**Resumen de lo que se ha hecho:**

Resumen del Notebook 1: "Edición por Noelia\_Primer borrador\_Predicción de enfermedades cardiovasculares"

Este notebook se centra en la predicción de enfermedades cardiovasculares utilizando diferentes técnicas de modelado y preprocesamiento de datos.

**1. Carga y Exploración de Datos:**

- Se cargó el dataset "Framingham" y se realizó una exploración inicial para entender la estructura de los datos.

- Se verificaron y manejaron valores faltantes, y se realizaron transformaciones iniciales en las variables categóricas.

**2. Análisis Exploratorio de Datos (EDA):**

- Se llevaron a cabo análisis univariados para entender la distribución de variables individuales a través de histogramas y gráficos de densidad.

- Se realizaron análisis bivariados, incluidos gráficos de caja y violín, para explorar las relaciones entre las variables predictoras y la variable objetivo `TenYearCHD`.

- Se calculó la matriz de correlación y se visualizó con un heatmap para identificar posibles relaciones entre las variables numéricas.

**3. Ingeniería de Características:**

- Se crearon nuevas características, como grupos de edad, interacción entre presión arterial sistólica y diastólica, y el ratio colesterol/edad.

- Se aplicaron técnicas de ingeniería avanzada de características, como la creación de características polinómicas para capturar relaciones no lineales entre las variables.

**4. Preprocesamiento de Datos:**

- Se aplicaron transformaciones de preprocesamiento utilizando pipelines de scikit-learn, que incluyeron imputación de valores faltantes, escalado de características numéricas y codificación de variables categóricas.

- Se balancearon las clases utilizando la técnica SMOTE para manejar el desbalance en la variable objetivo.

**5. Modelado y Evaluación:**

- Regresión Logística: Se entrenó un modelo de regresión logística y se evaluó utilizando precisión, matriz de confusión, y la métrica ROC-AUC. Este modelo mostró un `accuracy` de 86% pero con un desempeño pobre en la clase minoritaria (TenYearCHD = 1).

- Random Forest: Después del balanceo de clases, se entrenó un modelo de Random Forest con ajuste de hiperparámetros mediante `GridSearchCV`. Este modelo mostró un mejor rendimiento en comparación con la regresión logística, y se identificaron las características más importantes que influyen en la predicción.

**6. Conclusiones:**

- El modelo de Random Forest mostró un rendimiento superior, especialmente después del ajuste de hiperparámetros y el balanceo de clases.

- Se identificaron características clave que tienen mayor influencia en la predicción de enfermedades cardiovasculares, lo que puede ser útil para futuras investigaciones o intervenciones clínicas.

**Diferencias Clave con Otros Notebooks:**

**1. Enfoque del Modelado:**

- En este notebook, se utilizó un enfoque de Random Forest con ajuste de hiperparámetros, que resultó en un modelo más robusto y con mejor rendimiento en comparación con los intentos de redes neuronales en los otros notebooks.

**2. Balanceo de Clases:**

- Aquí se utilizó SMOTE para abordar el problema del desbalance de clases, lo cual no se hizo de manera tan efectiva en los notebooks de redes neuronales.

**3. Análisis de Importancia de Características:**

- Se realizó un análisis detallado de la importancia de las características en el modelo de Random Forest, proporcionando insights adicionales sobre los factores que más influyen en la predicción.

**Recomendaciones para los Compañeros:**

A partir de este notebook, podríamos centrarnos en:

- Refinar el modelo de Random Forest: Aprovechando el buen rendimiento mostrado, se puede explorar la combinación de Random Forest con otras técnicas de ensamble.

- Explorar más características: Basado en el análisis de importancia de características, se pueden crear nuevas variables o ajustar las existentes.

- Validación cruzada: Implementar validación cruzada para asegurar la estabilidad del modelo y su capacidad de generalización.

