Universidad Adolfo Ibañez
Master of Science in Data Science
Nombre: Alfonso Tobar

Reporte 1: MultiLayer Perceptron

Introducción

El siguiente reporte contiene todo el análisis realizado en el *The Forest Cover Type dataset*. Este dataset contiene información extraída mediante mediciones en terreno en el Parque Nacional Roosevelt y ordenada y disponibilizada por la Universidad de **Colorado State**. El dataset fue encontrado como parte de una competencia en Kaggle. La razón de la elección es porque tiene relación con mis temas de investigación en Percepción Remota.

El objetivo de este Reporte es estudiar el funcionamiento de un MultiLayer Perceptron con distintos hiperparámetros. El modelo fue entrenado en Pytorch [1] y Pytorch Lightning [2] para la predicción de la Cobertura Forestal. Es decir, dependiendo de distintas *features* medidas se quiere indicar si la Cobertura Forestal corresponde a especies de:

1. **Spruce/Fir**: Abetos.
2. **Lodgepole Pine**: Pino Contorta.
3. **Ponderosa Pine**: Pino Ponderoso.
4. **Cottonwood/Willow**: Sauce.
5. **Aspen**: Álamo Temblón.
6. **Douglas-fir**: Abeto Douglas.
7. **Krummholz**: Tipo de Vegetación Atrofiada.

Debido al gran tamaño de los datos (581.012 registros y 55 columnas) el modelo se validará utilizando un esquema de **Validación Holdout**, con un 80 % de los datos utilizados durante el entrenamiento y un 20 % para la evaluación del modelo. Para determinar la mejor performance se utilizará la métrica de **Accuracy** y **Macro F1 Score**.

1. Multilayer Perceptron

El perceptrón multicapa corresponde a un modelo presentado primeramente por Rosenblatt [3] en 1958. La idea inicial de este algoritmo, inspirado en el funcionamiento de las neuronas cerebrales, era clasificar datos dependiendo de patrones intrínsecos de éstos. El objetivo del algoritmo es que permita el aprendizaje autónomo sin necesidad de darle reglas explícitas y para ello grandes cantidades de datos son necesarias

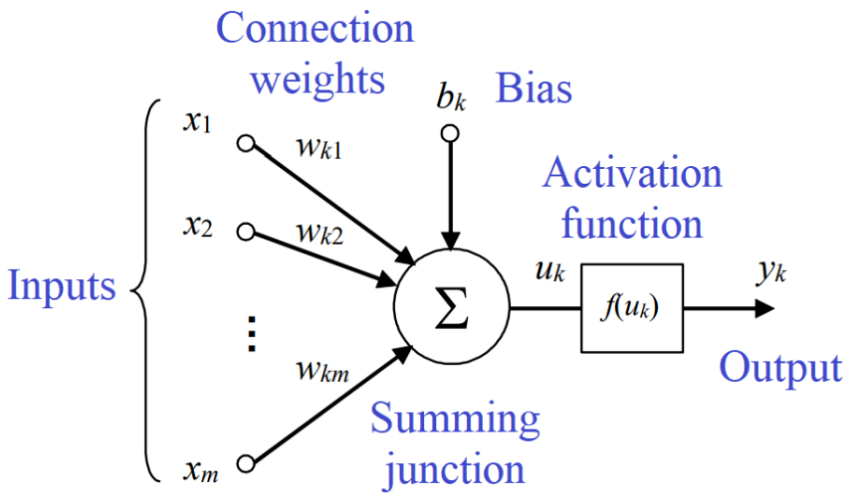

Figura 1: Multilayer Perceptron.

Para que un Perceptrón aprenda, requiere datos de entrada, los cuales van a ser ponderados mediante pesos. Toda esta información se agrega en una neurona y con la ayuda de una función de activación gana capacidades no lineales. La idea del algoritmo es que de manera iterativa, estos pesos vayan siendo ajustados de tal manera de disminuir un error asociado a la diferencia entre el valor real y el valor estimado por la red. El detalle de todo este mecanismo se puede observar en la Figura 1.

