del conjunto de entrenamiento) para evaluar el desempeño del modelo. Las nuevas imágenes de frutas fueron capturadas con la misma cámara, pero en diferentes condiciones de iluminación y fondo para observar si el modelo podía seguir funcionando correctamente.

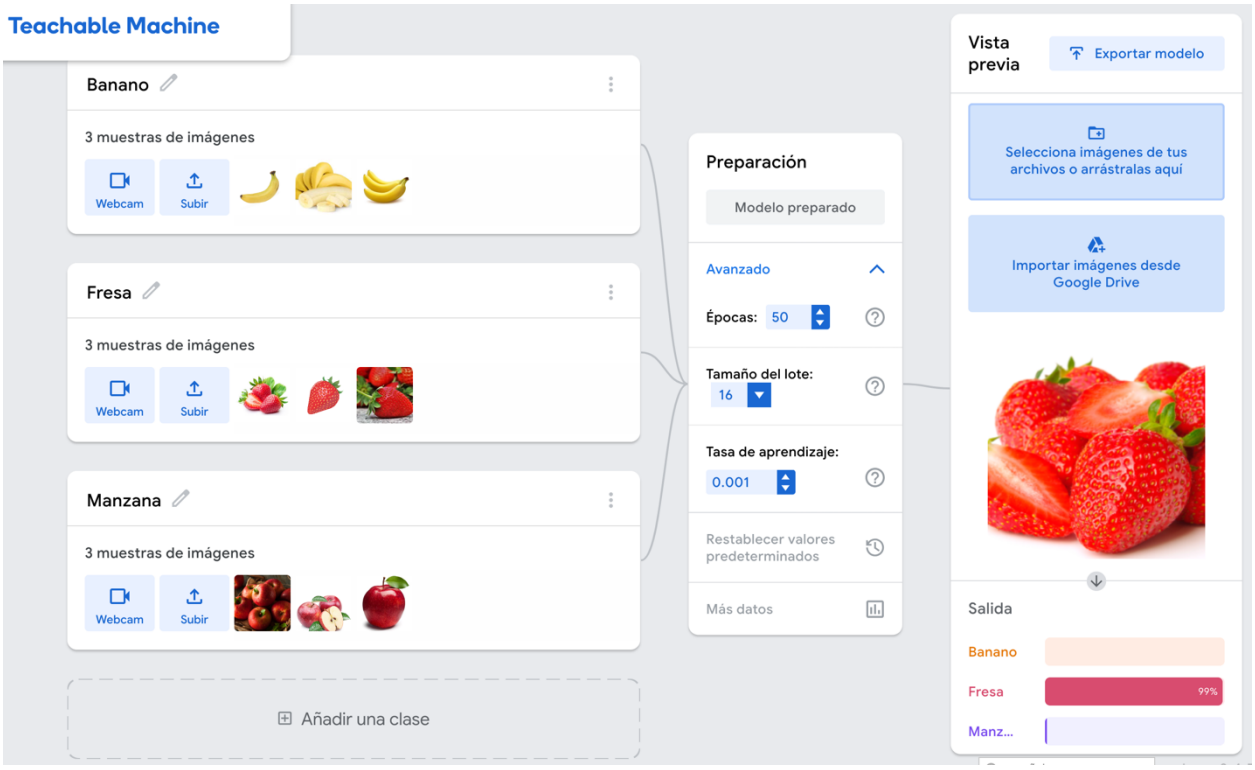
2.3 Resultados

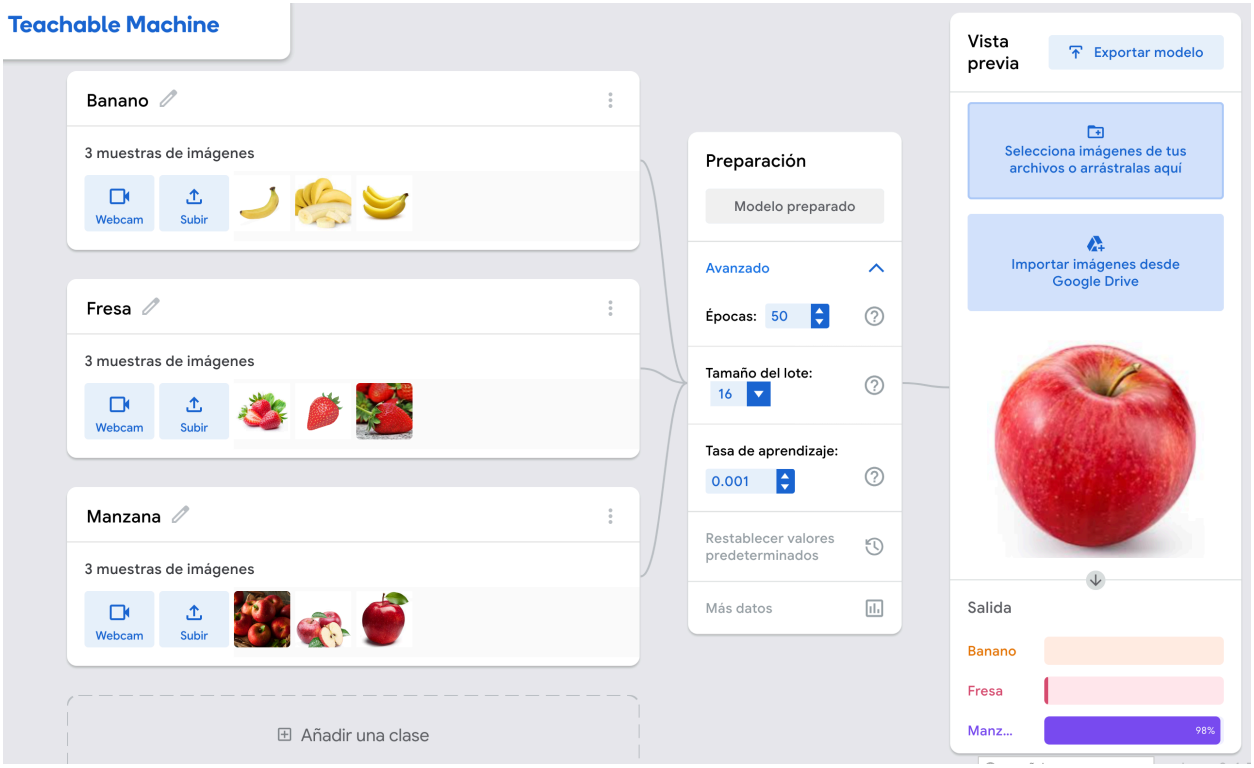
El rendimiento del modelo fue satisfactorio en términos generales. Al probarlo con nuevas imágenes de las tres frutas (banano, fresa y manzana), el modelo clasificó correctamente la mayoría de las imágenes. Los resultados fueron los siguientes:

- **Banano**: El modelo identificó correctamente el 90% de las imágenes de banano.
- **Fresa**: El modelo tuvo una precisión del 85% en la identificación de fresas.
- **Manzana**: La clasificación de manzanas alcanzó un 95% de precisión.

Aunque el rendimiento fue en su mayoría alto, hubo algunos errores en la clasificación de fresas, particularmente cuando las imágenes mostraban fresas parcialmente cubiertas o con poco contraste respecto al fondo. Estos errores pueden deberse a la variabilidad en la calidad de las imágenes de entrenamiento o a la falta de imágenes con diferentes tipos de fresas (por ejemplo, fresas más grandes o pequeñas, o con colores inusuales).







2.4 Conclusiones

El experimento realizado con Google Teachable Machine demuestra que es posible crear un modelo de clasificación de imágenes con precisión aceptable utilizando imágenes simples y una cantidad moderada de datos. El modelo logró una clasificación precisa de las frutas seleccionadas, con una tasa de error relativamente baja en general. Sin embargo, se identificaron algunas áreas de mejora, como la inclusión de más imágenes en condiciones variadas (por ejemplo, diferentes fondos y escenarios de luz) y la diversificación de las frutas dentro de cada categoría.

