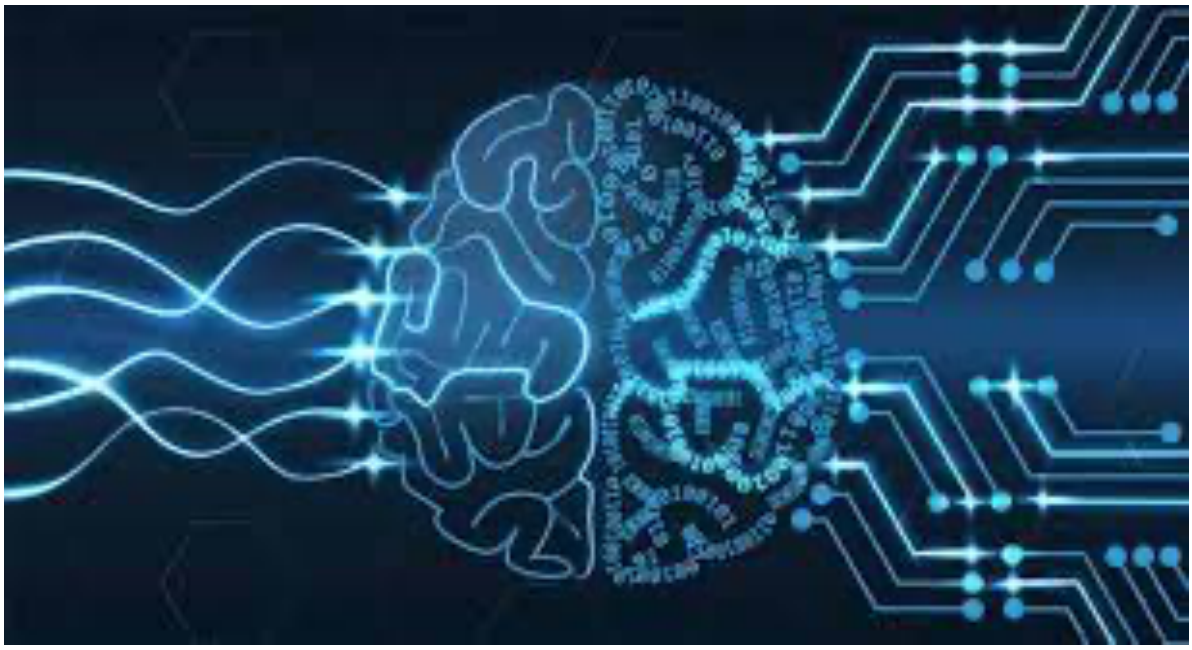
Trabajando con Redes Neuronales y Deep Learning

Técnicas de Inteligencia Artificial



Ibai Ceberio Basañez
30 de Mayo de 2022

Introducción

En esta tarea se nos ha solicitado profundizar en la aplicación de técnicas de aprendizaje supervisado basadas en redes neuronales. Se ha trabajado con dos problemas: regresión y clasificación.

Al mismo tiempo, para cada uno de los problemas se requiere la utilización de una red neuronal y una metodología distinta.

Regresión

EDA

Para la solución de este apartado hemos buscado diferentes dataset que nos permitieran llevar a cabo el proyecto. Finalmente hemos optado por un dataset de Kaggle que nos muestra datos de la calidad de un vino tinto "Vinho Verde" portugués.

El dataset escogido posee 1599 instancias y cuenta con 10 atributos de entrada: fixed acidity, volatile acidity, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates y alcohol. También cuenta con un atributo de salida llamado quality.

En este dataset todos los valores disponibles son numéricos por lo que no tendremos que hacer ninguna codificación en el preprocesamiento. El objetivo con este dataset es comprobar cómo afectan los distintos atributos de entrada a la calidad del vino final.

Gracias al comando `.info()` podemos comprobar que el dataset no posee ningún valor nulo.

En un siguiente paso, se ha procedido a representar los datos de distintas maneras para su exploración. Mediante los histogramas hemos podido ver que hay ciertas variables como la densidad y el pH que presentan una distribución normal. Del mismo modo, vemos que nuestra variable de salida posee 6 valores distintos solamente por lo que también podría considerarse como un problema de clasificación.

Después hemos hecho una matriz de puntos en el que en la diagonal tenemos los histogramas de cada variable de nuevo. Estas matrices nos permiten ver la correlación entre las distintas variables del dataset. En nuestro caso se han dibujado 3 colores principales como malo regular y bueno que se corresponden de la siguiente manera:

```
color=['red' if v <= 5 else 'orange' if v == 5 else 'green' for v in
                        dataset['quality']]
```

Seguidamente, se ha hecho un gráfico de cajas. La finalidad de estos gráficos entre otros. Es la de la identificación de outliers o valores fuera de la media de los datos.

