Reporte 2: Radial Basis Function Neural Networks x MLP

Introducción

El siguiente reporte contiene todo el análisis realizado en el *NYC Taxi Fare Prediction Dataset*. Este dataset contiene información correspondiente a viajes en Taxi en la ciudad de Nueva York, la cual fue disponibilizada a modo de competencia en Kaggle.

El objetivo de este Reporte es comparar el funcionamiento de un MultiLayer Perceptron con distintos hiperparámetros ahora en un problema de Regresión y un Radial Basis Function Neural Net. El modelo fue entrenado en Pytorch [1] y Pytorch Lightning [2] para la predicción de Tarifa del Viaje. Es decir, dependiendo de distintas *features* medidas se quiere indicar un estimado de la tarifa. Esto implica que el problema se modela como una tarea de Regresión.

El tamaño del dataset es de 120.000 registros con 8 columnas.
El proceso de entrenamiento contempla el estudio de varios sets de hiperparámetros validados en un esquema de **validación tipo Holdout** con un 80 % de los datos utilizados para entrenar un y 20 % para validar. Debido a que se trata de un problema de Regresión es que que se utilizará **Root Mean Squared Error** para validar la performance del modelo.

1. Radial Basis Function Neural Net

La Radial Basis Neural Net (RBF) [3] es una variante del perceptrón multicapa, en el cual se intenta emular un campo receptivo. Es una red compuesta por 3 capas conectadas en la cual la capa oculta contiene Radial Basis Functions como funciones de activación. Una de las caracterísiticas más importantes que destaca esta arquitectura es que es un aproximador universal, siendo capaces de aproximar cualquier tipo de función.

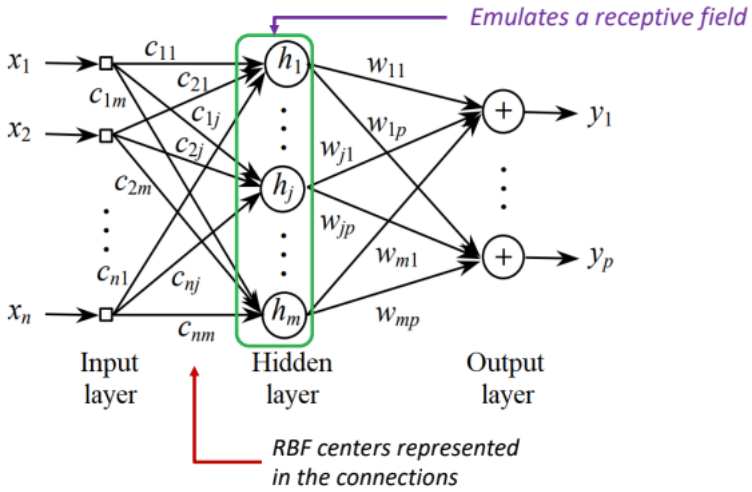

Figura 1: Radial Basis Function Neural Net.

Debido a la gran flexibilidad de utilizar distintos Kernels en la capa oculta, es que puede adaptarse a distintas distribuciones de datos. Algunos de los Kernels más conocidos que se pueden integrar con esta red son los Gaussianos y Multicuadráticos.

El resto del funcionamiento de esta red es bastante similar a un Multilayer Perceptron [4] presentado en el Reporte 1. En este caso particular se utilizará esta arquitectura para la resolución de un problema de Regresión y se comparará su desempeño con un MLP.

2. Los Datos

El dataset escogido se obtuvo de Kaggle desde una competencia Playground llamada NYC Taxi Fare Prediction[1]. El objetivo de esta competencia es predecir el precio del viaje en Taxi. La competencia menciona que con estimaciones de distancias es posible tener un error cercano a US$8. Por lo tanto, el desafío es poder disminuir este valor utilizando Machine Learning.

El Dataset cuenta con las siguientes features:

- **pickup_datetime** - Corresponde al timestamp en el que se inicio el viaje en Taxi.
- **pickup_longitude** - Corresponde a la longitud en grados en la que comenzó el viaje.
- **pickup_latitude** - Corresponde a la latitud en grados en la que comenzó el viaje.
- **dropoff_longitude** - Corresponde a la longitud en grados en la que terminó el viaje.
- **dropoff_latitude** - Corresponde a la latitud en grados en la que terminó el viaje.
- **passenger_count** - Corresponde al número de pasajeros en el viaje.
- **fare_class** - Tipo de Tarifa definida como variable Binaria.
- **fare_amount** - Corresponde al valor a predecir y corresponde al costo del viaje.