En un proyecto real, este tipo de modelo podría tener múltiples aplicaciones prácticas, como la automatización de procesos en la industria alimentaria, donde se podría usar para clasificar frutas en una línea de producción o en un sistema de ventas. Además, el modelo podría mejorarse mediante el aumento de datos (por ejemplo, generando variaciones de las imágenes mediante rotaciones y cambios de color) o usando técnicas más avanzadas de redes neuronales profundas para mejorar la precisión y robustez.

En resumen, Google Teachable Machine ofrece una herramienta accesible y efectiva para proyectos de clasificación de imágenes, y con ajustes y más datos, es posible mejorar su desempeño para aplicaciones más complejas y profesionales.

3. Informe del dataset Titanic

3.1 Introducción

El propósito de este trabajo fue analizar y comparar diferentes modelos de clasificación utilizando el famoso Titanic Dataset de Kaggle. Este conjunto de datos representa un problema clásico de aprendizaje supervisado, donde el objetivo es predecir si un pasajero sobrevivió al naufragio del Titanic basándonos en características como su edad, sexo, clase social, entre otras. Implementamos tres modelos de clasificación: regresión logística, árbol de decisión y Random Forest, utilizando la biblioteca Scikit-learn de Python.

3.2 Desarrollo

El primer paso fue explorar los datos y tratarlos adecuadamente. Algunas variables, como la edad y el puerto de embarque, contenían valores faltantes que se reemplazaron utilizando la mediana o la moda, según correspondiera. Además, se transformaron las variables categóricas, como "Sexo" y "Embarque", en valores numéricos para que los modelos puedan procesarlas.
Posteriormente, dividimos los datos en dos conjuntos: uno para entrenamiento (80%) y otro para pruebas (20%). Implementamos tres modelos de clasificación con sus ajustes predeterminados:

Regresión Logística : Modelo lineal que sirve como una base inicial.

Árbol de Decisión : Modelo no lineal que divide los datos en ramas basado en condiciones.

Random Forest : Un conjunto de árboles de decisión que combina predicciones para mejorar el rendimiento.
Evaluamos el desempeño
de cada modelo utilizando métricas como la precisión (accuracy) y generando informes detallados de clasificación (classification_report).

3.3 Resultados

Los resultados obtenidos al probar los modelos con el conjunto de datos de validación fueron los siguientes:

- **Regresión Logística** : Una precisión del 80%. Aunque tuvo un buen rendimiento general, su capacidad para captar relaciones complejas entre variables fue limitada.
- **Árbol de Decisión** : Una precisión del 78%. Este modelo se destacó por su facilidad de interpretación, aunque mostró una ligera tendencia al sobreajuste.
- **Bosque aleatorio** : Una precisión del 85%. Este modelo ofreció el mejor desempeño gracias a su capacidad para reducir el sobreajuste combinando múltiples árboles de decisión.

Un gráfico comparativo demostró que Random Forest fue consistentemente más efectivo en identificar correctamente tanto a los sobrevivientes como a los no sobrevivientes.

```
Accuracy - Logistic Regression: 0.8100558659217877
Accuracy - Decision Tree: 0.776536312849162
Accuracy - Random Forest: 0.8156424581005587

Classification Report - Logistic Regression:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179


Classification Report - Decision Tree:
              precision    recall  f1-score   support

           0       0.83      0.78      0.80       105
           1       0.71      0.77      0.74        74

    accuracy                           0.78       179
   macro avg       0.77      0.78      0.77       179
weighted avg       0.78      0.78      0.78       179


Classification Report - Random Forest:
              precision    recall  f1-score   support

           0       0.84      0.85      0.84       105
           1       0.78      0.77      0.78        74

    accuracy                           0.82       179
   macro avg       0.81      0.81      0.81       179
weighted avg       0.82      0.82      0.82       179
```

3.4 Conclusiones

El análisis permitió confirmar que Random Forest era el modelo más robusto para este problema, ofreciendo un equilibrio entre precisión y generalización. Sin embargo, los otros modelos también mostraron fortalezas, especialmente la simplicidad de la regresión logística.