2. Los Datos

El Dataset elegido en este proyecto ha sido utilizado en *"Comparison of neural networks and discriminant analysis in predicting forest cover types"*[1] y [4]. Está construido por distintas variables de tipo cartográficas. La cubierta de bosque para una observación en celdas de 30x30 metros fue determinada por el Servicio Forestal de Estados Unidos (USFS). El dataset contiene variables crudas con columnas binarias que indican tipos de suelos y áreas silvestres no intervenidas (wilderness). El estudio contempla 4 Áreas Silvestres del parque Nacional Roosevelt que cuentan con perturbaciones humanas mínimas, lo que implica que la cubierta de bosque está definida por procesos ecológicos más que por buenas prácticas en cuidados forestales.

Las áreas silvestres son:

1. **Rawah**
2. **Neota**
3. **Comanche Peak**
4. **Cache la Poudre**

Neota corresponde al área de mayor elevación promedio, seguido por Rawah y Comanche Peak. La menor elevación promedio la tiene Cache la Poudre.

El dataset contiene 581.012 mediciones, 54 features y una Variable Objetivo. Una descripción de las features se encuentra en la siguiente tabla:

Es importante recalcar que para cada observación se puede asignar una única clase, lo que transforma el problema en un tarea de clasificación multiclase.

[1]Esta es una Tesis de Phd, cuya única referencia encontrada fue esta: https://dl.acm.org/doi/10.5555/928509

Descripción de los Datos			
Nombre	Tipo	Unidad	Descripción
Elevation	Continua	Metros	Elevación en Metros
Aspect	Continua	Azimuth	Aspecto en Grados Azimuth
Slope	Continua	Grados	Pendiente en Grados
Horizontal_Distance_To_Hydrology	Continua	Metros	Distancia Horizontal a superficie
Vertical_Distance_To_Hydrology	Continua	Metros	Distancia Vertical a superficie
Horizontal_Distance_To_Roadways	Continua	Metros	Distancia Vertical a Carretera
Hillshade_9am	Continua	Índice de 0 a 255	Índice medido en verano
Hillshade_Noon	Continua	Índice de 0 a 255	Índice medido en verano
Hillshade_3pm	Continua	Índice de 0 a 255	Índice medido en verano
Horizontal_Distance_To_Fire_Points	Continua	Metros	Distancia Horizontal a Incendios
Wilderness_Area (4 Columnas)	Nominal	Binaria	Indicador de Área
Soil_Type (40 Columnas)	Nominal	Binaria	Indicador de Suelo
Cover_Type (7 Tipos)	Nominal	1 a 7	Indicador de Cubierta

3. Análisis Exploratorio

Al hacer una revisión del Dataset se puede ver que el vector objetivo (Target Variable) contiene un claro desbalance. La distribución de las clases se puede observar en la Figura 2:

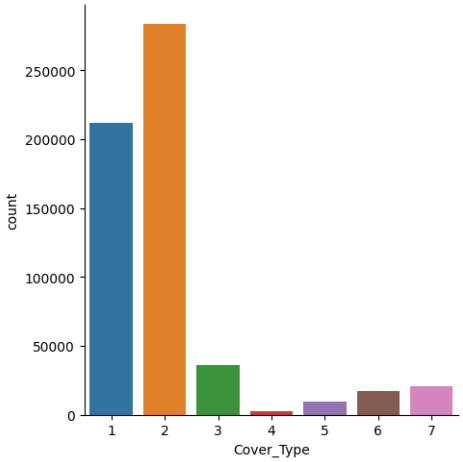

Figura 2: Distribución de Coberturas.

Es posible ver una proporción mucho mayor de las clases 1 y 2 (Abetos y Pino contorta). Esto es de esperar ya que son las especies de árboles más comunes en el parque. Por otra parte, la clase 4 (Sauce) es la menos abundante.

