Universidad Panamericana
Facultad de Ingeniería

Clase de Inteligencia Artificial
Proyecto del 3er Parcial
Mayo 2023

Redes Neuronales MLP: Breast Cancer

Sara Rocio Miranda Mateos

(0244643)
Dania Barbara Venegas Bofill (0241911)
Ivanna Valentina Tinedo Guerrero (0239947)
Christian Matos Basaldúa (0231673)

31 Mayo, 2023

Redes Neuronales MLP: Breast Cancer

Sara Rocio Miranda Mateos (0244643)
Dania Barbara Venegas Bofill (0241911)
Ivanna Valentina Tinedo Guerrero (0239947)
Christian Matos Basaldúa (0231673)

May 31, 2023

1 Instrucciones para ejecutar el código

1. Instalación de las bibliotecas necesarias: pandas, matplotlib, sklearn y seaborn.
2. Preparación de los datos en un formato adecuado para el análisis.
3. Carga de los datos desde un archivo en el directorio correcto o proporcionando una ruta de acceso completa al archivo en el código.
4. Ejecución del código que importará las bibliotecas necesarias y definirá varias funciones y clases para el análisis de datos y el entrenamiento de redes neuronales.

2 Definición del problema

Nosotros investigamos en 3 trabajos de investigacion, a continuación explicaremos que vimos en los distintos documentos.

- A Neural Network Model for Prognostic Prediction de W. Nick Street
  - Problemas: En este primer documento vimos que el autor se enfrentó al desafío de generar pronósticos con datos censurados, es decir, resultados aún desconocidos que son fundamentales para la toma de decisiones clínicas y tratamientos futuros. Estos pronósticos deben ser precisos y confiables. Además, se encontró con la dificultad de clasificar problemas no discretos, donde la tarea de pronóstico no se ajusta fácilmente a los enfoques convencionales de clasificación o aproximación de funciones. No existe un punto de corte claro para determinar la recurrencia en pacientes. Los datos censurados implican que solo se conoce el tiempo de recurrencia en ciertos subconjuntos de casos. Para abordar estos desafíos, se propone combinar enfoques de estadística no paramétrica y arquitecturas de redes neuronales, con el fin de aprovechar toda la información disponible. La integración óptima de estos datos censurados y los tiempos de recurrencia representa un reto para mejorar la precisión y flexibilidad de los modelos de pronóstico.

- Relevancia: El autor aborda un desafío médico crucial al tratar el pronóstico de enfermedades, particularmente el cáncer de mama. La precisión en la predicción del tiempo de recurrencia después de la cirugía es fundamental para la toma de decisiones clínicas, ya que influye en las opciones de tratamiento y la intensidad de la terapia. En este contexto, el autor propone una novedosa técnica de codificación de datos censurados utilizando una red neuronal artificial. Al combinar ideas de estadística no paramétrica, el enfoque propuesto mejora la precisión y utilidad de los resultados en comparación con métodos existentes. Este artículo busca generar modelos de pronóstico que sean altamente relevantes en un entorno clínico, brindando a los médicos decisiones más fundamentadas y a los pacientes un tratamiento más personalizado, al tiempo que evita intervenciones innecesarias.

- Exploiting Unlabel Data in Ensemble Methods de Kristin P. Benett

  - Problemas: Siendo el desafio principal fue el cómo utilizar de manera efectiva los datos no etiquetas en el aprendizaje semi-supervisado, tuvo que encontrar la forma adecuada de derivar un margen apropiado para estos datos y adaptas los algoritmos de ensamble existentes, por otro lado, su objetivo era desarrollar un enfoque de aprendizaje semi-supervisado que pudiera funcionar con cuaquiermétodo de clasificacion que pudiera funcionar con cualquier metodo de clasificacion sensible al costo, lo cual requeria adaptar los algoritmos de ASSAMBLE existentes para que así fuesen compatibles ocn el aprendizaje semi-supervisado y puedieran integrar datos no etiquetados de manera efectiva. Además otro problema era determinar cómo escalar el algoritmo ASSEMBLE para conjuntos de datos más grandes para poder evaluar su rendimiento en problemas más complejos.
  - Relevancia: El autor se centra en un problema fundamental del aprendizaje automático y presenta un enfoque novedoso y efectivo para el aprendizaje semi-supervisado. Este enfoque permite aprovechar tanto datos etiquetados como no etiquetados, lo cual es una ventaja significativa. Además, el artículo ofrece perspectivas futuras para ampliar y mejorar el método propuesto, lo cual indica un potencial prometedor. Asimismo, se proporciona evidencia empírica que demuestra la eficacia de ASSEMBLE en comparación con otros algoritmos, lo cual respalda su validez y relevancia en el campo del aprendizaje automático.

- An Evolutionary Artificial Neural Networks Approach For Breast Cancer Diagnosis

  - Problemas: Durante la investigación, se identificaron varios problemas que merecen atención. Uno de ellos es la percepción y confianza en las máquinas, es decir, cómo los usuarios perciben y confían en los resultados generados por las redes neuronales. Además, se destacó la importancia de utilizar adecuadamente las redes neuronales artificiales (ANNs) como herramientas de apoyo, teniendo en cuenta sus limitaciones y considerando su papel complementario a la experiencia humana. También se observó que el algoritmo de retropropagación, ampliamente utilizado en el entrenamiento de redes neuronales, tiene limitaciones y puede no ser adecuado en todos los casos. Por último, se mencionó el alto costo computacional asociado a algunos enfoques de redes neuronales, lo cual puede ser una limitación en términos de tiempo y recursos requeridos. Estos problemas resaltan la importancia de abordar de manera crítica y cuidadosa los desafíos asociados con el uso de las ANNs en diversos contextos.
  - Relevancia: Este archivo es relevante e interesante debido a su aplicación médica de gran importancia. En él, se utilizan algoritmos evolutivos para abordar el diagnóstico del cáncer de mama, lo cual representa un avance significativo en el campo de la inteligencia artificial

