Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

TC3006C. Inteligencia artificial avanzada para la ciencia de datos I
Grupo 101

Momento de Retroalimentación Módulo 2 :

"Análisis y Reporte Sobre el Desempeño del Modelo"

Evidencia presentada por el estudiante :

Javier Suarez Duran A01707380

Profesores :

Benjamín Valdés Aguirre
José Antonio Cantoral Ceballos
Carlos Alberto Dorantes Dosamantes
Denisse L. Maldonado Flores
Alejandro Fernández Vilchis

Fecha de entrega :

Domingo 10 de Septiembre de 2023

Resumen — En este reporte se analizará el desempeño del modelo de machine learning desarrollado en la actividad "Momento de Retroalimentación: Módulo 2: Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución." para conocer del desempeño conseguido por el modelo en la actividad y mejorar en la medida de lo posible. El programa con el modelo utiliza el lenguaje de programación "Python" y puede ser encontrado en el siguiente repositorio de GitHub: https://github.com/Javier1257/T3006C-ML_Model_With_Specialized_Framework.git
Palabras clave — *modelo, análisis, desempeño, machine learning, GitHub.*

I. Introducción

El modelo a ser analizado es una red neuronal multicapa cuya función es determinar qué pasajeros del gran barco "Titanic" sobrevivieron al fatídico suceso por el que pasó el navío; utilizando métricas y gráficas se pondrá a prueba el modelo de manera que se pueda medir con la mayor certeza posible el desempeño del mismo con el objetivo de conocer si ¿hay mejora posible?, o si es así ¿cuáles deben de ser las implementaciones por ser agregadas al modelo para que mejore?.

II. Descripción del modelo

Para poder analizar el modelo lo principal es conocerlo, así que partiendo de esta base retomaremos lo mencionado anteriormente lo cual es que se trata de una red neuronal multicapa, el modelo en total cuenta con cuatro capas, una capa entrada, una de salida y dos capas internas; entrando en detalle del modelo comenzamos por la primera capa que es la de entrada la cual cuenta con nueve neuronas dado que para el problema que se está resolviendo es el número de variables que se determinaron como entrada después de realizar el respectivo ETL (Explore - Transform - Load) al set de datos del problema, también cabe mencionar que los datos fueron estandarizados y que la distribución del mismo fue de un 70% entrenamiento, 15% validación y 15% para evaluación pero continuando con la descripción de la capa también nos encontramos con un bias y que utiliza "relu" como función de activación; ahora para el caso de la segunda capa esta tiene 32 neuronas, "relu" una vez más como función de activación, y "Regularización L2" para evitar overfitting en el modelo; en la tercera capa se repite el mismo proceso que en su predecesora, 32 neuronas, "relu" como función de activación y "Regularización L2" para evitar overfitting en el modelo; por último la capa de salida solo contiene una única neura y utiliza una sigmoide como función de activación.

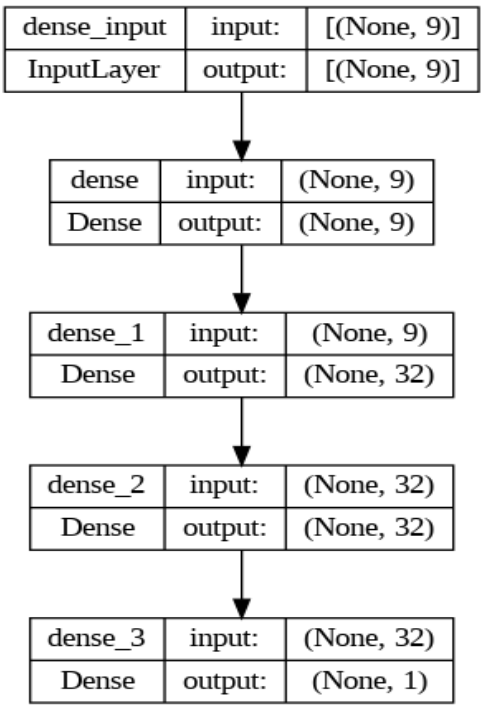
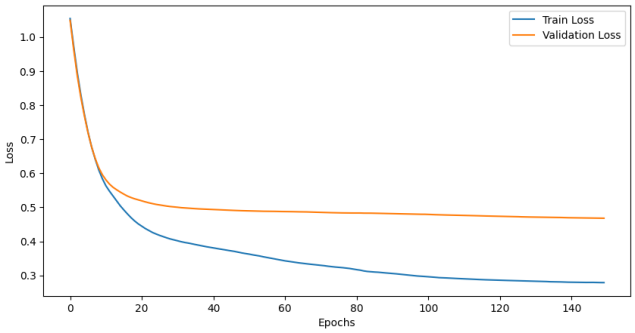