El dataset a utilizar corresponde a una muestra del dataset original y cuenta con 120.000 filas y 8 columnas.

3. Análisis Exploratorio

Comenzaremos el análisis entendiendo los viajes. Estos viajes parecen realizarse en un periodo de tiempo comprendido entre el 11 de Abril del 2010 y el 25 de Abril del 2010. Se puede observar en la Figura 2 que el número de viajes no presenta tanta variación a excepción del último día, esto podría deberse a la manera en la que se tomó la muestra. Se pueden ver ciertas tendencias estacionales las cuales se intentarán indagar más a fondo.

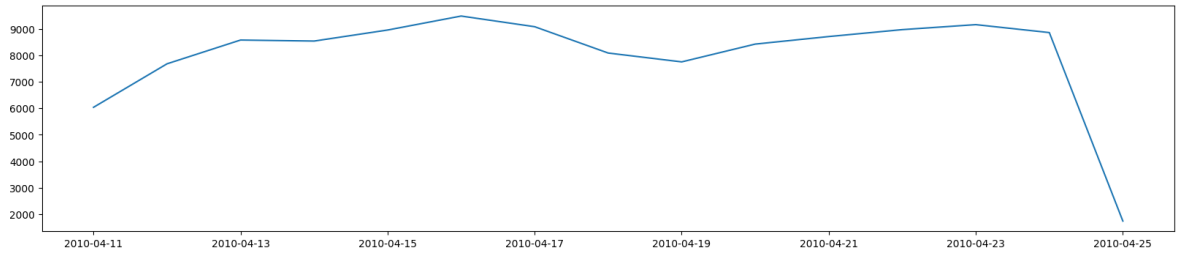

Figura 2: Distribución Temporal de Viajes.

[1] https://www.kaggle.com/competitions/new-york-city-taxi-fare-prediction/overview

La Figura 3 muestra un desglose por día de la semana. Se puede observar que el día con más viajes corresponde a los Viernes, mientras que los días con menos viajes son los Lunes. Además es posible chequear la tarifa que la tarifa es bastante parecida pero con un mayor valor los días Domingo y Jueves. Los días con tarifas más baratas corresponden a los Martes.

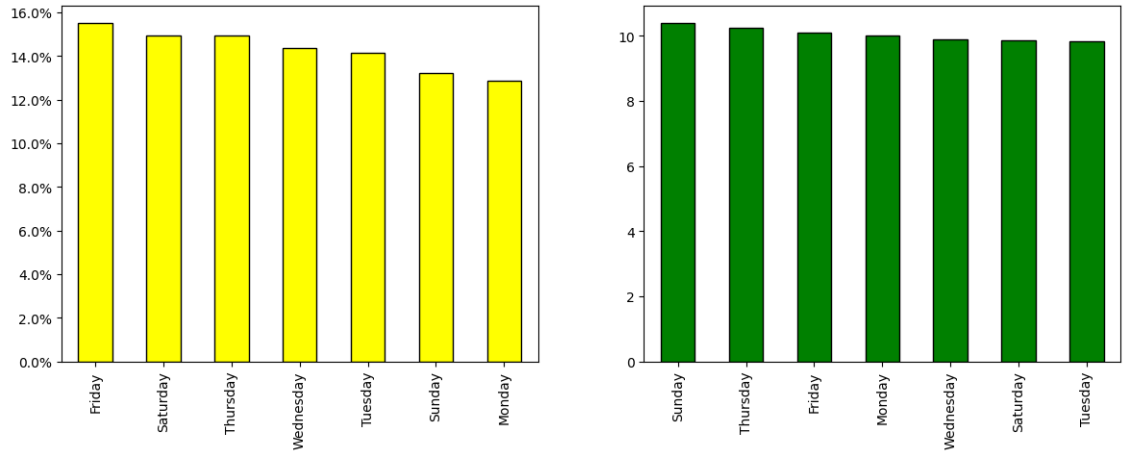

Figura 3: Porcentaje de Viajes por Día (Izquierda) y Precio Promedio de Viajes por Día (Derecha)

