| Mini Proyecto Inteligencia Artificial Avanzada.  Entrenando un modelo de Compatibilidad en Relaciones |
| --- |

| Emilia Salazar Leipen, Daniel Darwich Harari y Santiago Herrero Guzmán    [**I. introducción 2**](#_yp6rjdxmbr03)  [**II. Metodología 2**](#_7za8hrr3et8z)  [A. Recolección de datos 2](#_ya4zerusk19t)  [B. Preprocesamiento de Datos 3](#_hzrx0l82ma67)  [C. División del Conjunto de Datos 3](#_hl3pfvltr32n)  [D. Modelos de Predicción 3](#_ux0tj6i4xt)  [E. Optimización de los Modelos 5](#_o47s9epys4ty)  [F. Evaluación del Modelo 6](#_lzc87x86jc6s)  [**III. La matemática 6**](#_lym3ck4l89w7)  [**V. RESULTADOS 7**](#_iwfw9ihyyyn)  [A. Figuras y tablas 7](#_szjy81nfp25)  [B. Insights 9](#_le4kiinxkbe6)  [**VI. Algunos Errores Comunes 11**](#_83xfnx1ke5cw)  [**VII. Interfaz Interactiva 11**](#_181v4hm0ishq)  [Referencias 13](#_ung0p575yzas) |
| --- |

**Resumen - Este proyecto tiene como objetivo predecir la compatibilidad en relaciones basándose en un cuestionario que mide 31 características de la relación. Se recopilaron 126 respuestas, de las cuales algunas relaciones funcionaron (indicadas con un 1) y otras no (indicadas con un 0). El objetivo del modelo es clasificar correctamente si una relación es exitosa o no, basándose en las respuestas de los participantes.**

# I. introducción

Este proyecto se tratará de la manera en la que se entrenó un modelo de predicción para poder predecir la compatibilidad en relaciones sentimentales, utilizando como base un conjunto de datos recolectado a través de encuestas. El objetivo principal es evaluar si una relación funcionará o no, a partir de 31 características relacionadas con la comunicación, confianza, valores compartidos, y otros aspectos clave de la dinámica de pareja.

Para este fin, se implementaron y compararon tres enfoques diferentes basados en redes neuronales: una red neuronal profunda utilizando Keras, un modelo completamente entrenado de forma manual, y un perceptrón multicapa (MLP) con Scikit-Learn. Cada modelo fue evaluado utilizando métricas como precisión, recall, F1-score, y exactitud, con el fin de identificar cuál de ellos ofrece el mejor rendimiento en la predicción de la compatibilidad.

Este documento explorará en detalle el proceso de entrenamiento de los modelos, la arquitectura de cada uno, los métodos de optimización utilizados, y los resultados obtenidos, proporcionando un análisis comparativo que permita comprender la efectividad de cada enfoque en este contexto. Además, se discutirá la relevancia de las características del dataset y cómo estas influyen en el desempeño predictivo de los modelos.

II. Metodología

### *A.* *Recolección de datos*

Para el modelo de compatibilidad se inicializó un proceso de recolección de datos que tuvo una duración de 1 semana. Durante este tiempo, los tres integrantes del equipo buscaron personas con diferentes características (sexo, edad, ocupación, etc.) para que completaran una encuesta de 31 preguntas, enfocadas en aspectos clave de las relaciones sentimentales. Las preguntas fueron diseñadas para capturar información relevante sobre la comunicación, la confianza, los valores compartidos y otros elementos esenciales de una relación. La manera en la que se escribieron fueron hechas cómo frases en donde podían elegir del 1 al 5 que tanto se relacionaban con cada frase en cuánto a su relación. Esta encuesta se podía contestar tanto para una relación que no funcionó o una que sigue funcionando.

Las preguntas incluidas en la encuesta fueron las siguientes:

1. Diferencia de Edad entre las dos personas.
2. Nos comunicamos de manera abierta y honesta.
3. Confío plenamente en mi pareja.
4. Compartimos valores y creencias similares.
5. Somos capaces de resolver conflictos sin que afecten nuestra relación.
6. Nos brindamos apoyo emocional cuando es necesario.
7. Compartimos intereses y pasatiempos.
8. Nuestras metas futuras están alineadas.
9. Respetamos las opiniones y decisiones del otro.
10. Mantenemos nuestra independencia mientras estamos juntos.
11. Mostramos afecto regularmente.
12. Disfrutamos pasar tiempo juntos.
13. Tenemos un buen entendimiento de los hábitos financieros del otro.
14. Nuestras familias se llevan bien entre sí.
15. Tenemos un balance saludable entre trabajo y vida personal.
16. Estamos dispuestos a comprometernos por el bien de la relación.
17. Compartimos un sentido del humor similar.
18. Estamos satisfechos con el nivel de intimidad en nuestra relación.
19. Podemos resolver problemas juntos de manera efectiva.
20. Tenemos una vida social saludable juntos y por separado.
21. Entendemos y respetamos los antecedentes culturales del otro.
22. Planificamos juntos para el futuro.
23. Hay una fuerte atracción física entre nosotros.
24. Nos sentimos intelectualmente compatibles.
25. Podemos ponernos en el lugar del otro.
26. Podemos adaptarnos a los cambios en la relación.
27. Estoy generalmente satisfecho con nuestra relación.
28. Nos apoyamos mutuamente en tiempos difíciles.
29. Compartimos las responsabilidades de manera justa.
30. Tenemos la libertad de ser nosotros mismos en la relación.
31. Somos capaces de perdonarnos mutuamente.

Al final se consiguieron 126 respuestas, 63 para relaciones que no funcionaron y 63 para relaciones que siguen funcionando.

### B. Preprocesamiento de Datos

Una vez que se recolectaron todos los datos, fue necesario prepararlos para el proceso de entrenamiento del modelo. El primer paso consistió en limpiar los datos, eliminando cualquier entrada incompleta o inválida. Posteriormente, se realizó una transformación de los datos para asegurar que todas las características fueran numéricas, lo que es esencial para entrenar modelos de aprendizaje automático.

Para la variable de salida, se siguió un proceso de codificación binaria:

* Las respuestas que indicaban que la relación no funcionó fueron codificadas con un valor de **0**.
* Las respuestas que indicaban que la relación sigue funcionando fueron codificadas con un valor de **1**.

