Aplicación

Se construyó una aplicación, la cual usa datos de kaggle para el entrenamiento de una red neuronal. La aplicación es sencilla: En base a los datos recolectados, se entrena un modelo, con el que, por medio de unas preguntas al usuario, pueda determinar si éste es feliz en su ciudad.

Redes Neuronales

Se realizaron 3 variaciones en 3 diferentes archivos. La idea era probar 3 diferentes redes neuronales y saber el rendimiento de cada una. Comencemos con la primera, la cual se construyó así:

Red #1

```
modelo = keras.Sequential([
     keras.layers.Input(shape=(6,)),
     keras.layers.Dense(14,activation='linear'),
     keras.layers.Dense(14,activation='relu'),
     keras.layers.Dense(14,activation='tanh'),
     keras.layers.Dense(14,activation='relu'),
     keras.layers.Dense(14,activation='linear'),
     keras.layers.Dense(1,activation='sigmoid')
])
```

Tiene 5 capas ocultas con las funciones de activaciones:

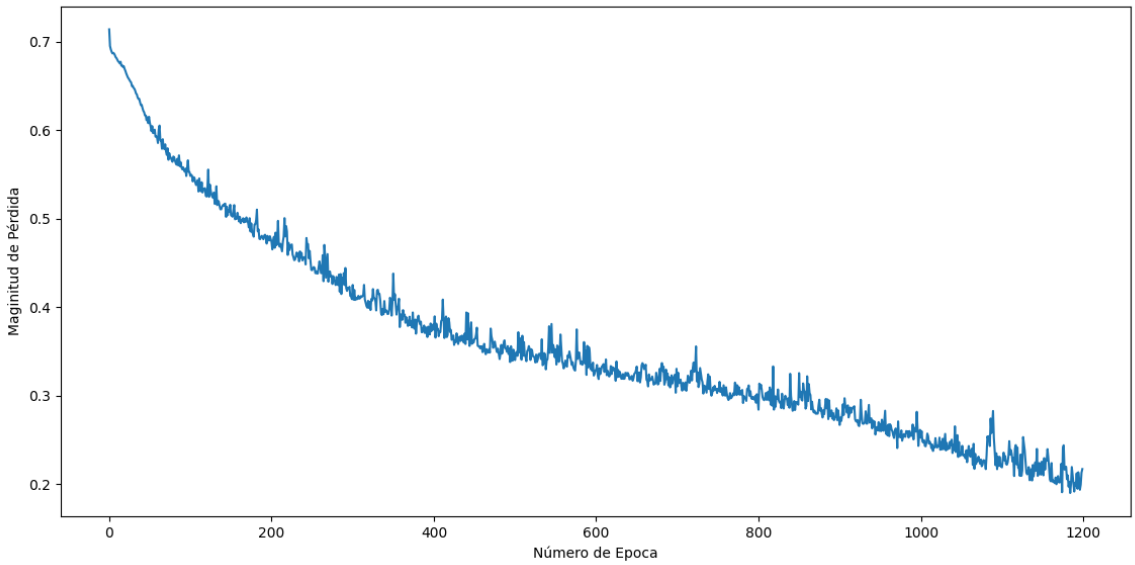
1. Lineal
2. Relu
3. Tanh
4. Relu
5. Lineal

Y la capa de salida se usa sigmoide, ya que el resultado va a ser binario. De hecho, en todas las redes neuronales ésta es la función de activación de las capas de salida. Solo varían las de las capas ocultas.