Si vamos más al detalle podemos notar que dependiendo de la hora del Viaje la tarifa también cambia significativa. La Figura 4 muestra el cambio de la Tarifa por Hora de Inicio del Viaje. Se puede ver claramente entre las 00:00 y 2:00 se dan las mayores tarifas. Otros peaks de precio se dan a eso del medio día y desde las 18:00 a 20:00 que probablemente corresponde a la salida del trabajo.

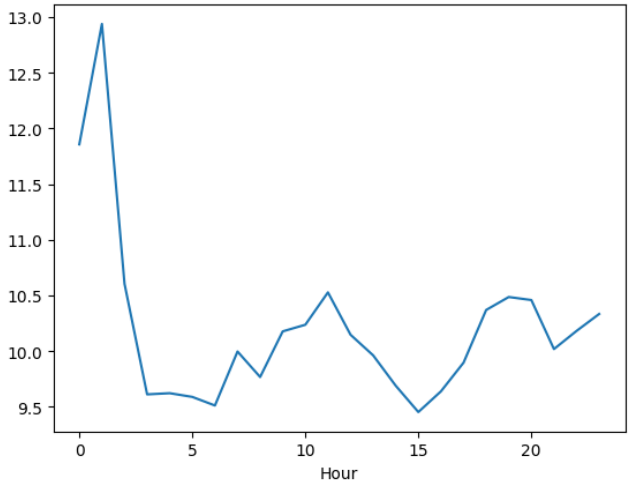

Figura 4: Precio de Viajes por Hora.

Las distribuciones de las variables numéricas se pueden ver en la Figura 7. Se puede observar que el vector objetivo fare_amount tiene una distribución asimétrica hacia la derecha (right-skewed). Una buena manera de lidiar con este tipo de distribución es aplicar una transformación que le permitiera tomar una distribución más cercana a la normal. En la Figura 6 se puede ver el resultado de aplicar una transformación logarítmica a fare_amount. Por otro lado, variables como *_longitud y *_latitude parecen estar fuertemente concentradas. Finalmente passenger_count parece tener una distribución decreciente pero de tipo discreta.