Este preprocesamiento convirtió el conjunto de datos en un formato adecuado para el entrenamiento de los modelos de predicción, asegurando que tanto las características como los resultados estuvieran correctamente estructurados y preparados para el análisis.

### C. División del Conjunto de Datos

El conjunto de datos se dividió en un 80% para entrenamiento, lo que equivale a 100 datos, y el 20% restante se utilizó para prueba. Esta división permite que el modelo aprenda de la mayor parte de los datos y que luego sea evaluado con datos que no ha visto previamente, garantizando una validación justa de su rendimiento. Esta estrategia asegura que el modelo no solo se ajuste bien a los datos de entrenamiento, sino que también generalice correctamente a nuevos ejemplos.

### D. Modelos de Predicción

1. **Modelo con Keras**

El primer modelo se implementó utilizando la biblioteca Keras. El modelo se diseñó con una arquitectura secuencial compuesta por varias capas densas que permiten extraer patrones complejos a partir de los datos.

Arquitectura del Modelo

* Capa de entrada: Se definió una capa de entrada que toma como dimensión de entrada el número de características del conjunto de datos (31 características).
* Primera capa oculta: Se agregó una capa densa con 128 neuronas y activación ReLU, lo que permite al modelo aprender representaciones no lineales.
* Segunda capa oculta: Posteriormente, se añadió una segunda capa densa con 64 neuronas, también con activación ReLU.
* Capa de salida: Finalmente, se añadió una capa de salida con una sola neurona y activación sigmoidal, que genera una probabilidad de si la relación funciona (1) o no (0).

Optimización y Entrenamiento

El modelo se entrenó utilizando el optimizador Adam con una tasa de aprendizaje de 0.0005, lo que proporciona una buena velocidad de convergencia y estabilidad. La función de pérdida utilizada fue binary\_crossentropy, adecuada para problemas de clasificación binaria.

El modelo se entrenó durante 15 épocas con un tamaño de lote de 16 datos. Este proceso permitió monitorear el rendimiento del modelo a medida que aprendía, evaluando tanto la precisión en el conjunto de entrenamiento como en el conjunto de validación.

1. **Modelo con implementación Manual**

El segundo modelo se implementó de manera manual, lo que permitió un control más detallado sobre cada paso del entrenamiento de la red neuronal. Este enfoque implicó la implementación desde cero de los algoritmos de forward propagation, backpropagation, y la actualización de pesos utilizando gradiente descendente.

Arquitectura del Modelo

* Capa de entrada: La red toma como entrada el mismo conjunto de 31 características numéricas.
* Primera capa oculta: Se implementó una capa oculta con 256 neuronas, donde se aplicó la función de activación sigmoidal.
* Segunda capa oculta: A continuación, se añadió una segunda capa oculta con 128 neuronas y la misma función de activación sigmoidal.
* Tercera capa oculta: Se incluyó una tercera capa oculta con 64 neuronas, nuevamente con activación sigmoidal.
* Capa de salida: Finalmente, la capa de salida consta de una sola neurona con activación sigmoidal, que genera una predicción de 0 o 1 para determinar si la relación funcionó o no.

Inicialización de Pesos y Propagación

Los pesos de cada capa fueron inicializados aleatoriamente con valores pequeños cercanos a 0 para evitar que el modelo se atasque en soluciones triviales. Durante el proceso de forward propagation, los datos se pasaron a través de las capas de la red, y se aplicó la función sigmoidal en cada capa oculta y en la salida para calcular la predicción.

Backpropagation y Actualización de Pesos

Durante el entrenamiento, se utilizó el algoritmo de backpropagation para calcular los gradientes del error con respecto a los pesos de la red. Estos gradientes fueron utilizados para ajustar los pesos, con el objetivo de minimizar el error de predicción. El proceso se repitió en múltiples iteraciones hasta que el error total se redujo por debajo de un umbral de 0.05.

El algoritmo de gradiente descendente fue utilizado para actualizar los pesos en cada iteración, con una tasa de aprendizaje fija de 0.01. Este proceso permitió ajustar el modelo para que aprendiera de manera progresiva a predecir con mayor precisión la compatibilidad en las relaciones.

1. **Modelo con Sckit-learn**

El tercer modelo se implementó utilizando la herramienta Scikit-Learn, específicamente el MLPClassifier (Perceptrón Multicapa). Este enfoque permite entrenar redes neuronales de manera rápida y eficiente utilizando funciones predefinidas de Scikit-Learn, que simplifican el proceso de construcción y optimización del modelo.

Arquitectura del Modelo

* Capa de entrada: Al igual que en los otros modelos, la capa de entrada toma las 31 características del conjunto de datos.
* Primera capa oculta: Se definió una primera capa oculta con 256 neuronas y la función de activación ReLU, la cual ayuda a manejar relaciones no lineales en los datos.
* Segunda capa oculta: La segunda capa oculta tiene 128 neuronas, también con la activación ReLU.
* Tercera capa oculta: La tercera capa oculta se implementó con 64 neuronas, usando nuevamente la activación ReLU.
* Capa de salida: La capa de salida tiene una neurona con activación sigmoidal, lo que permite que el modelo realice una predicción binaria (0 o 1).

Optimización y Entrenamiento

El modelo fue entrenado utilizando el optimizador Adam, que ajusta automáticamente la tasa de aprendizaje a lo largo del proceso de entrenamiento. La tasa de aprendizaje inicial se fijó en 0.0005, similar a la utilizada en el modelo de Keras.

El MLPClassifier en Scikit-Learn ajusta los pesos de las conexiones en la red neuronal usando backpropagation, y al final del proceso se obtiene una red ajustada para clasificar si una relación funcionó o no. Este proceso se realizó durante 100 iteraciones, o hasta que la convergencia del modelo se alcanzara antes de este límite.