**Resumen del Notebook 3: "Edición3\_Predicción de enfermedades cv\_red neuronal.ipynb"**

Este notebook se centró en la creación y evaluación de un modelo de red neuronal para predecir enfermedades cardiovasculares.

**1. Importación de Bibliotecas:**

- Se importaron las bibliotecas necesarias, incluyendo `pandas`, `scikit-learn`, `TensorFlow`, y `Keras`, entre otras.

- Se configuraron estilos gráficos y se imprimieron las versiones de las bibliotecas utilizadas.

**2. Carga y Visualización del Dataset:**

- Se cargó el dataset `framingham\_clean.csv` y se realizó una visualización inicial de los datos.

- Se verificaron los tipos de datos y se analizaron estadísticas descriptivas de las características.

**3. Preprocesamiento de los Datos:**

- Se realizó una limpieza básica de datos, incluyendo la conversión de algunas variables categóricas en numéricas.

- Se identificaron y gestionaron valores faltantes.

- Se aplicó balanceo de clases utilizando la técnica SMOTE para abordar el desbalance de la variable objetivo.

**4. Exploración de Datos (EDA):**

- Se realizaron análisis univariados y bivariados para explorar la distribución de las variables y su relación con la variable objetivo `TenYearCHD`.

- Se generó una matriz de correlación y un heatmap para visualizar las relaciones entre las variables numéricas.

**5. Análisis de Clustering:**

- Se aplicó PCA (Análisis de Componentes Principales) para reducir la dimensionalidad de los datos.

- Se realizó un clustering utilizando K-Means para explorar posibles agrupaciones en los datos.

**6. Preparación de Datos para el Modelado:**

- Se separaron las características (X) y la variable objetivo (y).

- Se preprocesaron las características utilizando pipelines para manejar tanto variables numéricas como categóricas.

**7. Implementación de la Red Neuronal con Ajuste de Hiperparámetros:**

- Se definió una red neuronal utilizando Keras, con varias capas densas (`Dense`) y Dropout para evitar el sobreajuste.

- Se utilizó `RandomizedSearchCV` para realizar una búsqueda de hiperparámetros, incluyendo el tamaño del batch, el número de épocas, el optimizador, y la inicialización del kernel.

- Los mejores hiperparámetros encontrados fueron: `rmsprop` como optimizador, `normal` como inicializador del kernel, 200 épocas, y batch size de 16.

**8. Evaluación del Modelo:**

- El mejor modelo fue evaluado en el conjunto de prueba, obteniendo un `accuracy` de aproximadamente 69.35%.

- Se calculó la pérdida (`loss`) en el conjunto de prueba, la cual fue 0.669.

**Conclusiones y Diferencias Clave con Otros Notebooks:**

**1. Desempeño de la Red Neuronal:**

- A pesar de los esfuerzos por ajustar los hiperparámetros, la red neuronal no logró un rendimiento destacado. El accuracy obtenido (69.35%) fue inferior al de otros modelos como Random Forest.

**2. Comparación con Otros Modelos:**

- En comparación con los otros notebooks, donde se utilizó Random Forest y se alcanzaron mejores resultados, la red neuronal no se comportó tan bien, lo que sugiere que este modelo puede no ser el más adecuado para este conjunto de datos.

**3. Desafíos Encontrados:**

- El proceso de ajuste de hiperparámetros y la complejidad del modelo pueden haber contribuido a la menor precisión. El sobreajuste también podría ser un problema, a pesar del uso de Dropout.

**Recomendaciones para Integrar los Resultados:**

Dado que los resultados de la red neuronal no fueron los esperados, se recomienda:

- Enfocarse en el modelo de Random Forest, que ha demostrado un mejor rendimiento.

- Documentar los resultados de la red neuronal como parte del proceso de investigación, pero centrar los esfuerzos en mejorar y optimizar Random Forest.

- Explorar ensambles: Si se desea combinar modelos, se podría explorar la integración de Random Forest con otros modelos, aunque la red neuronal, en este caso, no parece agregar valor significativo.