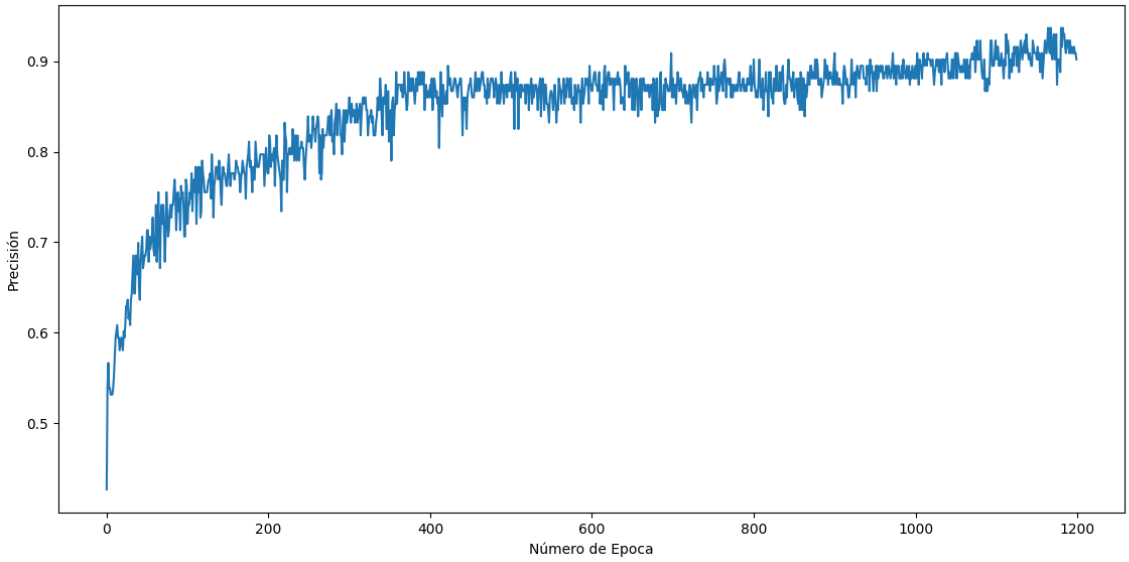
En cuanto al rendimiento de este, tenemos los siguientes datos:

- Sabemos que en el entrenamiento, se recorre por “épocas” y en nuestro caso, tenemos 1200 épocas. Vemos la siguiente gráfica:

En esta gráfica que la magnitud de pérdida o error tiene este comportamiento:



Y por otra parte, en la precisión tenemos:



Estas gráficas no siempre serán exactamente así cada vez que se ejecute el programa, porque obviamente los resultados pueden ser muy variables, pero nos pueden dar una idea de la precisión y error.

En la ejecución mostrada podemos observar que tendríamos un error entre el 20 y 25% cuando llega a las 1200 etapas. Mientras que en la precisión, al cabo de las 1200 épocas, está cerca del 90%, aunque en épocas pasadas llega a ese porcentaje o incluso sobrepasa.

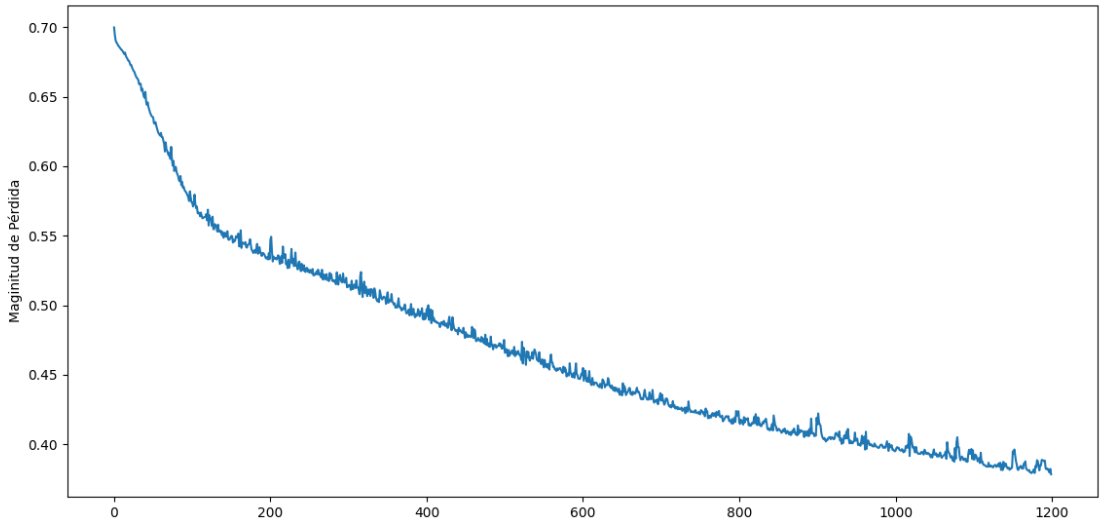
Red #2

El código donde muestra cómo se estructura la red es la siguiente:

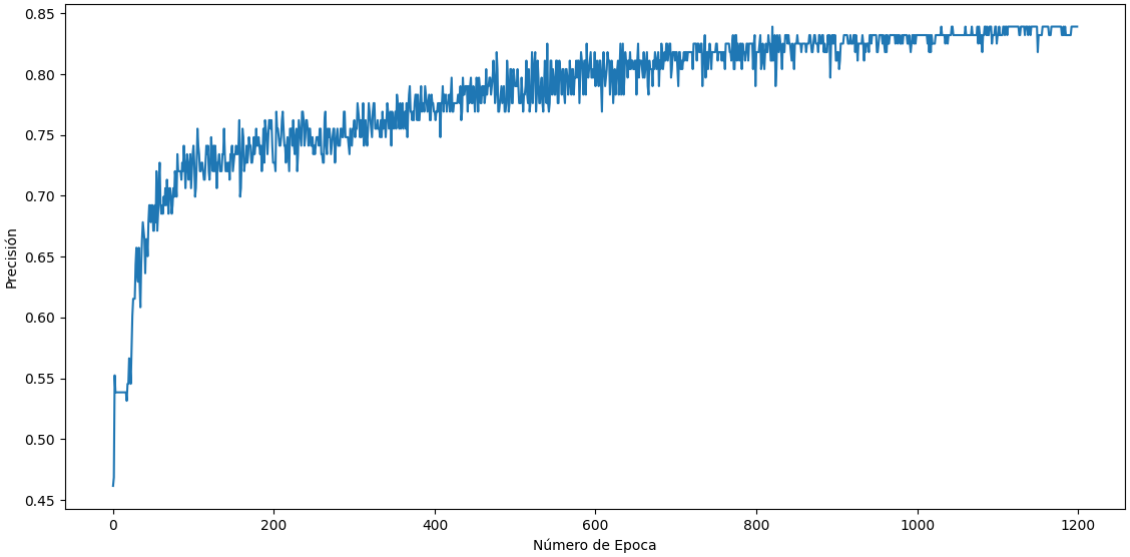
```
modelo = keras.Sequential([
        keras.layers.Input(shape=(6,)),
        keras.layers.Dense(14,activation='linear'),
        keras.layers.Dense(14,activation='relu'),
        keras.layers.Dense(14,activation='linear'),
        keras.layers.Dense(14,activation='softmax'),
        keras.layers.Dense(14,activation='linear'),
        keras.layers.Dense(1,activation='sigmoid')
    ])
```

Tiene 5 capas ocultas con las funciones de activaciones:

1. Lineal
2. Relu
3. Lineal
4. Softmax
5. Lineal



Podemos ver a simple vista que el error no varía tanto y es más estable, con tendencia a la baja, pero vemos que a las 1200 épocas, llega a ser levemente inferior al 40%. A comparación de la red anterior en la misma cantidad llegó a ser menor al 30%.



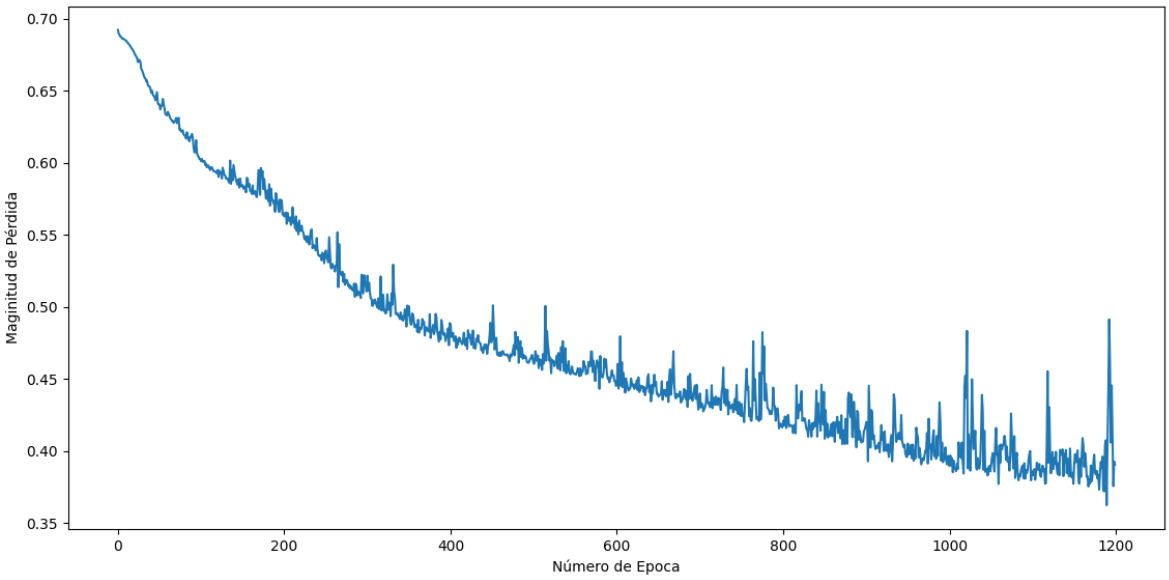
Algo parecido con la anterior, en la red #1 se llegó a una precisión cercana al 90%, mientras que acá es menor al 85%