### E. Optimización de los Modelos

Una vez que nos obtuvimos los resultados de los modelos, implementamos dos modelos diferentes en el que modificamos características y parámetros para mejorar la precisión de nuestro modelo. Estos cambios se realizaron en dos nuevas redes neuronales e incluyeron:

**1. Eliminación de Columnas Irrelevantes**

Uno de los primeros pasos en la optimización fue la eliminación de características (columnas) que no aportaban valor significativo al modelo. En la matriz de correlación mostrada en el heatmap, observamos que algunas variables tenían una correlación muy baja con la variable objetivo (Target, que indica si una relación funcionó o no). Estas fueron:

* Mantenemos nuestra independencia mientras estamos juntos
* Tenemos un balance saludable entre trabajo y vida personal
* Compartimos un sentido del humor similar
* Estamos satisfechos con el nivel de intimidad en nuestra relación

**2. Búsqueda de Mayor Precisión**

Durante el entrenamiento del modelo, nos enfocamos en mejorar la precisión como métrica clave. La precisión mide la proporción de verdaderos positivos entre todas las predicciones positivas, es decir, la capacidad del modelo de evitar falsos positivos. En el contexto de predecir la compatibilidad en relaciones, minimizar los falsos positivos es crucial, ya que no queremos clasificar incorrectamente una relación como exitosa cuando realmente no lo es.

**3. Uso de Dropout**

El Dropout es una técnica regularizadora que se utiliza para evitar el sobreajuste (overfitting) del modelo. Durante el entrenamiento, el modelo puede aprender a memorizar los datos en lugar de generalizar bien en datos nuevos. Para mitigar este problema, se agregó Dropout en las capas ocultas de la red.

El Dropout consiste en desactivar aleatoriamente un porcentaje de las neuronas durante el entrenamiento, evitando que el modelo dependa excesivamente de ciertos patrones. En nuestro caso, utilizamos una tasa de dropout del 10%, lo que significa que, en cada iteración, el 10% de las neuronas no participaron, lo que ayudó a hacer el modelo más robusto y generalizable.

**4. Cross Validation (Validación Cruzada)**

Para evaluar la robustez del modelo, implementamos validación cruzada con K-Fold. Este proceso divide los datos en K subconjuntos (folds) y entrena el modelo K veces, utilizando un subconjunto diferente para validar en cada iteración. Esto nos permite medir el rendimiento promedio del modelo y su capacidad de generalización.

La validación cruzada asegura que el modelo no esté sobreentrenado en un único conjunto de datos y, por lo tanto, nos da una evaluación más realista de cómo se comportaría en datos nuevos.

**5. Ajuste del Umbral de Clasificación (Threshold)**

Además de la arquitectura del modelo, también se ajustó el umbral de clasificación, conocido como threshold. Normalmente, el umbral predeterminado para clasificar una predicción positiva es 0.5; sin embargo, en este caso, se ajustó a 0.6 para mejorar la precisión. Este cambio nos ayudó a reducir el número de falsos positivos, priorizando predicciones con mayor confianza de que una relación realmente funcionó.

### F. Evaluación del Modelo

Todos los modelos (Keras, manual y Scikit-Learn) se evaluaron de manera consistente utilizando las mismas métricas clave, lo que permitió realizar una comparación justa entre los enfoques. Las métricas utilizadas para la evaluación fueron:

1. Precisión (Precision): La capacidad del modelo para identificar correctamente las relaciones que funcionaron, minimizando los falsos positivos.
2. Recall (Sensibilidad): La habilidad del modelo para detectar correctamente las relaciones que no funcionaron, evitando falsos negativos.
3. F1-Score: La media armónica entre precisión y recall, proporcionando una medida equilibrada de rendimiento cuando existe una cierta disparidad entre estas dos métricas.
4. Exactitud (Accuracy): El porcentaje total de predicciones correctas del modelo, tanto de relaciones que funcionaron como de aquellas que no funcionaron.

Igualmente se crearon diferentes gráficas para visualizar las comparaciones entre modelos.

# III. La matemática

**Función Sigmoidal**: Mapea cualquier valor de entrada al rango de 0 a 1. Se utiliza principalmente en clasificación binaria, pero sufre de saturación cuando los valores de entrada son muy altos o bajos, lo que puede dificultar el entrenamiento.

**Función ReLU (Rectified Linear Unit)**: Asigna los valores negativos a 0 y mantiene los valores positivos sin modificaciones. Es ampliamente usada debido a su simplicidad y eficiencia en el entrenamiento, evitando el problema de saturación.

**Gradiente Descendente:** es un algoritmo de optimización que busca el mínimo de una función iterando hacia abajo en la pendiente (gradiente) de la función en puntos específicos.

**Forward propagation:** consiste en pasar los datos a través de la red neuronal, calculando la salida en función de los pesos actuales y las entradas.