Imagen representativa del Modelo Generada por Código

El modelo fue entrenado por medio de "MiniBatches" de 32, a través de 150 épocas con un Learning Rate de 0.001, función de optimización "Adam", un valor de Regularización L2 de 0.01 y una función de pérdida "Binary - Cross Entropy" dado que solo se espera predecir una clase

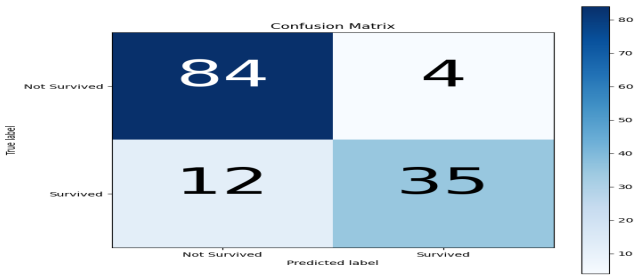
III. Análisis del Modelo

Tras el entrenamiento del modelo se generó la siguiente gráfica de pérdida en relación al set de datos de entrenamiento y validación:



Gráfica de pérdidas generada por código

En la que se puede apreciar como el valor para ambas medidas se encuentra debajo de uno, la sin embargo la línea azul que representa al set de datos del entrenamiento llega aún más abajo indicando un muy bajo sesgo en el modelo,aunado a esto la linealidad en la gráfica también sugiere una muy poca varianza así que por lo que se ve es claro que el modelo se adaptó bien a set datos y aprendió adecuadamente; y lo cual se verá reflejado en el siguiente gráfico:



Matriz de Confusión generada por código

El gráfico el cual es una matriz de confusión se pueden ver los verdaderos y falsos positivos y negativos que hubo en las predicciones del modelo durante su evaluación y con la cuál se pueden obtener las siguientes métricas:

- Precision: 0.90
- Accuracy: 0.88
- Recall: 0.74
- F1 Score: 0.81

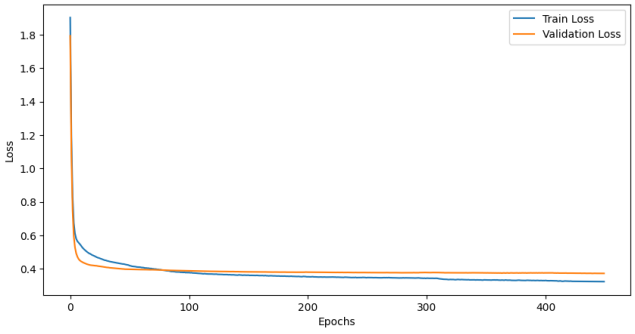
Las cuáles nos indican que el 90% de las veces las predicciones positivas realizadas son correctas, el 74% de las predicciones positivas son correctas en relación a las predicción positivas reales, que el 88% de las veces las predicciones son correctas y por último que el 81% de las veces las respuestas correctas pueden ser correctas en relación a las respuestas reales; por lo que generalizando lo que estas métricas dicen del modelo es que en general desarrolla un buen desempeño, por lo que se puede decir que el modelo en general realmente aprendió y se ajustó ("fit") así que está trabajando adecuadamente y logra buenas predicciones .