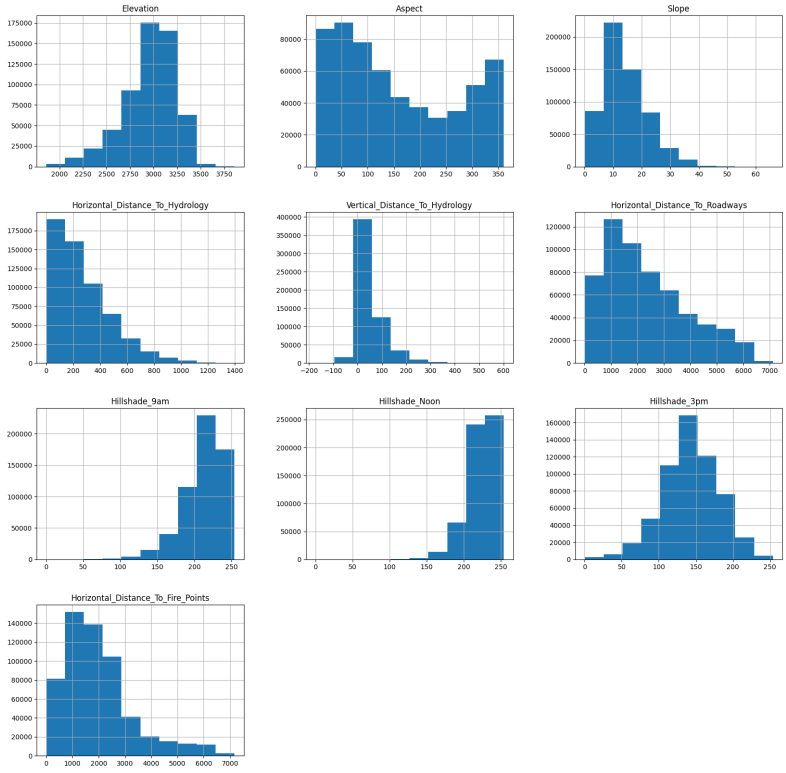

Figura 3: Distribución Features Numéricas.

La Figura 3 muestra la distribución de los datos numéricos. Al analizar los gráficos se pueden ver tanto distribuciones seudo normales como en el caso de Elevation, Hillshade_3pm y Vertical_Distance_To_Hydrology y otros casos caso muy asimétricos como Horizontal_Distance_To_Hydrology, Slope, Hillshade_9am y Hillshade_Noon. Es de esperarse bastante variabilidad debido a que los puntos fueron medidos en partes diversas del Parque Roosevelt.

Por otra parte, la relación entre las variables numéricas y el vector objetivo se puede ver en la Figura 4:

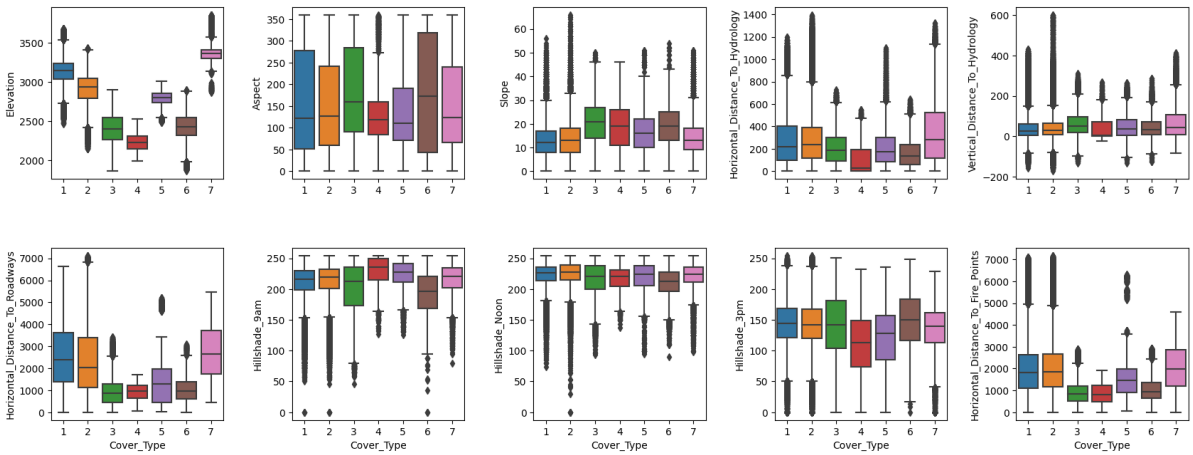

Figura 4: Relación Features Numéricas con Vector Objetivo.

De acá se desprende que quizás variables como Elevation, Slope y Horizontal_Distance_to_Roadways son claras features que permiten diferenciar los distintos tipos de cubierta. Otras variables como Hillsha-

de_9am y Hillshade_Noon se ven muy similares entre ellas y quizás hasta redundantes. Variables como Vertical_Distance_To_Hydrology parecen no ser muy diferenciadores del tipo de cubierta forestal.