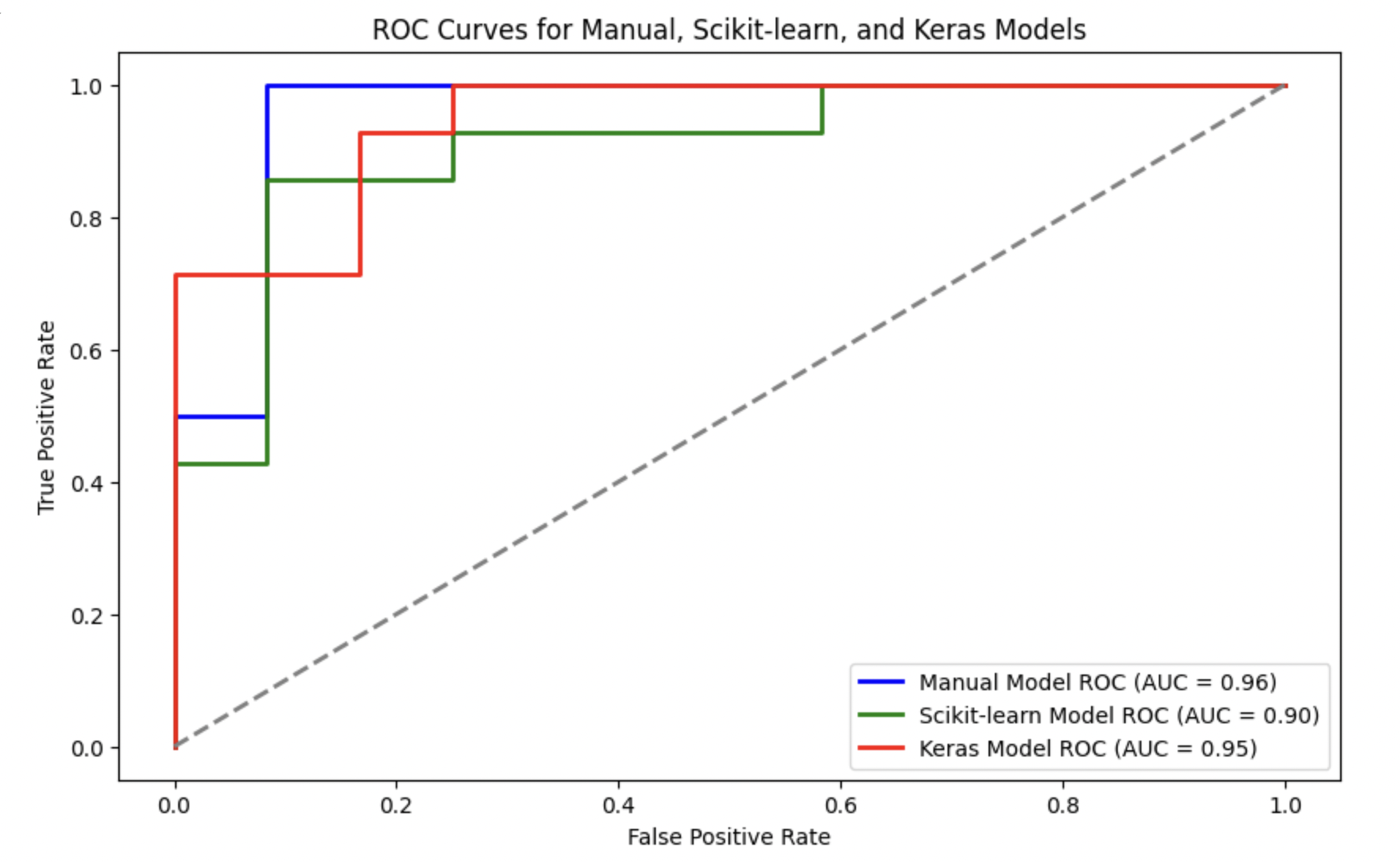
**Backpropagation:** utiliza la derivada de la función de pérdida con respecto a los pesos para ajustar estos en dirección opuesta al gradiente, minimizando el error total a través de la regla de la cadena. Ambos procesos son esenciales para entrenar redes neuronales de manera efectiva.

**Adam (Adaptive Moment Estimation)**: combina las ventajas de RMSProp y Momentum. Utiliza una tasa de aprendizaje adaptativa para cada parámetro en función del momento y la magnitud del gradiente. Esto lo hace más eficiente para ajustar los parámetros del modelo, lo que puede mejorar la precisión en comparación con otros métodos de optimización.