aplicada a la medicina. Además, el artículo realiza una comparación exhaustiva con enfoques existentes, lo que resalta la contribución y el valor añadido de la propuesta presentada. Uno de los aspectos destacados es su potencial para mejorar la precisión y eficiencia en el diagnóstico de enfermedades, lo cual puede tener un impacto positivo en la atención médica y en la toma de decisiones clínicas. Además, el artículo proporciona nuevas perspectivas y avances en el campo, lo que fomenta el desarrollo y la investigación continua en la aplicación de la inteligencia artificial en medicina. En resumen, este archivo es relevante debido a su aplicación médica importante, su utilización de algoritmos evolutivos, su comparación con enfoques existentes y su contribución a la confianza en las máquinas en el ámbito del diagnóstico del cáncer de mama, así como su potencial para mejorar la precisión y eficiencia en el diagnóstico de enfermedades.

Pudimos observar al menos tres aspectos en los que estos problemas y sus resoluciones aportan de manera positiva a la humanidad:

1. Pronóstico preciso de enfermedades: Obtener un pronóstico preciso es fundamental para los médicos, ya que les proporciona información crucial sobre el tiempo de recurrencia de una enfermedad. Esto les permite tomar decisiones informadas y determinar el tratamiento más adecuado para cada paciente, lo que a su vez conduce a un enfoque de tratamiento personalizado y mejorado.
2. Mejora en el diagnóstico médico: La utilización de redes neuronales en el diagnóstico de enfermedades, como el cáncer de mama, supera los desafíos asociados con la percepción y la confianza en las máquinas. Además, gracias a una mayor precisión en el diagnóstico, se logra una detección temprana de enfermedades, lo que conlleva tratamientos más efectivos y la posibilidad de salvar vidas.
3. Avance en la medicina personalizada: Los avances en el pronóstico y diagnóstico de enfermedades nos permiten un enfoque más personalizado en el cuidado de los pacientes. Mediante el uso de técnicas avanzadas de aprendizaje automático y el análisis de datos específicos de cada paciente, podemos identificar patrones y características individuales que nos ayudan a comprender mejor las necesidades de cada persona. Esto se traduce en una medicina más personalizada y en la mejora de los resultados en salud.

También analizamos que existen varios aspecto negativos que afectan a la humanidad.

Entre los más destacados se encuentran:

1. Desconfianza y resistencia hacia las máquinas: En muchas ocasiones, los usuarios no confían en los resultados generados por las redes neuronales o no comprenden cómo se obtienen, lo que genera desconfianza. Esta falta de confianza dificulta significativamente la adopción de tecnologías de inteligencia artificial en el ámbito médico.
2. Limitaciones del algoritmo de retropropagación: El algoritmo utilizado en el entrenamiento de redes neuronales presenta limitaciones que afectan la eficiencia y precisión de los modelos de aprendizaje automático. Estas limitaciones deben ser abordadas de manera adecuada para evitar resultados subóptimos y dificultades en el avance de la inteligencia artificial aplicada a la medicina.
3. Alto costo computacional: Algunos enfoques de investigación requieren un alto costo computacional, lo cual limita la accesibilidad y la adopción generalizada de estas tecnologías. La

falta de infraestructura adecuada obstaculiza el progreso y la aplicabilidad de la inteligencia artificial en el campo médico.

En base a nuestras opiniones, la implementación de estas tecnologías genera implicaciones éticas que no deben ser ignoradas. Además de la desconfianza que ya existe en las personas con respecto a los resultados generados por las máquinas y la falta de transparencia, existe el riesgo de sesgos y discriminación en los algoritmos. También se plantea la preocupación por la privación y seguridad de la información delicada de los pacientes, así como las cuestiones relacionadas con la responsabilidad y la toma de decisiones en casos de errores o conflicto éticos.

Es crucial abordar estos temas de manera seria y rigurosa. Se necesitan políticas y marcos regulatorios sólidos que fomenten la transparencia, equidad, privacidad y responsabilidad de la inteligencia artificial en la medicina.

3 Conjunto de datos

3.1 Sobre los datos

Los datos usados para este modelo de red neuronal corresponden a un estudio de 569 diagnósticos de cáncer de mama en hospitales de Universidad de Wisconsin hasta noviembre de 1995. Los datos fueron computados a partit de imágenes digitalizadas del procedimiento de diagnóstico Fine-Needle Aspiration (FNA). Los datos fueron donados por Nick Street al Machine Learning Repository de UCI. Sus creadores son:

1. Dr. William H. Wolberg, General Surgery Dept.
   University of Wisconsin, Clinical Sciences Center
   Madison, WI 53792
   wolberg '@' eagle.surgery.wisc.edu
2. W. Nick Street, Computer Sciences Dept.
   University of Wisconsin, 1210 West Dayton St.
   Madison, WI 53706
   street '@' cs.wisc.edu 608-262-6619
3. Olvi L. Mangasarian, Computer Sciences Dept.
   University of Wisconsin, 1210 West Dayton St.
   Madison, WI 53706
   olvi '@' cs.wisc.edu

3.2 Atributos de datos

Los datos contienen un número de identificación junto con 11 otros rubros:

1. Número de identificación
2. Diagnóstico (M = maligno, B = benigno)

Se calculan diez características de valor real para cada núcleo celular:

1. Radio (promedio de distancias desde el centro hasta los puntos en el perímetro)
2. Textura (desviación estándar de los valores en escala de grises)
3. Perímetro
4. Área
5. Suavidad (variación local en las longitudes de radio)
6. Compacidad ($\frac{Perímetro^2}{Área} - 1.0$)
7. Concavidad (gravedad de las porciones cóncavas del contorno)
8. Puntos cóncavos (número de porciones cóncavas del contorno)
9. Simetría
10. Dimensión fractal ("aproximación de la línea costera" - 1)

Table 1: Análisis de cinco números

	clump_thickness	cell_size	cell_shape	marginal_adhesion	epithelial_size	bland_chromatin	normal_nucleoli	mitoses
count	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000	699.000000
mean	4.417740	3.134478	3.207439	2.806867	3.216023	3.437768	2.866953	1.589413
std	2.815741	3.051459	2.971913	2.855379	2.214300	2.438364	3.053634	1.715078
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	2.000000	1.000000	1.000000	1.000000	2.000000	2.000000	1.000000	1.000000
50%	4.000000	1.000000	1.000000	1.000000	2.000000	3.000000	1.000000	1.000000
75%	6.000000	5.000000	5.000000	4.000000	4.000000	5.000000	4.000000	1.000000
max	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000

Table 2: Primeras cinco filas de los datos

	sample_code	clump_thickness	cell_size	cell_shape	marginal_adhesion	epithelial_size	bare_nuclei	bland_chromatin	normal_nucleoli	mitoses	classd
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	

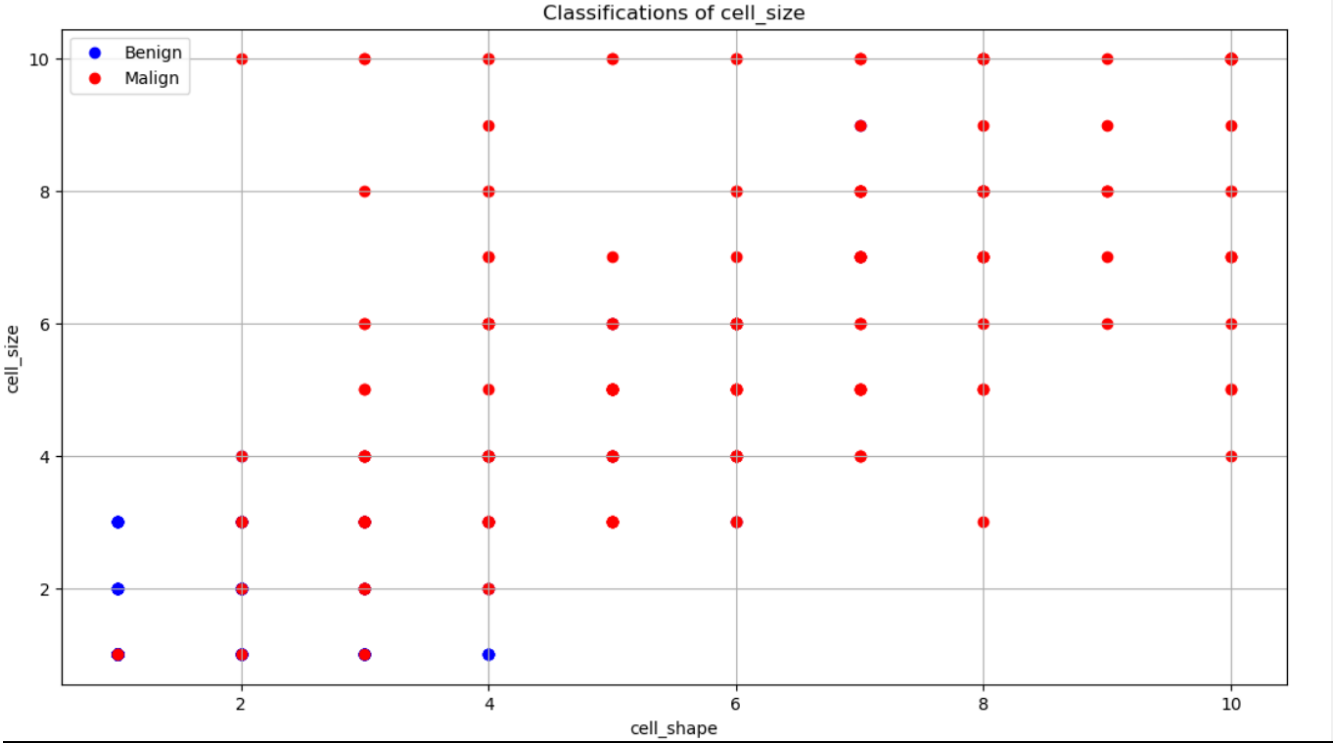
3.3 Cómo se cargan los datos

1. Carga de los datos desde un archivo ”breast-cancer-wisconsin.data” en el mismo directorio que el archivo de código o proporcionando la ruta de acceso completa al archivo en la función ‘read_csv‘.
2. Visualización del número de observaciones y características en el dataset y mostrar en las primeras filas del dataset utilizando el método ‘head()‘.
3. Agrupación de las observaciones del dataframe por la columna ‘’classd’‘ y visualización del recuento de observaciones para cada clase o especie.

Procedimiento para importar datos

- Debemos tener seguro el archivo de datos ”breast-cancer-wisconsin.data” en el mismo directorio que el archivo de código o proporcionar la ruta de acceso completa al archivo en la función ‘read_csv‘.
- Asegurarnos de que la biblioteca ‘pandas‘ esté instalada.
- Ejecutar el código donde la función ‘pd.read_csv()‘ cargará el archivo de datos en un dataframe de pandas llamado ‘dataset‘.
- Luego, el código debe imprimir el número de observaciones y características en el dataset y mostrar en las primeras filas del dataset utilizando el método ‘head()‘.
- Después de tener el dataframe ‘dataset‘ creado significa que se debió haber ejecutado el código anterior que carga el archivo de datos en el dataframe.
- Luego se ejecuta el código donde la línea ‘dataset.groupby(’classd’).size()‘ agrupa las observaciones del dataframe por la columna ‘’classd’‘ y muestra el recuento de observaciones para cada clase o especie. La línea ‘dataset.describe()‘ calcula varias estadísticas descriptivas para cada columna del dataframe y las muestra, incluyendo el recuento, la media, la desviación estándar, los valores mínimos, los percentiles 25, 50 y 75, y los valores máximos.
- Si todo corre y está el dataframe ‘dataset‘ correctamente cargado, se debe ver los resultados impresos en la salida. Esto incluyendo el recuento de observaciones para cada clase y las estadísticas descriptivas para cada columna del dataset.