Red #3

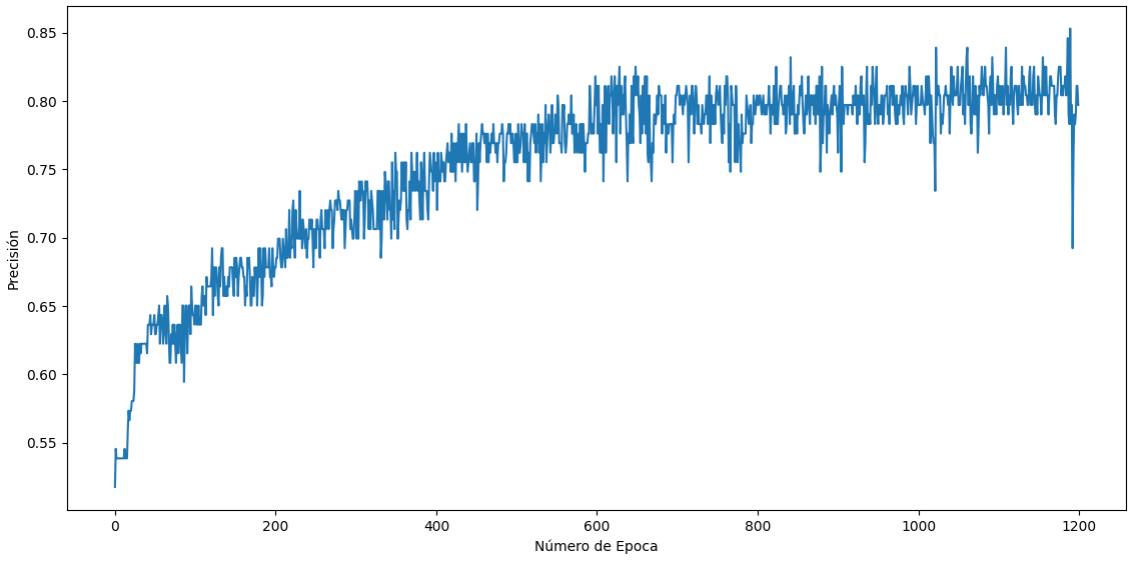
```
modelo = keras.Sequential([
     keras.layers.Input(shape=(6,)),
     keras.layers.Dense(14,activation='softmax'),
     keras.layers.Dense(14,activation='linear'),
     keras.layers.Dense(14,activation='relu'),
     keras.layers.Dense(14,activation='linear'),
     keras.layers.Dense(14,activation='linear'),
     keras.layers.Dense(1,activation='sigmoid')
])
```

Tiene 5 capas ocultas con las funciones de activaciones:

1. Softmax
2. Lineal
3. Relu
4. Lineal
5. Lineal



La magnitud del error al final de las 1200 épocas de esta red es similar al de la red #2, a diferencia de que esta es más variable en el error.



Vemos que la precisión al final de las 1200 epocas puede estar cercano al 80%, incluso un poco menor que la red #2, y aún más variable en su precisión.

Cabe aclarar que estas conclusiones no son totalmente definitivas, pues todo varía y en siguientes ejecuciones de cada una de las redes puede haber resultados diferentes. Sin embargo, podemos hacernos una idea para cada modelo.

En conclusión. La red 2 muestra que quizás sea un poco menos eficiente que las demás en cuanto al error a lo largo de las épocas, sin embargo, vemos que la variabilidad en el error no es demasiada, cosa que sí tienen las otras dos redes en las gráficas de la magnitud de pérdida.

La red #1 puede que tenga mayor eficiencia, pues en las 1200 epocas logró obtener un mejor rendimiento que la #2 y #3, a cambio de un poco de mayor variabilidad en los resultados.

Si queremos mayor estabilidad a cuanto precisión y errores, podríamos escoger la #2, sin embargo, si queremos tener la mejor precisión de entre las 3, podemos escoger a la #1.