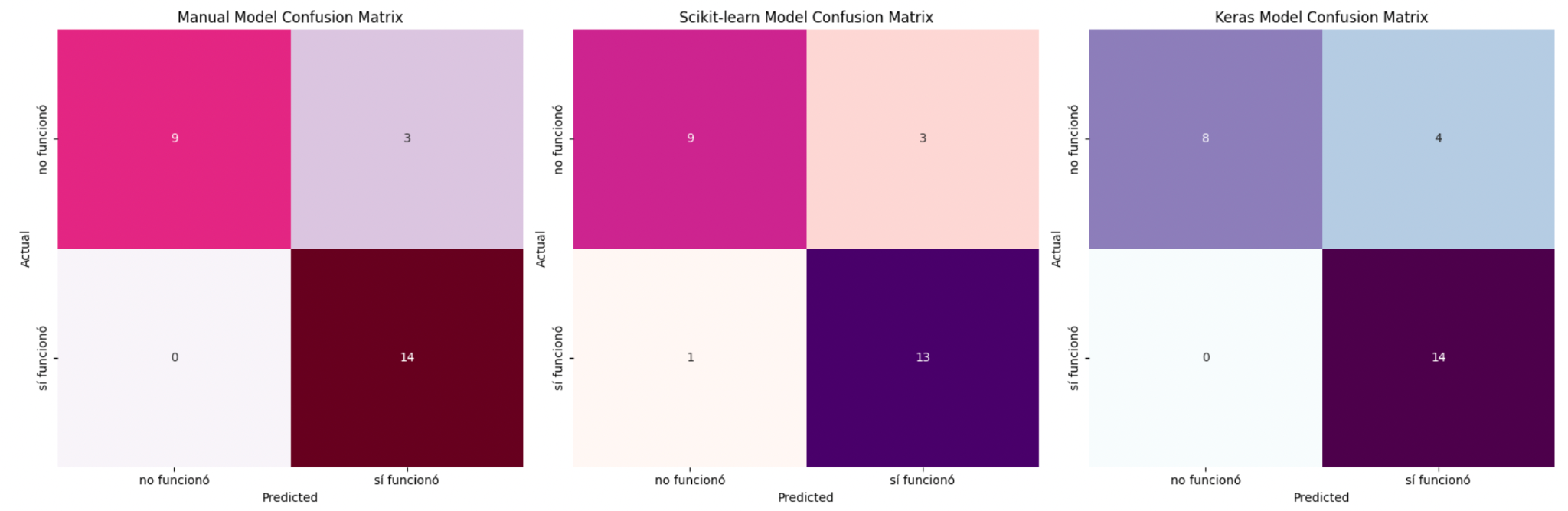
# V. RESULTADOS

### A. Figuras y tablas

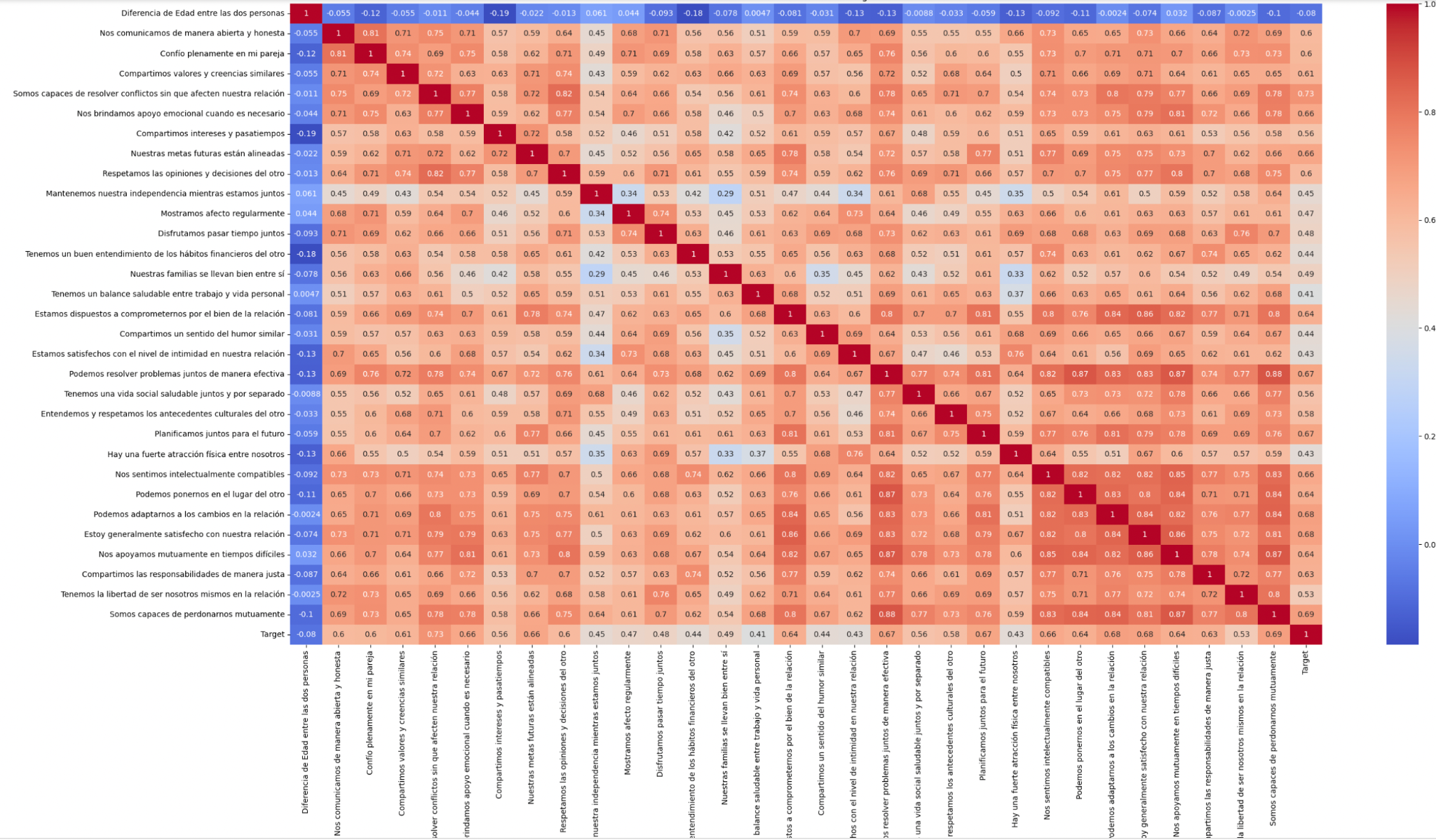
*Figura 1: Gráfico de comparación de métricas redes iniciales*



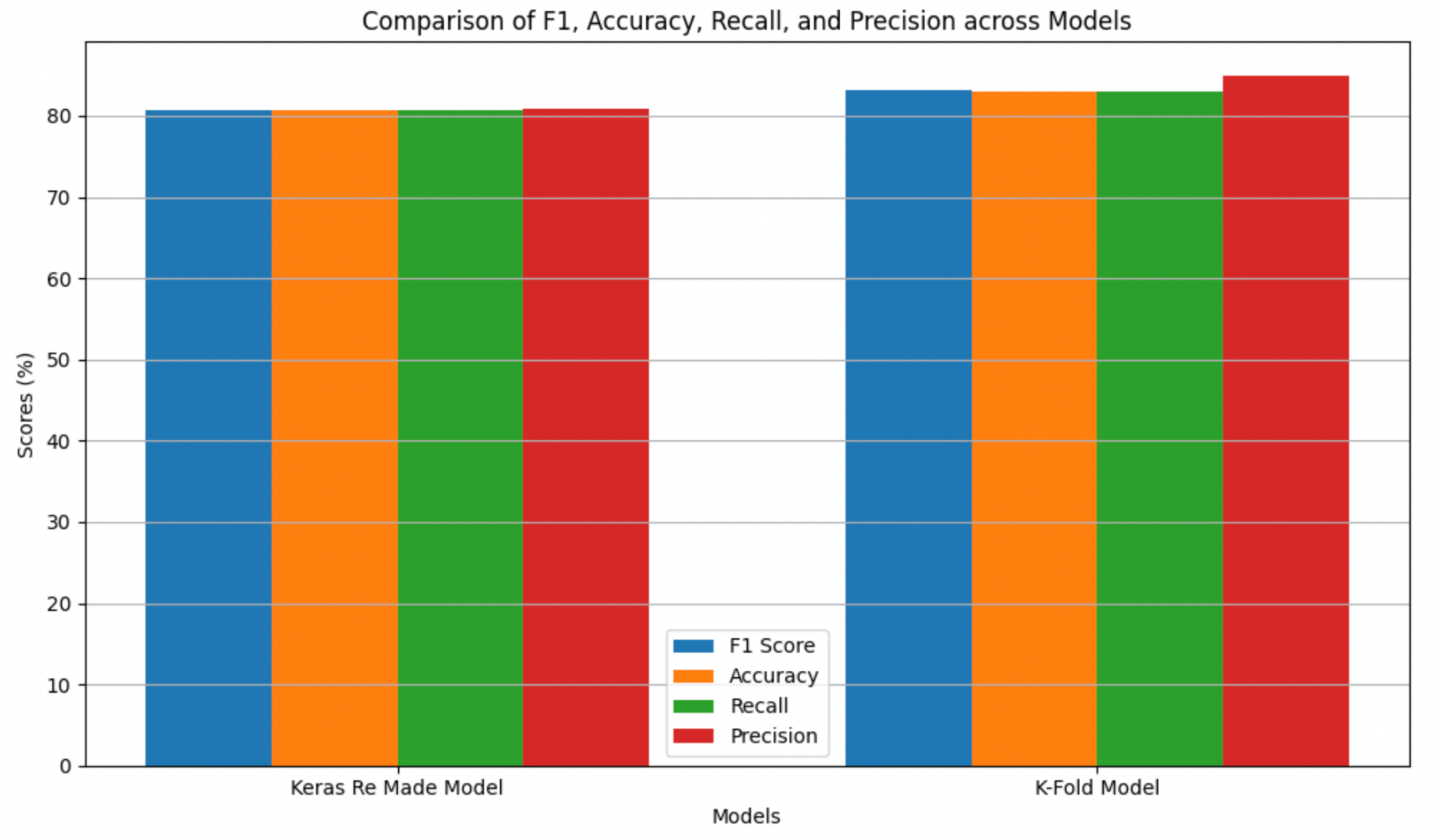
*Figura 2: Gráfico ROC/AUC de tres redes iniciales*