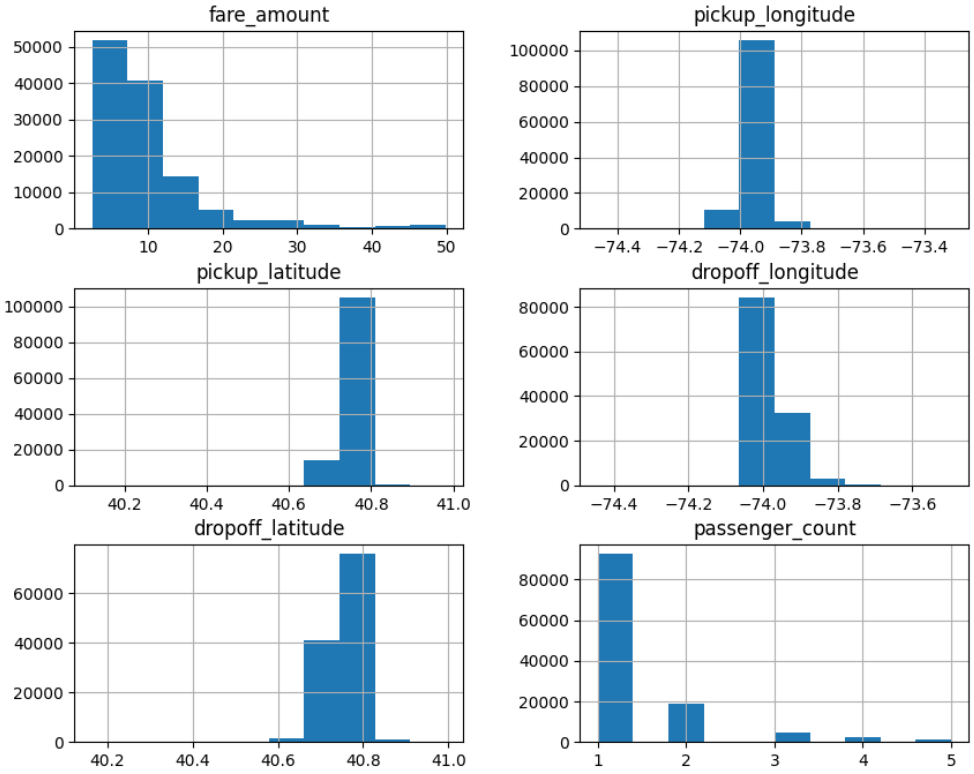

Figura 5: Distribución de Variables Numéricas.

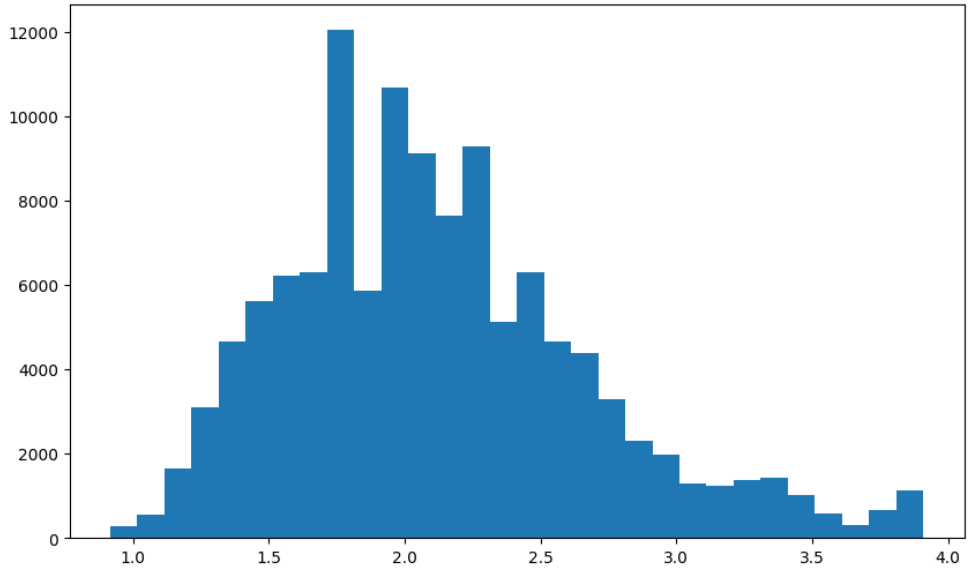

Figura 6: Distribución fare_amount luego de Transformación Logarítmica.

Al ver estos resultados, se considera prudente indagar un poco más en fare_amount debido a que es la variable a predecir. La Figura 7 muestra el efecto de fare_class. Al parecer es un tipo de categoría del precio. La categoría 1 es sólo un precio más alto que la categoría 0.

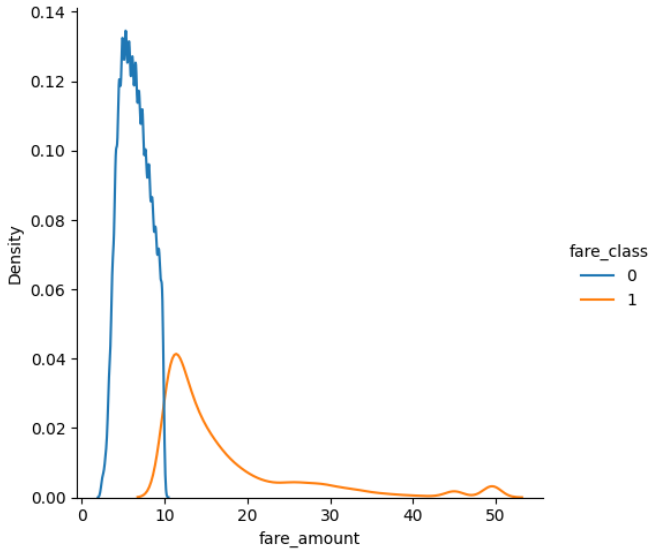

Figura 7: Distribución de Tarifa por fare_class.