Finalmente podemos observar la relación existente entre los distintos tipos de suelo y áreas silvestres por medio de la Figura 5:

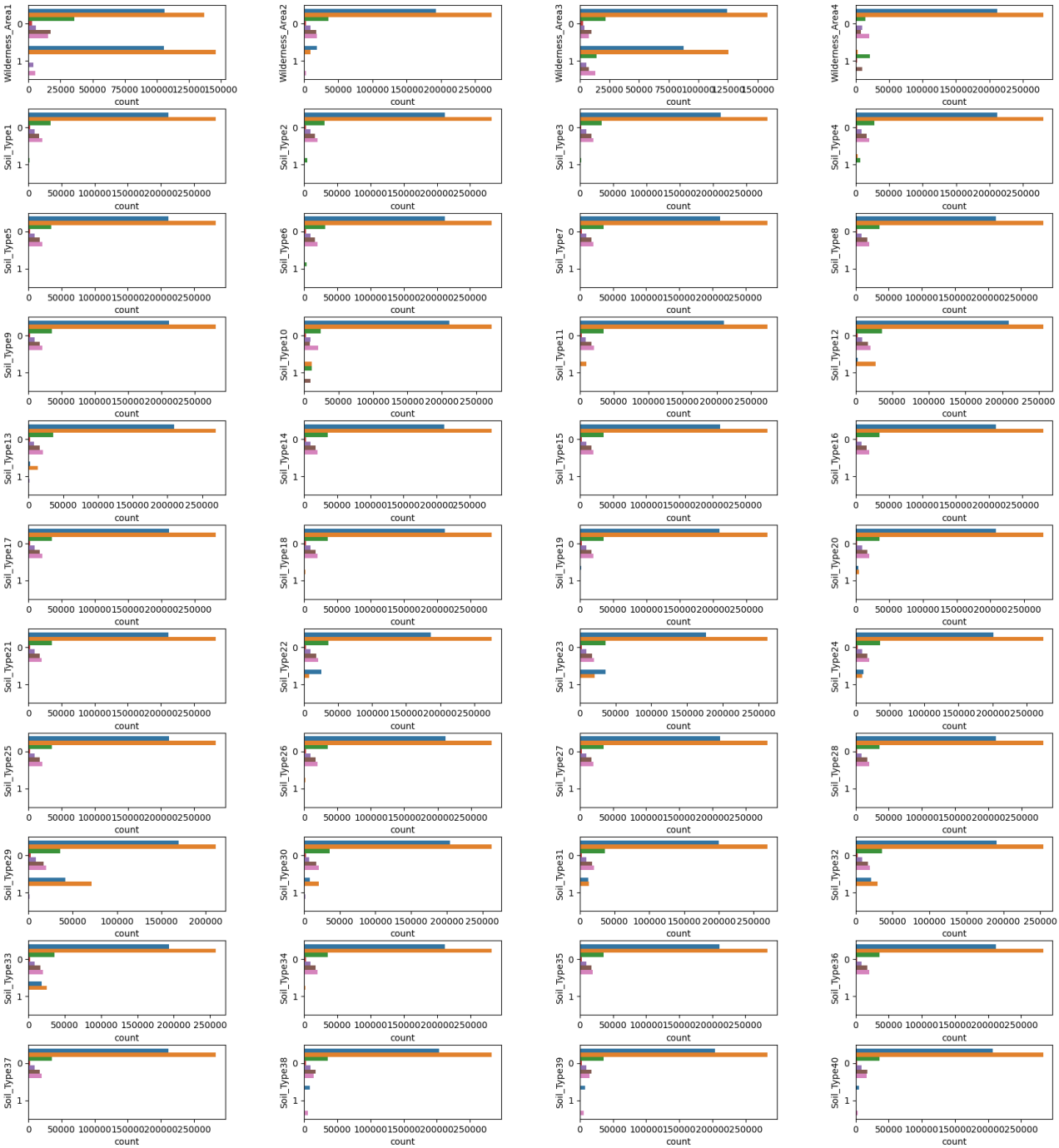

Figura 5: Relación Features Categóricas con el Vector Objetivo.