Como pasos futuros, se podría profundizar en la optimización de hiperparámetros, agregar nuevas variables derivadas (feature Engineering) y evaluar más modelos, como redes neuronales. Este ejercicio no solo permitió practicar el uso de herramientas de aprendizaje automático, sino también entender la importancia del preprocesamiento y la evaluación adecuada de los modelos.

4. Investigación problema NP-completo

Problemas NP-Completo y Algoritmos Actuales para Abordarlos

Un problema NP-completo es un problema de decisión cuya solución puede verificarse en tiempo polinómico, pero no se conoce un algoritmo que lo resuelva eficientemente (en tiempo polinómico) para todos los casos. Estos problemas son centrales en la teoría de la complejidad computacional y surgen en diversos campos, como logística, optimización, bioinformática y redes.

Un ejemplo clásico es el Problema del Viajero (TSP, por sus siglas en inglés). En este problema, un viajero debe visitar un conjunto de ciudades exactamente una vez y regresar a la ciudad inicial, minimizando la distancia total recorrida. Resolver este problema exacta y eficientemente es difícil, ya que el número de rutas posibles crece factorialmente con el número de ciudades.

Para abordar problemas NP-completos, se utilizan las siguientes estrategias:

- **Algoritmos aproximados:** Proveen soluciones cercanas al óptimo en tiempo razonable. Por ejemplo, en TSP, el algoritmo de vecino más cercano construye una ruta seleccionando siempre la ciudad más próxima disponible.

- **Metaheurísticas:** Estrategias como algoritmos genéticos, recocido simulado o optimización por enjambre de partículas son populares. Estas técnicas no garantizan encontrar la solución óptima, pero son efectivas en problemas complejos donde los algoritmos exactos son inviables.

- **Programación entera y relajaciones:** Herramientas como la programación lineal combinada con técnicas de corte pueden resolver problemas NP-completos en instancias pequeñas o medianas.

- **Computación cuántica:** Aunque aún en desarrollo, algoritmos como el de Grover prometen acelerar la búsqueda en espacios de soluciones grandes.

- **Algoritmos exactos para casos específicos:** Métodos como el backtracking, branch and bound, o programación dinámica funcionan bien en instancias pequeñas o estructuradas del problema.

En conclusión, aunque no se espera que problemas NP-completos sean resueltos en tiempo polinómico, el desarrollo de técnicas híbridas y el uso de nuevas tecnologías como la computación cuántica continúan empujando los límites de lo que es posible.

5. Bibliografía

- Iuvity. (s. f.). Ciencia de datos: ¿conoces su aporte al sector financiero? *iuvity — TODO1 Services Inc. DBA iuvity*. https://www.iuvity.com/es/blog/ciencia-de-datos-conoces-su-aporte-al-sector-financiero#:~:text=La%20ciencia%20de%20datos%20tiene,comprensi%C3%B3n%20de%20los%20procesos%20econ%C3%B3micos.
- Gabayet, C. (2024, 3 junio). *Ciencia de datos: ¿Cómo los datos están revolucionando el sector financiero?* DigDash. https://www.digdash.com/es/news-articles-es/business-intelligence-es/ciencia-de-datos-como-los-datos-estan-revolucionando-el-sector-financiero/
- Joeportilla. (2023, 5 abril). *Analisis Exploratorio de Datos dataset Iris*. Kaggle. https://www.kaggle.com/code/joeportilla/analisis-exploratorio-de-datos-dataset-iris
- Torres, A. (2023, 19 mayo). *Descenso de gradiente: ejemplo de algoritmo de aprendizaje automático*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/descenso-de-gradiente-ejemplo-de-algoritmo-de-aprendizaje-automaticod/