Otro aspecto que quizás vale la pena investigar es cuánto influye el número de pasajeros en la tarifa. La Figura 8 muestra que los precios promedios por Número de pasajeros son practicamente idénticos. Sólo cambia la frecuencia en la que se dan los viajes con distinto número de pasajeros. Esto es de esperarse ya que precio del viaje depende más de la distancia de viaje más que el número de pasajero.

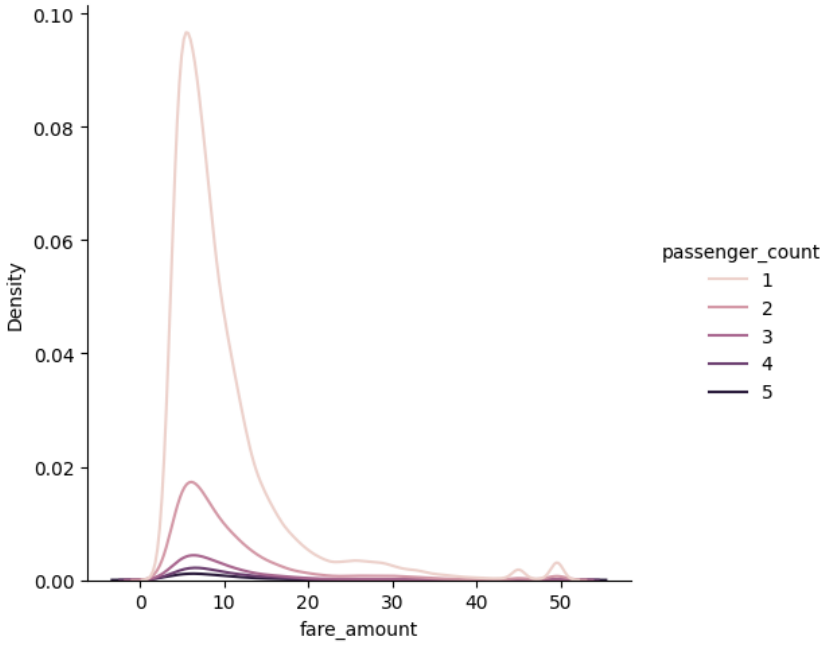

Figura 8: Distribución de Precios por Número de Pasajeros.

Un aspecto importante de investigar es la ubicación de inicio y término del viaje además de los precios asociados a dichos viajes. La Figura 9 muestra que los viajes tienen un costo menor cuando inician desde zona oeste o en el sur. Por otro lado los viajes más caros parecen ser los de la zona noreste. Con respecto a la llegada se puede ver que los que terminan en casi toda la zona tienen un alto precio. Esto nos hace pensar que la mayoría de viajes del dataset se da desde la zona noreste hacia el resto del territorio y son

más caros quizás por la distancia, mientras que los viajes de la zona oeste son más cortos y por eso más baratos.

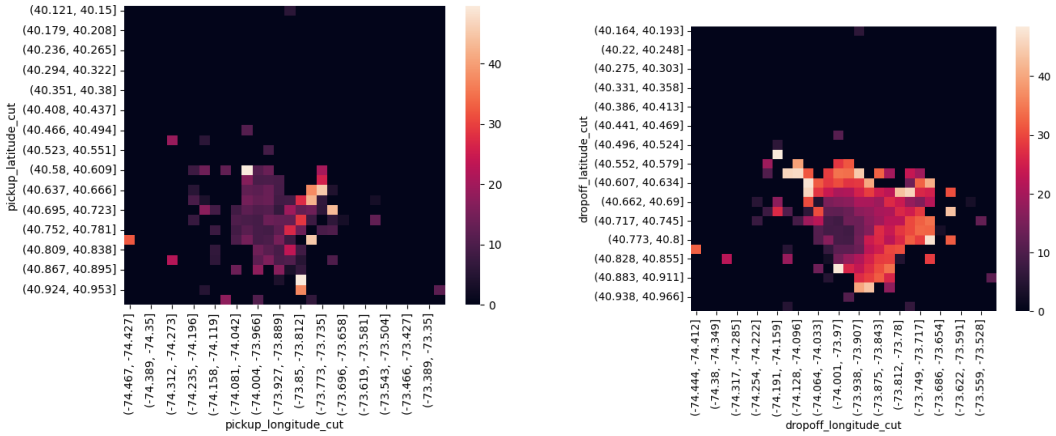

Figura 9: Precio Promedio con respecto al Inicio de Viaje (Izquierda) y Precio Promedio con respecto al Final del Viaje (Derecha).