Este tipo de relaciones es un poco más dificil de interpretar dado que algunas cubiertas como la 1 y 2 son casi omnipresentes y se dan siempre. Debido a la alta complejidad en la interpretación de estas variables es que se espera que la Red Neuronal propuesta pueda entender qué features son las que permiten predecir la

Cubierta Forestal de mejor manera y permita generar interacciones apropiadas para un buen desempeño del modelo.

4. Modelamiento

4.1. Selección de Variables

Antes de comenzar con el proceso de Modelamiento es importante chequear si las variables que se tienen son significativas. Para determinar esto utilizaremos inicialmente una matriz de correlación para Features Numéricas (Ver Figura 6) para chequear si es que existen features redundantes.

Figura 6: Matriz de Correlación de Features Numéricas.

Por otro lado para el caso de las Features Categóricas se utilizará un filtrado por varianza. Varianzas muy pequeñas puede implicar un valor constante en los datos, lo cual no debería aportar al modelo. Los resultados se pueden ver en la Figura 7:

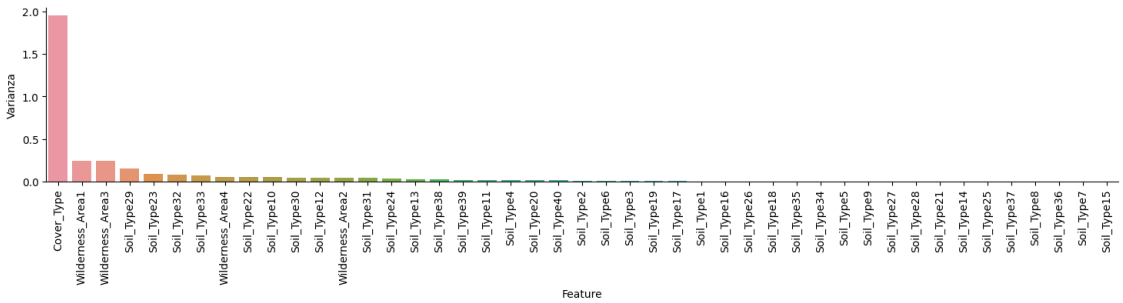

Figura 7: Varianza de Features Categóricas.

A partir de estos dos procesos se concluye que existen variables que quizás valga la pena eliminar. Por ejemplo dado su bajo nivel de varianza se decide eliminar Soil_Type2 y todas las que tengan una varianza menor a ella. Por otro lado las variables más correlacionadas son Aspect y Hillshade_9am y Hillshade_3pm, además de Slope con Hillshade_Noon. Debido a la poco relevancia observada en el EDA se decide eliminar Hillshade_3pm y Hillshade_Noon. A partir de este análisis se probarán dos set de features: El dataset completo (54 features) y una configuración reducida eliminando las variables con mayor correlación y con poca varianza quedando con 29 features.

4.2. Preprocesamiento

El preprocesamiento realizado a las variables fue bastante sencillo, la Cubierta se llevó de 1 a 7 a 0 a 6. Esto fue necesario ya que Pytorch es 0-based y no permitía clases partiendo en 1. Por otro lado las features fueron escaladas utilizando Z-Score según la Ecuación (1).

$$X_{i(escalado)} = \frac{X_i - \mu_i}{\sigma_i} \tag{1}$$

4.3. Evaluación

El modelo será evaluado en modalidad Holdout utilizando un 80 % para entrenar y un 20 % para validar. Cabe destacar que dado el alto desbalance de clases es que se hace un split estratificado, respetando la representatividad de las clases en el set de entrenamiento y validación.

Las métricas que se utilizarán para evaluar el modelo serán Accuracy y F1 Score. Si bien el Accuracy no es una métrica robusta al desbalance de clase se quiso a usar ya que era una métrica que se utilizaba en la competencia asociada al dataset. Se decide agregar Macro F1 Score debido a que es más robusta al desbalance y una métrica más representativa del problema.

$$Accuracy = \frac{\text{Observaciones Predichas Correctamente}}{\text{Total de Observaciones}}$$

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Cabe destacar que el Macro F1-Score corresponde al promedio del F1 Score de cada una de las clases.