**

*Figura 3: Matriz de confusión 3 de tres redes iniciales*

**

*Figura 4: Mapa de calor de características de entrenamiento*

**

*Figura 5: Gráfico de comparación de metricas de 2 redes evaluadas*

### B. Insights

**1. Gráfica de comparación de F1, Exactitud, Recall y Precisión entre los modelos:**

Esta gráfica compara el desempeño de tres modelos (Keras, Manual y Scikit-learn) en cuanto a F1 Score, Exactitud, Recall y Precisión. El Modelo Manual se destaca por su alto Recall, lo que sugiere que captura mejor los casos positivos (minimizando falsos negativos), mientras que el Modelo de Scikit-learn tiene la mayor Precisión, lo que implica que tiene menos falsos positivos. El Modelo de Keras, por otro lado, muestra un equilibrio entre todas las métricas, lo que lo convierte en una opción bien balanceada para aquellos que busquen rendimiento general.

**2. Curvas ROC para los tres modelos:**

En la gráfica de las curvas ROC, los tres modelos son comparados por su capacidad de clasificación mediante el AUC (Área bajo la curva). El Modelo Manual se destaca ligeramente con un AUC de 0.96, lo que significa que tiene una alta capacidad para distinguir correctamente entre las clases positivas y negativas. El Modelo de Keras sigue de cerca con un AUC de 0.95, mientras que el Modelo de Scikit-learn tiene un AUC de 0.90, que aunque es menor, sigue siendo un buen resultado. Esto indica que todos los modelos tienen un buen rendimiento en términos de separación de clases, pero el Modelo Manual sobresale ligeramente.

**3. Matrices de Confusión de los tres modelos:**

Las matrices de confusión comparan la precisión de las predicciones de los tres modelos en términos de casos correctamente e incorrectamente clasificados. El Modelo Manual muestra una alta precisión en la clasificación de casos positivos, sin embargo, comete algunos errores al clasificar los negativos. El Modelo de Scikit-learn tiene un patrón similar con algunos errores tanto en casos positivos como negativos. El Modelo de Keras tiene un rendimiento equilibrado pero comete más errores al clasificar casos negativos. En general, el Modelo Manual es particularmente bueno para identificar casos positivos, aunque todos los modelos podrían beneficiarse de una mejor clasificación de los negativos.

**4. Mapa de calor de correlación entre características y el objetivo:**

El mapa de calor muestra las correlaciones entre las 31 características del modelo y el éxito de la relación (si funcionó o no). Algunas características como "Confío plenamente en mi pareja" y "Nos brindamos apoyo emocional" tienen una fuerte correlación positiva con el éxito de la relación, lo que sugiere que son buenos predictores. Por el contrario, la "Diferencia de Edad entre las dos personas" tiene una correlación débil y negativa con el éxito de la relación. Este mapa es útil para identificar cuáles características tienen un mayor impacto en la predicción de relaciones exitosas y puede orientar el enfoque de futuros modelos predictivos.

**5. Comparación de F1, Exactitud, Recall y Precisión en modelos Keras remade y K-Fold:**

Esta gráfica compara dos modelos: uno de Keras mejorado (remade) y otro basado en validación cruzada K-Fold. Ambos modelos presentan un rendimiento alto y equilibrado en todas las métricas evaluadas (F1 Score, Exactitud, Recall y Precisión). Sin embargo, el modelo de Keras remade tiene un pequeño margen de ventaja en Recall y Precisión, lo que sugiere que es más efectivo para reducir tanto falsos negativos como falsos positivos. El modelo K-Fold, aunque también efectivo, tiene un ligero descenso en la Precisión, pero sigue siendo un modelo robusto para la clasificación.

A lo largo del proyecto, se exploraron diversas estrategias para mejorar el rendimiento del modelo. Sin embargo, una limitación clave fue el tamaño del dataset, que era muy pequeño para obtener conclusiones definitivas o lograr un modelo altamente preciso. Como es bien sabido, "cuantos más datos, mejor la información," y en este caso, la falta de un gran volumen de datos afectó el rendimiento potencial del modelo.

En la segunda parte de reevaluación del modelo se brindó mayor atención en prevenir el sobreajuste (overfitting) utilizando la técnica de k-fold cross-validation. Esta metodología fue crucial, ya que nos permitió dividir el conjunto de datos en diferentes subconjuntos de entrenamiento y validación, asegurando que el modelo fuera capaz de generalizar y no se limitara a memorizar el conjunto de entrenamiento.

Nuestro enfoque para mejorar la precisión implicó ajustar tanto la estructura de la red neuronal como los hiperparámetros clave. Por ejemplo, mientras que otros modelos intentaron maximizar el recall, en este caso se realizaron ajustes específicos, como modificar la tasa de aprendizaje y reducir el número de neuronas en las capas finales, para evitar que el modelo aprendiera patrones irrelevantes o poco útiles. Este enfoque está alineado con la necesidad de reducir el número de falsos positivos, dado que en nuestro caso, predecir incorrectamente una relación positiva sería más perjudicial que perder algunos casos verdaderamente positivos (falsos negativos).

Al final del día, aunque el uso de k-fold y los ajustes específicos ayudaron a mejorar la precisión, el tamaño del dataset siguió siendo un desafío que limitó el rendimiento general del modelo. Un conjunto de datos más grande proporcionaría más información y permitiría al modelo capturar patrones más robustos, lo que llevaría a mejores resultados en términos de precisión y generalización. Este proyecto deja claro que la cantidad de datos disponibles juega un rol crucial, y futuras versiones del modelo deberían considerar ampliar el dataset para mejorar el rendimiento general.