Figure 1: Diagnósticos de cáncer de mama de acuerdo con el tamaño y la forma de las células



- Luego tener seguro de haber ejecutado el código anterior para cargar el archivo de datos en el dataframe antes de ejecutar estos fragmentos de código.
- También asegurarse de tener el dataframe ‘dataset‘ creado anteriormente en tu entorno de Python. Esto significa que ejecutamos el código anterior que carga el archivo de datos en el dataframe.
- Nos aseguramos de tener la biblioteca ‘matplotlib‘ instalada.
- Ejecutamos el código que creará una figura y un eje (subplot) utilizando ‘plt.subplots()‘. Ajustará el tamaño de la figura utilizando ‘fig.set_size_inches()‘.
- Luego, trazará un diagrama de dispersión (scatter plot) de las características ‘’cell_shape’‘ y ‘’cell_size’‘ del dataframe ‘dataset‘. Los puntos pertenecientes a la clase ‘’Benign’‘ se mostrarán en azul y los puntos pertenecientes a la clase ‘’Malign’‘ se mostrarán en rojo. También se generarán etiquetas, títulos y se mostrará una leyenda en el gráfico.
- Si corre bien, teniendo el dataframe ‘dataset‘ correctamente cargado y la biblioteca ‘matplotlib‘ instalada, deberías ver el gráfico de dispersión que muestra la relación entre las características ‘’cell_shape’‘ y ‘’cell_size’‘ del dataset, con los puntos coloreados según su clase.

4 Clasificación

4.1 Red Neuronal de base

Creación de una instancia de ‘MLPClassifier‘ llamada ‘mlp_clf‘ con los parámetros especificados y entrenamiento del modelo utilizando el conjunto de datos de entrenamiento.

– Nos aseguramos de tener la biblioteca ‘sklearn.neural_network‘ importada correctamente.
– También de tener el conjunto de datos y las características adecuadas. Esto significa que se deben haber ejecutado los fragmentos de código anteriores para cargar el conjunto de datos, dividirlo en conjuntos de entrenamiento y prueba, y escalar los datos si es necesario.

Entrenamiento

– Nos percatamos de tener la biblioteca ‘sklearn.neural_network‘ importada correctamente. Esta biblioteca se utiliza para crear el clasificador MLP (Multilayer Perceptron).
– Ejecutamos el código donde se creará una instancia de ‘MLPClassifier‘ llamada ‘mlp_clf‘ con los parámetros especificados:
  * ‘hidden_layer_sizes=(2)‘: Define la estructura de la red neuronal, con una capa oculta que contiene 2 neuronas.
  * ‘max_iter = 2000‘: Especifica el número máximo de iteraciones para el solver (optimizador) durante el entrenamiento del modelo.
  * ‘learning_rate_init‘: El ritmo inicial de aprendizaje que se va a usar. Controla el tamaño de los escalones al actualizar el peso.
  * ‘activation=’relu’‘: Define la función de activación utilizada en las neuronas, en este caso, la función de activación ReLU.
  * ‘solver=’adam’‘: Especifica el optimizador utilizado para ajustar los pesos de la red neuronal, en este caso, el optimizador Adam.
– Si todo corre bien y los datos son adecuados y la biblioteca ‘sklearn.neural_network‘ importada, se creará el clasificador MLP con los parámetros especificados en ‘mlp_clf‘.

Evaluación

Procedimientos

– Verificar que tengan las bibliotecas ‘sklearn.metrics‘ y ‘sklearn.model_selection‘ importadas correctamente. Estas bibliotecas se utilizan para calcular y mostrar las métricas de evaluación del modelo.
– Verificar que se tenga el conjunto de datos y el modelo ‘mlp_clf‘ correctamente definidos y ajustados en tu entorno de Python. Esto significa que se debe haber ejecutado los fragmentos de código anteriores para cargar los datos, dividirlos en conjuntos de entrenamiento y prueba, escalarlos y crear el modelo MLP.
– Ejecutamos el código donde se calcularán y mostrarán varias métricas de evaluación del modelo:
  * ‘accuracy_score(y_test, y_pred)‘: Calcula la precisión del modelo al predecir las etiquetas de clase para el conjunto de prueba ‘y_test‘. La precisión es la proporción de predicciones correctas sobre el total de predicciones.
  * ‘classification_report(y_test, y_pred)‘: Calcula y muestra un informe detallado que incluye la precisión, la recuperación, el puntaje F1 y el soporte para cada clase en comparación con las etiquetas de clase reales en ‘y_test‘.
– Si todo va bien y tenemos los datos y el modelo adecuados, se deberían ver los resultados impresos en la salida. Esto incluirá la precisión del modelo y el informe de clasificación que muestra la precisión, la recuperación, el puntaje F1 y el soporte para cada clase.

– Luego nos aseguramos de tener la biblioteca ‘matplotlib.pyplot‘ importada correctamente. Esta biblioteca se utiliza para visualizar la curva de pérdida del modelo.