Obviamente el principal predictor del precio es la distancia. En la Figura 10 se puede ver que la tarifa a pagar tiene una altísima correlación con la distancia recorrida.

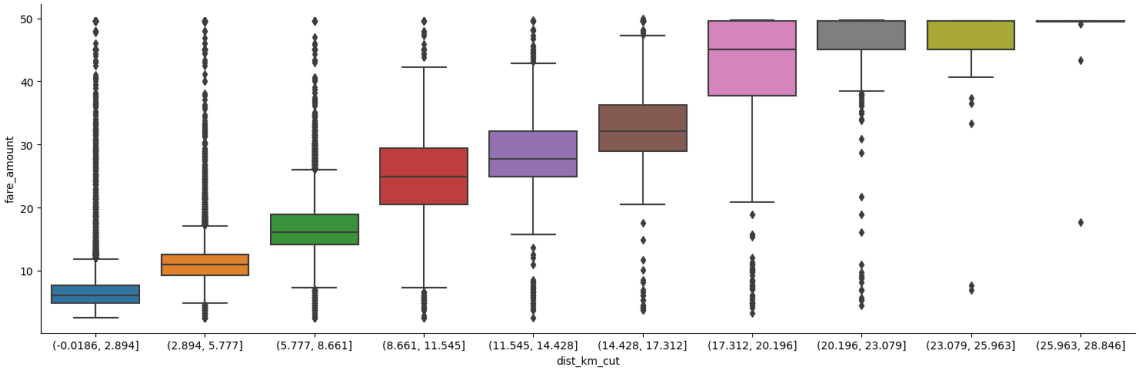

Figura 10: Distribución de Precios por Distancia.

Debido a que el número de features es pequeño, se conservarán todas al momento de modelar. Además en el reporte 1 se pudo constatar que el efecto de eliminar variables no es significativos, ya que los mismos pesos de la red probablemente detereminan cuanta importancia darle.

4. Modelamiento

El proceso de modelamiento contempla el entrenamiento de un MLP y un RBF Neural Net para Regresión.

4.1. Feature Engineering

Comenzaremos el proceso de modelamiento con un proceso de Feature Engineering creando variables adicionales. Como mencionamos en el proceso de Análisis Exploratorio, la distancia es la mejor manera de poder diferenciar precio. Por lo tanto crearemos una feature de distancia en Kilómetros. Para ello utilizaremos la distancia Haversine que considera la distancia en la esfera dependiendo de la latitud

y longitud. La distancia Haversine se describe en la Ecuación (1), donde φ_1 y φ_2 corresponden a las Latitudes de los puntos a medir su distancia, mientras que λ_1 y λ_2 corresponden a la longitud.

$$d_{Haversine} = 2 \cdot arcsin\left(\sqrt{sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + cos\varphi_1 \cdot cos\varphi_2 \cdot sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{1}$$

Por otra parte para poder aprovechar el pickup_date y la importancia de la variable tiempo en el modelo se crearán las siguientes variables:

- Hour: La hora de inicio del viaje. Será un entero de 0 a 23.
- AMorPM: Variable Binaria que indica si es día (AM) o noche (PM).
- Weekday: Indicando el día de la semana que se realiza el viaje. Corresponde a un entero de 0 a 6.

4.2. Preprocesamiento

El procesamiento a realizar para este modelo es estandar y corresponde a un escalamiento con Z-Score según la Ecuación (2).

$$X_{i(escalado)} = \frac{X_i - \mu_i}{\sigma_i} \tag{2}$$

Adicionalmente se hará un preprocesamiento al vector objetivo, aplicando una transformación logarítmica. Esto con el objetivo de explorar si tener una distribución similar a la normal ayuda al proceso de aprendizaje de la red.