Sin embargo creemos que el modelo K-fold es el modelo mejor hecho ya que no esta sobreentrenado y el que utilizaríamos en iteraciones futuras del proyecto ya que es el que da los datos más reales y mejor entrenamiento a un modelo.

# VI. Algunos Errores Comunes

Si volviéramos a hacer el proyecto, uno de los principales cambios sería modificar la clasificación. En lugar de utilizar solo dos etiquetas (funcionó o no funcionó), optaríamos por una clasificación más detallada, dividiendo la compatibilidad en cinco clases. Esto permitiría tener un rango más amplio sobre el nivel de éxito de la relación. Sin embargo, cuando intentamos implementar este enfoque durante el proyecto, nos encontramos con un gran desafío: la distribución de etiquetas estaba completamente desequilibrada, con muy pocos ejemplos en las clases intermedias. Este desequilibrio hacía que el modelo no pudiera aprender correctamente y tuviera una tendencia a inclinarse hacia las clases más comunes, lo que afectaba su precisión. Debido a esto, optamos por mantener la clasificación binaria, ya que forzar el modelo multi-clase no habría proporcionado resultados útiles.

Intentamos mejorar este problema ofreciendo una probabilidad de compatibilidad basada en la mediana de las características, en lugar de una simple clasificación binaria. Esto permitió que nuestras predicciones fueran más flexibles y que proporcionáramos un rango de probabilidad sobre la compatibilidad de la pareja, aunque aún nos limitó la falta de datos suficientes y equilibrados.

Otro cambio importante que haríamos sería en el diseño de las preguntas del cuestionario. En el proyecto actual, algunas preguntas requieren que ambas personas respondan de manera conjunta, lo que puede generar respuestas influenciadas o no tan objetivas. Para corregir esto, diseñaríamos más preguntas que pudieran responderse de manera individual. Al hacer esto, nos aseguraríamos de obtener respuestas más honestas y personales, que reflejen mejor las percepciones individuales de cada persona en la relación, lo que a su vez mejoraría la calidad de los datos y la precisión del modelo. Este ajuste ayudaría a evitar respuestas sesgadas o influenciadas por la presencia o las expectativas de la pareja.

Otra mejora sería incluir más características demográficas como género, edad y nivel socioeconómico. Esto permitiría realizar un análisis de datos más detallado, capturando mejor las dinámicas entre distintos grupos y cómo estas variables afectan la compatibilidad en las relaciones. Con más características, el modelo podría ofrecer predicciones más personalizadas y precisas.

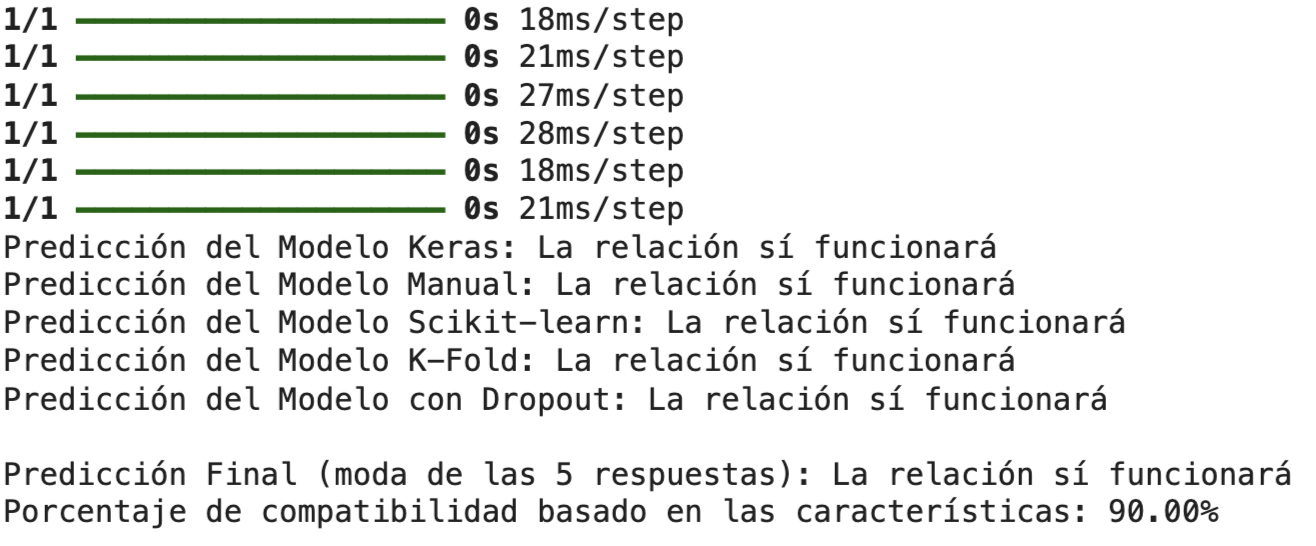
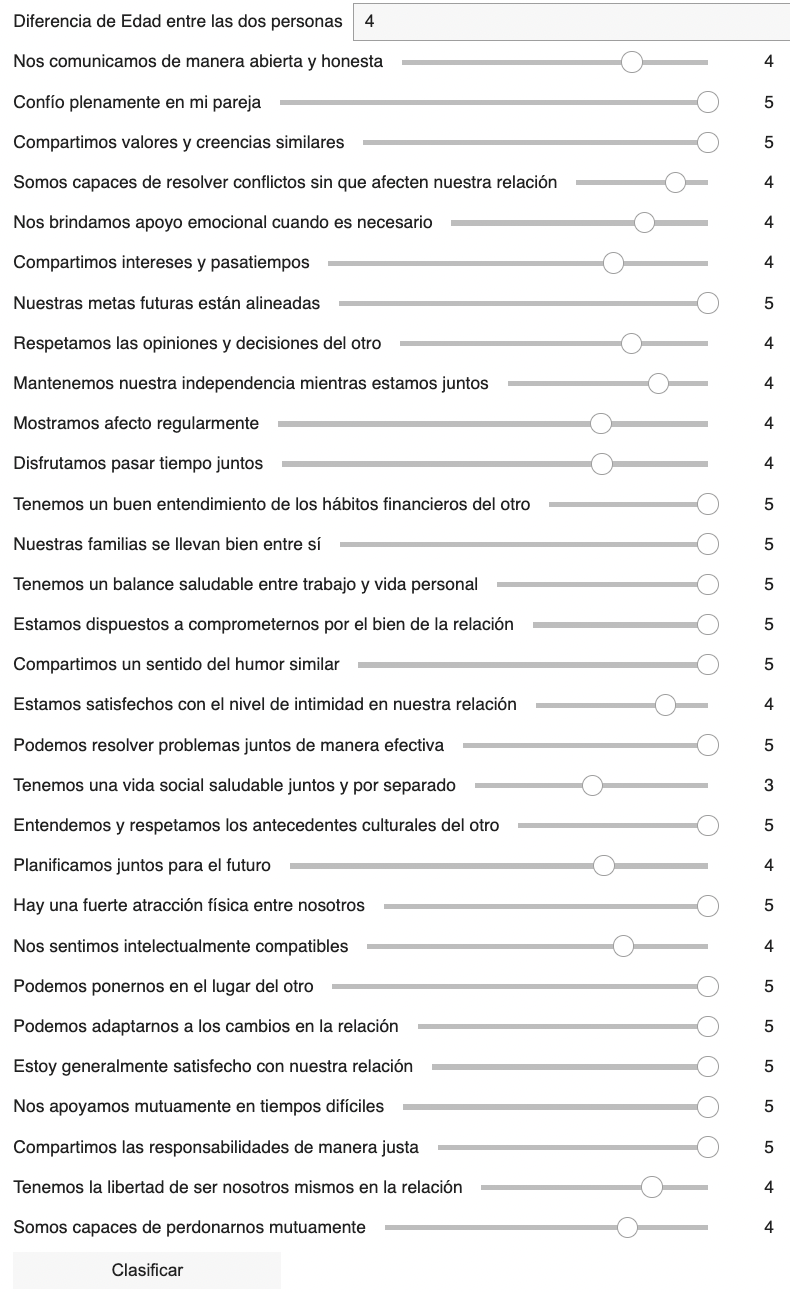
# VII. Interfaz Interactiva