4.4. Arquitectura

Con respecto a la Arquitectura se utilizará una red con características similares a lo que se ve en la Figura 8. Es decir consideraremos un MLP de 4 Capas de Nodos, en el cual conectaremos:

- **n** Neuronas de Entrada. **n** será un Hiperparámetro correspondiente al número de variables a utilizar.
- Se tendrán dos capas ocultas de m1 y m2 neuronas respectivamente. Se probarán distintas configuraciones pero siempre considerando m1 mayor a m2.
- El número k outputs viene definido por las clases a predecir, serán 7 clases (índices 0 a 6, ya que los frameworks lo requieren de esa forma.
- La función de Activación de la última capa será Softmax al tratarse de un problema multiclase. Además se utilizara CrossEntropy como Loss Function.
- Se utilizará ReLU y Tanh como funciones de activación en las capas ocultas.
- Se probará con SGD y Adam como Optimizadores utilizando un learning rate de 0.001.
- Se probarán 1024 y 5096 como batch_size. La razón para elegir batch_size tan altos es debido a que es mucha data y utilizar valores pequeños genera inestabilidad y altos tiempos de entrenamiento.
- Se entrena el modelo por 100 Epochs utilizando EarlyStopping con paciencia de 20 Epochs y Checkpointing, es decir, guarda los pesos del mejor puntaje de validación.

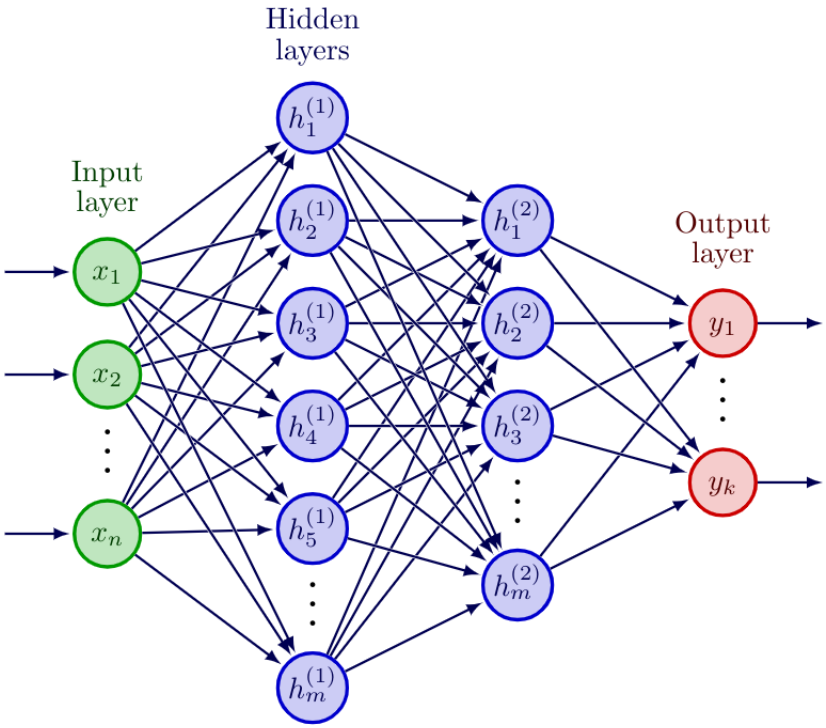

Figura 8: Esquema MLP utilizado.

4.5. Resultados del Modelo

Los resultados del entrenamiento del modelo se pueden encontrar en la Tabla 1.

4.5.1. 1era etapa del Modelo

El proceso de modelamiento partió con una red sencilla de 4 capas con 64 y 32 neuronas en las capas ocultas utilizando ReLU como activación y SGD como optimizador. Se probó el efecto de un batch size de 1024 y 5096 y también se comparó el efecto de eliminar variables redundantes o poco significativas. El

problema de esta configuración es que si bien se obtiene un Accuracy razonable, el F1 Score era demasiado bajo. Al ser este un problema con desbalance de clases el Accuracy no era un buen indicador por lo que se optó por complejizar el modelo.

4.5.2. 2da etapa del Modelo

Se modificó la red a una con más neuronas, 128 y 64 en las capas ocultas. Se probó con la misma configuración de función de activación y optimizador además de los distintos set de features y no hubo muchas mejoras. Los resultados de esta etapa obtienen un f1 Score bajo 0.5 de manera sostenida, lo cual no indicaba buenos augurios.