**Resumen del Notebook 4: "E4PECV\_Redes neuronales.ipynb"**

Este notebook se centra en la optimización y evaluación de un modelo de red neuronal para predecir enfermedades cardiovasculares.

**1. Importación de Bibliotecas y Configuración del Entorno:**

- Se importaron las bibliotecas necesarias, incluyendo `pandas`, `numpy`, `scikit-learn`, `tensorflow`, y `Keras`.

- Se instalaron y actualizaron varias bibliotecas, como `scikeras` y `tensorflow`, para asegurar la compatibilidad con el entorno de Google Colab.

**2. Carga y Preprocesamiento del Dataset:**

- Se cargó el dataset `framingham\_clean.csv` y se reemplazaron valores "Missing" con `NaN`, los cuales fueron posteriormente rellenados con la mediana de cada columna.

- Las variables categóricas fueron convertidas a numéricas, y los datos fueron estandarizados utilizando `StandardScaler`.

- Se dividió el dataset en conjuntos de entrenamiento y prueba (70% y 30%, respectivamente).

**3. Definición y Entrenamiento de la Red Neuronal:**

- Se definió un modelo de red neuronal utilizando `Keras`, con dos capas densas (`Dense`) y una capa de salida con activación `sigmoid`.

- El modelo fue compilado con `binary\_crossentropy` como función de pérdida y `accuracy` como métrica.

- Se entrenó el modelo en el conjunto de entrenamiento con 100 épocas y un tamaño de batch de 32.

- Los resultados del entrenamiento mostraron una precisión (`accuracy`) en el conjunto de validación de aproximadamente 85% y una pérdida (`loss`) de alrededor de 0.39.

**4. Optimización de Hiperparámetros:**

- Se intentó optimizar los hiperparámetros del modelo (número de neuronas, optimizador, épocas y tamaño del batch) utilizando un enfoque manual.

- Se probaron diferentes configuraciones de hiperparámetros con los optimizadores `Adam` y `RMSprop`.

- La mejor precisión (`accuracy`) obtenida fue 85.14% utilizando `Adam` como optimizador, 12 neuronas, 50 épocas, y un batch size de 16.

**5. Evaluación del Modelo:**

- El mejor modelo fue evaluado en el conjunto de prueba, obteniendo una precisión (`accuracy`) de 84.90% y una pérdida (`loss`) de 0.39.

- Se realizó un bucle para probar diferentes combinaciones de hiperparámetros y se confirmó que la configuración mencionada anteriormente era la óptima.

**Conclusiones y Diferencias Clave con Otros Notebooks:**

1. Desempeño de la Red Neuronal:

- Este notebook logró mejorar la precisión del modelo de red neuronal en comparación con intentos anteriores, alcanzando un `accuracy` cercano al 85%.

- La optimización manual de hiperparámetros resultó en una configuración que proporcionó los mejores resultados hasta el momento para una red neuronal.

2. Comparación con Otros Modelos:

- Aunque se logró una precisión razonable con la red neuronal, sigue siendo relevante comparar estos resultados con otros modelos, como el Random Forest en otros notebooks, que podría haber mostrado un rendimiento superior.

3. Desafíos Encontrados:

- Se realizaron múltiples instalaciones y actualizaciones de bibliotecas debido a problemas de compatibilidad en el entorno de Colab, lo que podría haber afectado la reproducibilidad del modelo.

**Recomendaciones para Integrar los Resultados:**

Dado que este notebook logró mejorar el rendimiento del modelo de red neuronal, se recomienda:

- Comparar directamente este modelo con el modelo de Random Forest obtenido en el primer notebook para decidir cuál debería ser el modelo final.

- Documentar claramente el proceso de optimización y los resultados obtenidos para que se pueda reproducir fácilmente el mejor modelo en otros entornos.

- Considerar el uso de técnicas de ensamble si se desea combinar la red neuronal con otros modelos, como el Random Forest, para intentar mejorar aún más el rendimiento.