En esta sección del proyecto, diseñamos una interfaz interactiva que permite a los usuarios ingresar sus propias características de relación y obtener una predicción personalizada sobre la compatibilidad de su relación. Utilizando la librería ipywidgets, implementamos una serie de sliders que permiten al usuario seleccionar valores para cada una de las 31 características del modelo.

Una de las principales innovaciones en esta fase fue la implementación de un modelo de ensamble. En lugar de utilizar un solo modelo para realizar la predicción, empleamos cinco modelos entrenados diferentes (Keras, Manual, Scikit-learn, K-Fold y Dropout), y la predicción final se determina a través de la moda de las respuestas proporcionadas por todos los modelos. Esto significa que se toma en cuenta la mayoría de las predicciones realizadas por los distintos modelos para ofrecer un resultado más robusto y confiable. Este enfoque de ensamble mejora la precisión general del sistema, ya que considera múltiples perspectivas antes de tomar una decisión final.

Además de la predicción de "si la relación funcionará" o "no funcionará", implementamos un sistema que calcula el promedio de las características ingresadas para generar un porcentaje de compatibilidad. Este porcentaje brinda una visión más profunda y continua del nivel de compatibilidad entre las personas, lo que permite ofrecer no solo una predicción definitiva, sino también una recomendación basada en la probabilidad estimada de compatibilidad. De esta manera, el usuario no solo recibe una respuesta binaria, sino también una sugerencia ponderada que refleja las fortalezas y debilidades relativas en las diferentes áreas de la relación.

Este enfoque, que combina la predicción de la mayoría de los modelos con un análisis basado en promedios, ayuda a ofrecer una respuesta más completa y útil para el usuario, al darle tanto una decisión final como una probabilidad de éxito, basada en las características clave de su relación.



VIII. CONCLUSIÓN

Este proyecto ha demostrado cómo las técnicas avanzadas de inteligencia artificial, como las redes neuronales profundas y los enfoques de ensamble, pueden aplicarse exitosamente para predecir la compatibilidad en relaciones sentimentales. A lo largo del proceso, recopilamos y preprocesamos un conjunto de datos con 31 características clave de relaciones, las cuales fueron fundamentales para entrenar tres modelos diferentes: un modelo con Keras, un modelo manual y un modelo con Scikit-learn, evaluados con métricas estándar de clasificación.

Durante la segunda fase, se realizaron ajustes importantes para mejorar la precisión y robustez del modelo, incluyendo la eliminación de características irrelevantes, el uso de técnicas de regularización como Dropout, la implementación de validación cruzada con K-Fold y la modificación del umbral de clasificación. Estos cambios ayudaron a mejorar el rendimiento del modelo, minimizando errores comunes como el sobreajuste y optimizando su capacidad de generalización.

Además, desarrollamos una interfaz interactiva que permite a los usuarios ingresar sus características y obtener una predicción personalizada de compatibilidad, basada en un sistema de ensamble de cinco modelos diferentes. Este enfoque, junto con la presentación de un porcentaje de compatibilidad, brinda una respuesta más matizada y completa que solo una simple clasificación binaria.

A pesar de estos avances, el proyecto también identificó áreas de mejora, como la necesidad de un conjunto de datos más grande y equilibrado, lo que permitiría una mayor precisión y un mejor ajuste del modelo a casos más diversos. En futuras iteraciones, un aumento del tamaño del dataset y la posibilidad de incorporar más clases de compatibilidad podrían conducir a resultados aún más precisos y útiles.

## Referencias

* Interactive Chaos. (2024). *Adam*.<https://interactivechaos.com/es/manual/tutorial-de-machine-learning/adam>
* Sotaquirá, M. (2018). *¿Qué es el gradiente descendente?*. Codificando Bits.<https://www.codificandobits.com/blog/el-gradiente-descendente/>
* Sotaquirá, M. (2018). *La función de activación*. Codificando Bits.<https://www.codificandobits.com/blog/funcion-de-activacion/>