4.5.3. 3era etapa del Modelo

En esta etapa final se probó el efecto de Tanh como activación lo cual inicialmente no tuvo buenos resultados. Debido a esto se optó por cambiar el optimizador a Adam. Inmediatamente esto trajo una gran mejora utilizando tanto Tanh o ReLU como activación de las capas ocultas.

Finalmente el mejor modelo en el set de validación se obtuvo utilizando todas las variables, Tanh como activación y Adam como optimizador.

n	m1	m2	Función Activación	Optimizador	Batch Size	Accuracy	Macro F1-Score
54	64	32	ReLU	SGD	1024	0.7278	0.4056
29	64	32	ReLU	SGD	1024	0.7209	0.3971
54	64	32	ReLU	SGD	5096	0.6313	0.2322
29	64	32	ReLU	SGD	5096	0.6240	0.2626
54	128	64	ReLU	SGD	1024	0.7303	0.4155
29	128	64	ReLU	SGD	1024	0.7234	0.4056
54	128	64	ReLU	SGD	5096	0.6735	0.2997
29	128	64	ReLU	SGD	5096	0.6698	0.2910
54	128	64	Tanh	SGD	1024	0.7214	0.4113
29	128	64	Tanh	SGD	1024	0.7138	0.4064
54	128	64	Tanh	SGD	5096	0.6971	0.3163
29	128	64	Tanh	SGD	5096	0.6850	0.3185
54	128	64	ReLU	Adam	1024	0.8913	0.8493
29	128	64	ReLU	Adam	1024	0.8916	0.8399
54	128	64	ReLU	Adam	5096	0.8808	0.8114
29	128	64	ReLU	Adam	5096	0.8717	0.7857
54	**128**	**64**	**Tanh**	**Adam**	**1024**	**0.9348**	**0.8958**
29	128	64	Tanh	Adam	1024	0.93	0.8807
54	128	64	Tanh	Adam	5096	0.9189	0.8744
29	128	64	Tanh	Adam	5096	0.9118	0.8506

Tabla 1: Resultados Entrenamiento

5. Conclusiones

Se puede concluir que el MultiLayer Perceptron puede ser un modelo altamente competitivo en problemas de datos tabulares. En particular el Puntaje obtenido es bastante bueno tanto en Accuracy como en Macro F1-Score.

Se pudo concluir que fue una buena decisión confiar más en el puntaje de Macro F1-Score. Se pudo apreciar claramente que el Accuracy entrega una falsa sensación de éxito. Varios modelos presentaron un puntaje de Accuracy bastante descente pero con Macro F-1 paupérrimos confirmando que el Accuracy no es una métrica confiable con altos desbalances de clase.

Algunos aspectos importantes que se notaron en el proceso de Experimentación es que los Hiperparámetros que más influyeron fueron el Número de Neuronas por Capa y el Optimizador. Se hizo sumamente

importante utilizar una cantidad de parámetros acorde al tamaño del dataset y probablemente la configuración (64,32) era muy pequeña para la cantidad de datos disponibles. Por otra parte Adam presentó una mejora muy significativa al proceso de entrenamiento. Probablemente debido a que es un método adaptativo, logró evitar algún mínimo local que el clásico Stochastic Gradient Descent no logró sortear.

Por otro lado, las funciones de activación también influyeron en cierta manera. Curiosamente al utilizar SGD como optimizador ReLU entregó pqueñas mejoras respecto a Tanh, mientras que al cambiar el optimizador por Adam, Tanh demostró una mejora en la performance bastante significativa.

Con respecto al Batch Size, se puede notar que el Batch más pequeño logró mejores resultados sostenidamente. Esto porque genera más actualizaciones de los pesos logrando un descenso del loss function más pronunciada aunque con el costo de un mayor tiempo de entrenamiento.

Increíblemente un parámetro que casi no tuvo relevancia fue la limpieza de variables. Eliminar variables correlacionadas y con poca variabilidad terminó perjudicando al modelo y sostenidamente experimentos con menos variables obtuvieron menor puntaje que con el set de features completo.

Referencias

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[2] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," 2019.

[3] F. Rosenblatt, "The perceptron: A probabilistic model for imformation storage and organization in the brain.," 1958.

[4] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. 24, no. 3, pp. 131–151, 1999.