IV. Mejora

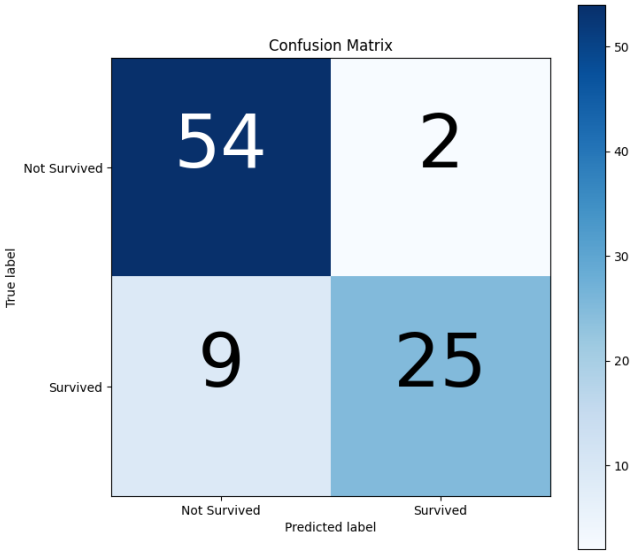
El modelo en general hace un buen desempeño y logra satisfactoriamente un buen margen de predicciones correctas, sin embargo a pesar de esto también es cierto que aún hay un pequeño margen de mejora en el modelo de manera que se puedan mejorar los actuales valores en las métricas de desempeño del modelo y tomando en cuenta lo realizado durante el proceso de programación y entrenamiento del modelo lo principal es modificar sus hiperparametros hasta encontrar valores que hagan que el modelo logre pasar de una buen aprendizaje a un mejor aprendizaje en base a los datos; proceso por el cual paso el modelo hasta llegar al actual punto

en el que se encuentra, por lo que el siguiente paso a realizar será generar un modelo más complejo, no excesivamente pero agregar una capa puede ser la solución necesaria para hacer que modelo mejore, y en caso de que esto no funcione las siguiente prueba a realizar será realizar un ajuste en el set de datos que se utiliza para entrenar el modelo de manera que pueda aprender más.

Por consiguiente después de haber hecho las pruebas pertinentes y ajustado los hiper parámetros en Learning rate a 0.003, Regularizador L2 a 0.03, además de haber sido agregada una capa extra con 32 neuronas el resultado fue el siguiente:



Gráfica de pérdidas generada por código



Matriz de Confusión generada por código

Con las siguientes métricas:

- Precision: 0.93
- Accuracy: 0.88
- Recall: 0.74
- F1: 0.82

Por lo que se puede apreciar que en términos generales no se realizo una diferencia significativa entre el nuevo modelo entrenado y el anterior, así que la siguiente conclusión que se puede obtener al respecto es que para continuar mejorando el modelo es posible que se requieran más datos para que el modelo aprenda y hace sentido porque este solo está aprendiendo con aproximadamente 700 instancias de un dataset de casi 900 instancias, sin embargo a esto también se une la posibilidad de transformar los datos de otra manera para que el modelo encuentre una mejor manera de identificar los patrones en el.

V. Conclusión

En conclusión el proceso realizado durante el desarrollo del análisis ha sido importante para estudiar y comprender más acerca del modelo en este caso una red neuronal multicapa, sin embargo el proceso aplica para todos los modelos de machine learning de manera que se puede llegar a obtener los mejores resultados posibles y así poder desplegar estas importantes soluciones que facilitan la realización de todo tipo de tareas y problemas; por lo que conocer las métricas y gráficas que nos ayudan a visualizar de mejor manera lo que pasa dentro del modelo es de vital importancia para su continua mejora.

VI. Anexos

Github del Modelo con la Red Neuronal Analizada:
https://github.com/Javier1257/T3006C-ML_Model_With_Specialized_Framework.git