4.3. Evaluación

El modelo será evaluado en modalidad Holdout utilizando un 80 % para entrenar y un 20 % para validar. Dado que este problema corresponde a una tarea de predicción, la métrica que se utilizará para evaluar el modelo será el RMSE el cuál se calcula según la Ecuación (3):

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (y_{pred} - y_{real})^2} \tag{3}$$

4.4. Arquitecturas

La RBF Neural Net se compone de 1 capa de entrada, una capa RBF y una de salida. Para realizar una comparación justa se utilizará también un MLP con 1 capa oculta. El proceso de experimentación contemplará la exploración de los siguientes hiperparámetros:

- **m** corresponderá a la cantidad de neuronas en la capa oculta o la capa RBF respectivamente.
- El número k outputs viene definido y corresponde a una neurona que es la tarifa del viaje.
- La última capa no requerirá de función de activación debido a que es un problema de regresión. Se utilizará Mean Squared Error (MSE) como Loss Function.
- Se utilizará ReLU y Tanh como funciones de activación en las capas ocultas y se utilizará Kernel Gaussiano y Multiquadratic para la capa RBF.
- Se probará el efecto de aplicar Logaritmo al vector Objetivo.

- Se probará con SGD y Adam como Optimizadores con un learning rate de 0.001.
- Se probará sólo con un batch size de 1024.
- Se entrena el modelo por 500 Epochs utilizando EarlyStopping con paciencia de 15 Epochs y Checkpointing, es decir, guarda los pesos del mejor puntaje de validación.

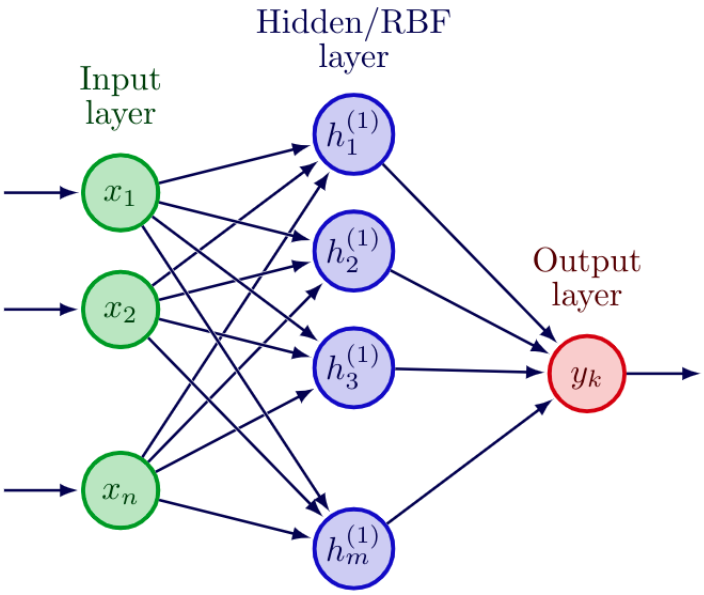

Figura 11: Arquitectura MLP/RBF utilizado.

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 comparando el MLP y el RBF modificando Activación, Kernel y Optimizador. Inicialmente no se vieron grandes cambios ni mejoras al ir variando estos valores. Debido al poco cambio en el resultado del entrenamiento es que se decide aplicar un preprocesamiento al vector objetivo. Este consistió en aplicar una transformación logarítmica. Esto ya que el fare_amount tenía una cola larga hacia la derecha y durante el EDA se encontró que aplicar logaritmo podría dejar una distribución mucho más similar a la Gaussiana.

4.5.2. 2da etapa del Modelo

Entrenar el vector objetivo en escala logarítmica fue una decisión muy acertada ya que los resultados fueron consistentemente mejores que en la escala original.

Finalmente el mejor modelo en el set de validacion se obtuvo utilizando RBF con 256 Neuronas en la capa RBF y Kernel Multicuadrático. El optimizador utilizado fue Adam.