– Nos aseguramos de haber ajustado el modelo ‘mlp_clf‘ esto significa que debes haber ejecutado los fragmentos de código anteriores para cargar los datos, dividirlos en conjuntos de entrenamiento y prueba, escalarlos y crear el modelo MLP.

– Ejecutamos el código y se trazará la curva de pérdida del modelo en función del número de iteraciones. La curva de pérdida muestra cómo cambia la función de pérdida del modelo a medida que se realiza el entrenamiento. Esto puede proporcionar información sobre la convergencia y el rendimiento del modelo.

– Si todo corre y tienes el modelo ajustado y la biblioteca ‘matplotlib.pyplot‘ importada correctamente, deberías ver la visualización de la curva de pérdida en la salida. Esto mostrará la curva de pérdida con el número de iteraciones en el eje x y el costo en el eje y.

Resultados

El modelo base resulta en una exactitud del 94%, la cual indica que el modelo neuronal tiene un buen alcance, pero no suficiente para hacer diagnósticos médicos exactos, ya que para ello es necesario más del 98%.

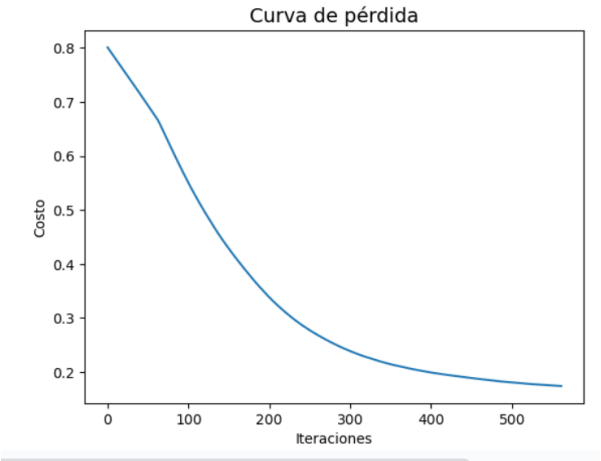


Figure 2: Curva de Pérdida

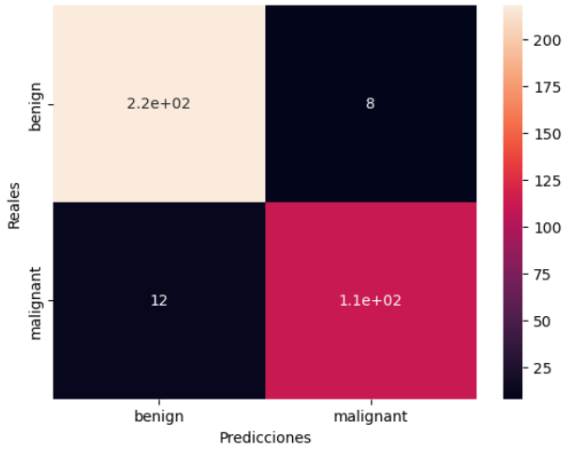


Figure 3: Prediciones vs Reales

4.2 Búsqueda de hiper-parámetros

Procedimientos

Realización de la búsqueda en cuadrícula de hiperparámetros utilizando el modelo ‘mlp_clf‘ y el conjunto de hiperparámetros definidos en ‘param_grid‘.

Sintonización de Hiper-parámetros

– Debemos de tener la biblioteca ‘sklearn.model_selection‘ importada correctamente. Esta biblioteca se utiliza para realizar la búsqueda en cuadrícula de hiperparámetros.

- Nos aseguramos de haber definido previamente el modelo ‘mlp_clf‘ y escalado los datos de entrenamiento ‘X_train_scaled‘ y las etiquetas de clase de entrenamiento ‘y_train‘. Esto significa que se deben haber ejecutado los fragmentos de código anteriores para cargar los datos, dividirlos en conjuntos de entrenamiento y prueba, y escalarlos.
- Ejecutamos el código. Se realizará la búsqueda en cuadrícula de hiperparámetros utilizando el modelo ‘mlp_clf‘ y el conjunto de hiperparámetros definidos en ‘param_grid‘. Esto implica entrenar y evaluar el modelo con diferentes combinaciones de hiperparámetros para encontrar la mejor configuración
  - ‘GridSearchCV(mlp_clf, param_grid, n_jobs=-1, cv=5)‘: Crea un objeto ‘GridSearchCV‘ con el modelo ‘mlp_clf‘, el conjunto de hiperparámetros ‘param_grid‘, ‘-1‘ para utilizar todos los procesadores disponibles en paralelo y ‘cv=5‘ para realizar la validación cruzada con 5 divisiones.
  - ‘Grid.fit(X_train_scaled, y_train)‘: Ejecuta la búsqueda en cuadrícula de hiperparámetros utilizando los datos de entrenamiento escalados ‘X_train_scaled‘ y las etiquetas de clase de entrenamiento ‘y_train‘.
- Si corre el código y se tiene el modelo definido y los datos escalados correctamente, la búsqueda en cuadrícula de hiperparámetros se ejecutará y se probarán diferentes combinaciones de hiperparámetros. Esto puede llevar algún tiempo dependiendo de la cantidad de combinaciones y el tamaño de los datos de entrenamiento.
- Luego de haber ejecutado el fragmento de código anterior que realiza la búsqueda en cuadrícula de hiperparámetros. Esto es necesario para que ‘grid‘ contenga los resultados de la búsqueda en cuadrícula.v
- Ejecutamos el código y se imprimirán los mejores valores de hiperparámetros encontrados por la búsqueda en cuadrícula en la salida.
  - ‘grid.best_params_‘: Devuelve un diccionario con los mejores valores de hiperparámetros encontrados por la búsqueda en cuadrícula. Esto se basa en la métrica de puntuación utilizada para evaluar los modelos en la validación cruzada.
  - ‘pd.DataFrame(grid.cv_results_)‘: Crea un objeto ‘DataFrame‘ de pandas con los resultados de la búsqueda en cuadrícula. Esto incluye información sobre los hiperparámetros probados, las puntuaciones de validación cruzada y otras métricas de rendimiento
- Si todo va bien y se ha realizado la búsqueda en la cuadrícula previamente, se imprimirán los mejores valores de hiperparámetros encontrados y se creará el objeto ‘DataFrame‘ con los resultados de la búsqueda en cuadrícula. Esto nos permite interactuar y analizar fácilmente los resultados obtenidos.
- Revisamos que tengamos la biblioteca ‘pandas‘ importada correctamente. Esta biblioteca se utiliza para crear y manipular el objeto ‘DataFrame‘ de pandas. Puedes importarla al comienzo del archivo de código si aún no lo has hecho.
- Vemos que se ha cargado y preparado previamente el conjunto de datos en el objeto ‘dataset‘. Esto significa que debes haber ejecutado los fragmentos de código anteriores para cargar los datos y realizar cualquier procesamiento necesario.
- Ejecutamos el código que aplicará la función ‘coder‘ a la columna ’classd’ del DataFrame ‘dataset‘ para codificar las clases. Luego, se agruparán los datos según la columna ’classd’ y se mostrará el tamaño de cada grupo.

- Si todo va bien y tienes el DataFrame ‘dataset‘ correctamente cargado, la función ‘coder‘ se aplicará a la columna ’classd’ y se mostrará el tamaño de cada grupo después de la codificación. Esto te dará una idea de la distribución de las clases después de la codificación.

Resultados

Estos son los parámetros encontrados para un modelo optimizado:

```
mlp_clf_opt = MLPClassifier(
    activation = 'tanh',
    alpha = 0.0001,
    hidden_layer_sizes = (150, 100, 50),
    learning_rate = 'adaptive',
    max_iter = 1200,
    solver = 'adam'
    )
```

4.3 Red Neuronal Optimizada

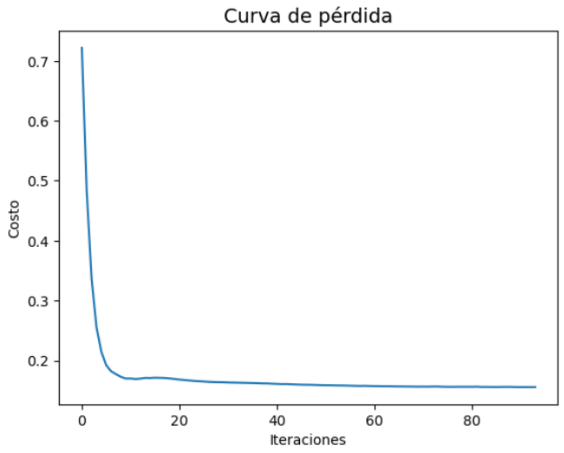


Figure 4: Curva de Pérdida

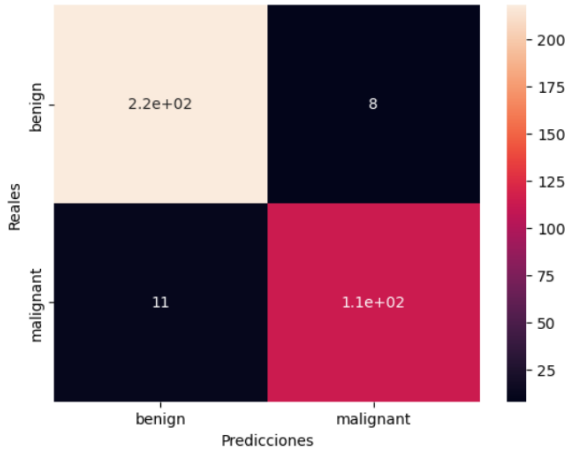


Figure 5: Prediciones vs Reales

4.4 Análisis de las redes neuronales

Procedimiento

- Debemos tener las bibliotecas necesarias importadas correctamente al comienzo del archivo de código. Esto incluye ‘pandas‘, ‘sklearn.preprocessing‘, ‘sklearn.model_selection.train_test_split‘ y ‘sklearn.preprocessing.StandardScaler‘.
- Nos percatamos de haber cargado y preparado previamente el conjunto de datos en el objeto ‘dataset‘ lo que significa que debemos haber ejecutado los fragmentos de código anteriores para cargar los datos y realizar cualquier procesamiento necesario.

Resultado

El nuevo modelo tomó menos iteraciones en llegar a una exactitud mejorada del 95%. La optimización se puede ver claramente en la caída más rápida del costo por iteración, llegando a su mínimo cerca de la iteración 40 en comparación con el modelo original, que llegaba al mismo punto hasta aproximadamente la iteración 600. No obstante, la exactitud sigue sin llegar al requisito para uso médico.

5 Regresión

5.1 Red Neuronal de base

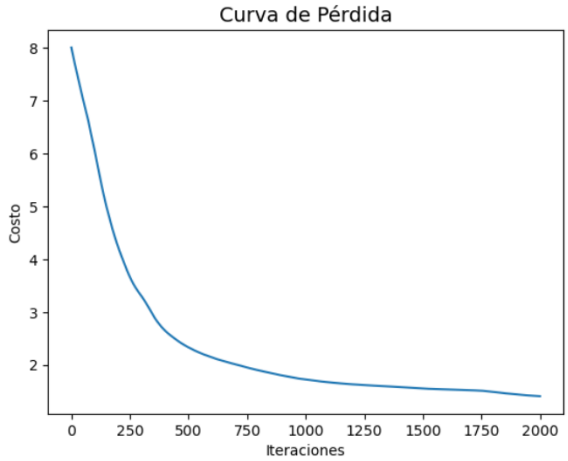
Como el diagnóstico solo puede resultar en un valor binario (benigno o maligno), optamos por usar el tamaño de la célula como la variable dependiente para nuestro modelo de regresión. Así podemos estimar un valor numérico a partir de las variables de grosor y adhesión marginal.