Modelo	m	Activación/Kernel	Optimizador	Log	RMSE
RBF	256	Gaussian	SGD	No	7.4510
MLP	256	ReLU	SGD	No	7.4338
RBF	256	Multiquadratic	SGD	No	7.6482
MLP	256	Tanh	SGD	No	7.4357
RBF	128	Gaussian	SGD	No	7.4491
MLP	128	ReLU	SGD	No	7.4390
RBF	128	Multiquadratic	SGD	No	7.4192
MLP	128	Tanh	SGD	No	7.4207
RBF	256	Gaussian	Adam	No	7.4378
MLP	256	ReLU	Adam	No	7.4305
RBF	256	Multiquadratic	Adam	No	7.3836
MLP	256	Tanh	Adam	No	7.4266
RBF	128	Gaussian	Adam	No	7.4365
MLP	128	ReLU	Adam	No	7.4506
RBF	128	Multiquadratic	Adam	No	7.4644
MLP	128	Tanh	Adam	No	7.4072
RBF	256	Gaussian	SGD	Sí	6.2751
MLP	256	ReLU	SGD	Sí	6.3269
RBF	256	Multiquadratic	SGD	Sí	16.10
MLP	256	Tanh	SGD	Sí	6.3184
RBF	128	Gaussian	SGD	Sí	6.2773
MLP	128	ReLU	SGD	Sí	6.2612
RBF	128	Multiquadratic	SGD	Sí	6.5852
MLP	128	Tanh	SGD	Sí	6.2951
RBF	256	Gaussian	Adam	Sí	6.2665
MLP	256	ReLU	Adam	Sí	6.5553
RBF	**256**	**Multiquadratic**	**Adam**	**Sí**	**5.8353**
MLP	**256**	**Tanh**	**Adam**	**Sí**	**6.1137**
RBF	128	Gaussian	Adam	Sí	6.2848
MLP	128	ReLU	Adam	Sí	6.3684
RBF	128	Multiquadratic	Adam	Sí	6.7562
MLP	128	Tanh	Adam	Sí	6.5011

Tabla 1: Resultados del Entrenamiento: Puntajes medidos en el Set de Validación

5. Conclusiones

Luego de realizar muchos experimentos se puede concluir que la RBF Neural Net vence en términos de Performance al MLP pra este problema de Regresión. En general los puntajes obtenidos no fueron tan buenos hasta que se aplicaron algunos preprocesamientos que permitieron mejoras muy significativas.

En este caos los Hiperparámetros de la Red que más influyeron fueron principalmente la Función de Activación o el Kernel según corresponda, el Número de Neuronas en la Capa oculta o RBF y el Optimizador.

En general se puede concluir que utilizar un número de Neuronas en capas intermedias entregó mejores resultados la mayoría de las veces. Se puede observar que normalmente para el MLP ReLU y Tanh funcionan bien cuando la escala del vector objetivo no es alterada. En el caso de cambiar a escala Logarítmica se ve que Tanh funciona muchísimo mejor. El caso del RBF es un poco más complejo. Tanto Kernel Gaussiano como Multicuadrático suelen funcionar bien. Hay un caso particular en el que el Kernel Multicuadrático da un error excesivamente alto. Se cree que esto se puede haber dado por un claro sobreajuste del modelo que entrega una generalización pésima. Para el caso Logarítmico, Multicuadrático es muy superior y el que entrega los mejores resultados de la experimentación. Finalmente el Optimizador tiene un comportamiento parecido. En el caso del vector objetivo sin alterar no hay mucha diferencia entre SGD o Adam. Una vez que se usa la transformación logarítmica, Adam es claramente superior.

La transformación logarítmica fue muy éxitosa. Al inicio de los experimentos se tuvo muchos problemas para lograr disminuir el error. La transformación logarítmica entregó consistentes errores menores que el caso en escala original. Además encontró el mejor modelo por mucho, bajando de los $6 de error.

Alguno de los aspectos que se pudieron notar es que RBF tendía a activar el Early Stopping mucho después que el MLP. En general RBF requería de 350-400 epochs, mientras que el MLP un poco menos de 200.

Debo decir que no me esperaba que el RBF superara al MLP, en especial porque al inicio de los experimentos no se veían grandes cambios al modificar los Hiperparámetros. El mejor RBF superó con creces al mejor MLP, obteniendo **RMSE de 5.8353 y 6.1137 respectivamente**.

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] D. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional interpolation and adaptive networks," *ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM)*, vol. RSRE-MEMO-4148, 03 1988.

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