5.1.1 Conjunto de datos para entrenamiento

- Se eliminará la columna ’cell_size” del DataFrame ‘dataset‘ y se asignará el resultado a ‘X‘. Esto es para crear el conjunto de datos ‘X‘ sin la característica de interés.
- Se convertirá ‘X‘ en un arreglo NumPy y se asignarán las columnas 0 y 1 (sepal length y sepal width) a ‘X‘.
- La columna ’classd’ se asignará a ‘y‘ como el conjunto de datos de la característica de interés.
- Se dividirá ‘X‘ y ‘y‘ en conjuntos de entrenamiento y prueba utilizando ‘train_test_split‘.
- Se imprimirán los valores mínimos y máximos del DataFrame ‘dataset‘.
- Se creará un objeto ‘StandardScaler‘ y se ajustará a los datos de entrenamiento (‘X_train‘) utilizando ‘fit‘. Luego, se escalarán tanto los conjuntos de entrenamiento (‘X_train‘) como los conjuntos de prueba (‘X_test‘) utilizando ‘transform‘.
- Se imprimirán los valores mínimos y máximos del conjunto de datos escalado (‘X_train_scaled‘).
- Si el código corre bien y el DataFrame ‘dataset‘ está correctamente cargado, se realizarán las operaciones mencionadas y se mostrarán los valores mínimos y máximos del conjunto de datos original y escalado.

Entrenamiento

Definición de un objeto ‘MLPRegressor‘ con los parámetros especificados, incluyendo la estructura de capas, el número máximo de iteraciones, la función de activación y el optimizador. Entrenamiento del modelo utilizando el conjunto de datos de entrenamiento escalado (‘X_train_scaled‘, ‘y _train‘) utilizando el método ‘fit‘



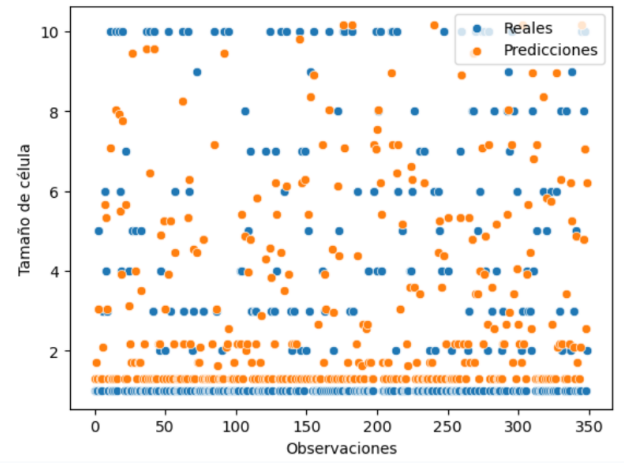
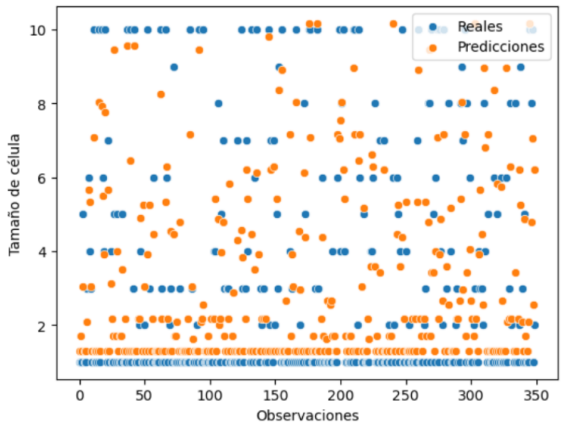


Figure 6: Observaciones vs Tamaño de la Célula

5.2 Búsqueda de hiper-parámetros

```
mlp_reg_opt = MLPRegressor(
    activation = 'relu',
    alpha = 0.0001,
    hidden_layer_sizes = (120, 80),
    learning_rate = 'constant',
    max_iter = 600,
    solver = 'adam'
    )
```

5.3 Red Neuronal Optimizada



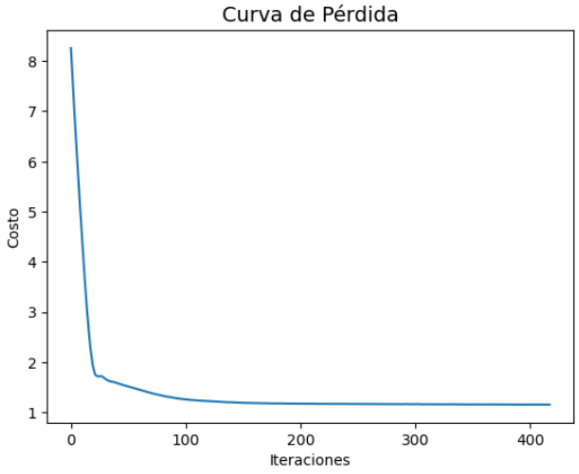


Figure 7: Observaciones vs Tamaño de la Célula

5.4 Análisis de las redes neuronales

5.4.1 Comparación de redes

A diferencia de los modelos anteriores, ambas versiones de la red neuronal de correlación mostraron una exactitud similar, con coeficientes de correlación del 67%, errores cuadráticos promedios de 3.17 y 3.18, y varianza explicada del 67%. Esto significa que sí existe una correlación importante entre las variables. Aproximadamente el 67% de la variación en el tamaño de célula puede ser explicada con las variables de grosor y adhesión marginal, con el otro 33% siendo atribuible a las demás variables dentro del estudio.

Modelo de Regresión

- Chequeamos tener las bibliotecas necesarias importadas correctamente al comienzo del archivo de código. Esto incluye ‘sklearn.neural_network.MLPRegressor‘.
- Aseguramos el haber cargado y preparado previamente los conjuntos de datos de entrenamiento (‘X_train_scaled‘, ‘y_train‘) y prueba (‘X_test_scaled‘). Esto significa que se debe haber ejecutado los fragmentos de código anteriores para cargar los datos, realizar cualquier procesamiento necesario y escalar los conjuntos de datos.
- Ejecutamos el código y se realizarán las siguientes operaciones:
  * Se definirá un objeto ‘MLPRegressor‘ con los parámetros especificados, incluyendo la estructura de capas, el número máximo de iteraciones, la función de activación y el optimizador.
  * Se entrenará el modelo utilizando el conjunto de datos de entrenamiento escalado (‘X_train_scaled‘, ‘y_train‘) utilizando el método ‘fit‘.
  * Se utilizará el conjunto de datos de prueba escalado (‘X_test_scaled‘) para realizar las predicciones utilizando el método ‘predict‘. El resultado se asignará a ‘y_pred‘.

- Si funciona y se tienen los conjuntos de datos de entrenamiento y prueba correctamente cargados y escalados, se realizarán las operaciones mencionadas y se obtendrán las predicciones ‘y_pred‘ para el conjunto de datos de prueba.
- Nos aseguramos de tener las bibliotecas necesarias importadas correctamente al comienzo del archivo de código. Esto incluye ‘sklearn.metrics.r2_score‘, ‘sklearn.metrics.mean_squared_error‘y ‘sklearn.metrics.explained_variance_score‘.
- Aseguramos haber ejecutado previamente el fragmento de código donde se entrena el modelo y se obtienen las predicciones ‘y_pred‘.
- Ejecuta el código. Se calcularán y mostrarán las siguientes métricas de evaluación:
  * R2 score: Es una medida de la bondad del ajuste del modelo. El mejor puntaje posible es 1.0 y puede ser negativo. Se calcula utilizando el método ‘r2_score‘.
  * Mean square error (MSE): Es una medida del error cuadrático promedio entre las predicciones y los valores reales. El mejor valor posible es 0.0. Se calcula utilizando el método ‘mean_squared_error‘.
  * Explained variance score: Es una medida de qué tan bien el modelo explica la varianza de los datos. El mejor valor posible es 1.0, valores más bajos indican un peor ajuste. Se calcula utilizando el método ‘explained_variance_score‘.
  * Se trazará la curva de pérdida (loss curve) del modelo entrenado utilizando el método ‘plot‘ de ‘matplotlib.pyplot‘. Esto mostrará cómo disminuye la pérdida a lo largo de las iteraciones del entrenamiento.
- Si todo va bien y tienes las predicciones ‘y_pred‘ calculadas correctamente, se calcularán las métricas de evaluación y se mostrarán los resultados. Además, se mostrará la curva de pérdida del modelo entrenado.

6 Conclusiones

Ambos modelos demostraron una exactitud elevada en la clasificación y regresión de datos relativos al cáncer de mama. El primer modelo, en especial, promete un alto potencial en usos prácticos como herramienta para el diagnóstico de cáncer de mama.

El primer modelo mostró los resultados más importantes. Nuestra red neuronal logró predecir con un 95% de exactitud si una célula es benigna o maligna, indicando que los datos de tamaño y forma pueden ser suficientes para realizar diagnósticos médicos con relativamente poca información. Si bien la exactitud no supera el 98% necesario para hacer diagnósticos formales, este modelo puede informar a un médico si debería de efectuar más pruebas para diagnosticar el cáncer.

En el segundo modelo pudimos encontrar una correlación considerable entre tamaño de célula con respecto al grosor y adhesión marginal, pero el 67% de coeficiente r2 indica que existe otro 33% que podría ser explicado con las demás variables numéricas del estudio. Convendría expandir este estudio para considerar los otros rubros dentro de la red neuronal.

En ambas instancias, una mayor población de muestras podría aumentar drásticamente la exactitud de estos modelos hasta la exactitud necesaria como para hacer diagnósticos médicos legítimos.

Los resultados fueron suficientemente adecuados para un espacio con solo 569 muestras, pero un espacio de cientos de miles o millones de registros podría hacer la diferencia.

7 Referencias

- L. Mangasarian and W. H. Wolberg: ”Cancer diagnosis via linear programming”, SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.
- William H. Wolberg and O.L. Mangasarian: ”Multisurface method of pattern separation for medical diagnosis applied to breast cytology”, Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.
- O. L. Mangasarian, R. Setiono, and W.H. Wolberg: ”Pattern recognition via linear programming: Theory and application to medical diagnosis”, in: ”Large-scale numerical optimization”, Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30.
- K. P. Bennett & O. L. Mangasarian: ”Robust linear programming discrimination of two linearly inseparable sets”, Optimization Methods and Software 1, 1992, 23-34 (Gordon & Breach Science Publishers)
- UCI Machine Learning Repository: Breast Cancer Wisconsin (Diagnostic) Data Set. (n.d.).

  https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29
- Abbass, H. A. (2002). An evolutionary artificial neural networks approach for breast cancer diagnosis. Artificial Intelligence in Medicine, 25(3), 265-281. https://doi.org/10.1016/s0933-3657(02)00028-3
- Bennett, Kristin & Demiriz, Ayhan & Maclin, Richard. (2002). Exploiting Unlabeled Data in Ensemble Methods. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 10.1145/775047.775090.
- Street, N. (1999). A Neural Network Model for Prognostic Prediction. ResearchGate.

  https://www.researchgate.net/publication/2417794_A_Neural_Network_Model_for_Prognostic_Prediction/