Preprocesamiento de los datos

Una vez observados los datos continuamos con el preprocesamiento de estos. Se ha podido ver que muchos de los datos tenían valores que empeorarán nuestro modelo y consecuentemente han sido limpiados. Si volvemos a dibujar las cajas veremos que donde antes había muchos outliers ahora hay menos y que consecuentemente, los datos restantes

se encuentran en su gran mayoria dentro de la media. Del mismo modo, vemos que hemos limpiado 420 líneas del dataset.

Más adelante, se ha procedido a dividir los datos.

División del dataset en datos de entrenamiento y datos de test

Se ha dividido el dataset en un 80% de entrenamiento y un 20% de test. Para asegurar la convergencia, es una buena práctica normalizar funciones que utilizan diferentes escalas y rangos. Aunque el modelo podría converger sin normalización de características, dificulta el entrenamiento y hace que el modelo resultante dependa de la elección de las unidades utilizadas en la entrada. Esto se puede conseguir utilizando la función `.normalize()`.

Propuesta de arquitectura de red neuronal

En keras, un callback es una función que nos permite parar el entrenamiento de un modelo en cualquier momento. Cuando utilizamos early stopping, podemos elegir una cantidad grande de epochs, más de los que necesitamos.

Podemos añadir capas de dropout para corregir el overfitting. Si dejamos caer una fracción de la capa en cada paso del entrenamiento conseguimos que sea mucho más complicado que el modelo aprenda patrones inadecuados y obligamos al modelo a buscar los patrones más generales. Estos patrones suelen tener pesos mayores.

Evaluación del Modelo

Para evaluar el modelo hemos solicitado al programa que ejecute el mismo proceso con una cantidad distinta de epochs. Gracias a los callback conseguimos pararnos en los parámetros en los que los valores de val y mae son lo más pequeños. Con esto conseguimos que el error sea lo menor posible.

Regresión lineal multivariable

En este caso se ha utilizado un modelo de regresión lineal proveniente del paquete scikit-learn. En este caso se ha utilizado el error absoluto medio como medida de la fiabilidad del modelo. Y curiosamente, el modelo lineal obtiene un valor menor en comparación al modelo de la red neuronal.

Finalmente podemos utilizar la eliminación inversa de variables para ver que variables afectan más al resultado final. Utilizando el paquete statsmodel podemos ir viendo que variables afectan más o menos. Otra de las opciones es utilizar un explicador como Shapley para ilustrar las variables que tienen más o menos peso.

Clasificación

En la parte II de la actividad se nos pide repetir el mismo proceso de exploración pero con un problema de clasificación.

En este caso se ha utilizado otro dataset de Kaggle. Este nos muestra la capacidad de adaptación de estudiantes a los métodos de enseñanza online. El dataset posee 1205 instancias con 13 atributos de entrada y uno de salida. Los atributos de entrada son los siguientes: ['Gender', 'Age Range', 'Education Level', 'Institution Type', 'IT Student', 'Location', 'Load-shedding', 'Financial Condition', 'Internet Type', 'Network Type', 'Class Duration', 'Self Lms', 'Device', 'Adaptivity']. El atributo de salida es la adaptabilidad de los mismos estudiantes.

El dataset con el que hemos trabajado no tiene ningún valor nulo por lo que no debemos preocuparnos por ello.

Por otro lado, cambiaremos el nombre de alguna columna para su mejor entendimiento.

Mediante histogramas mostramos como afecta cada atributo a la respuesta final. Podemos observar por ejemplo como para el rango de edad de 1 a 5 años la adaptabilidad es o baja o moderada como era de esperar.

Como la edad viene por rangos en un string, python automaticamente los ordena por orden alfabético. Para ordenarlos numericamente, se coge el valor minimo de cada rango. A continuación, se convierte esta nueva columna llamada ‘Lower limit Age’ a tipo integer.

Esta vez, en cambio, las variables no son numéricas. La mayoría son objetos o variables cualitativas por lo que hemos de codificarlos. Para ello se ha utilizado un codificador ordinal. Gracias a ello, hemos convertido todas las variables cualitativas en cuantitativas.

Igual que anteriormente, se ha dividido el dataset en un 80% entrenamiento y el 20% restante en test. A pesar de haber codificado todos los valores todavía no están normalizados por lo que en unas pocas líneas de código arreglamos la situación.

Siguiendo adelante se ha utilizado el mismo modelo de red neuronal que se ha utilizado para el primer caso. El error medio absoluto que conseguimos esta vez es igual de pequeño que el anterior por lo que llegamos a la conclusión de que nuestro modelo funciona de manera adecuada.

Se ha obtenido un mae final de 0,05 y un accuracy de 0,72.

Para el caso distinto de la red neuronal se ha optado por utilizar un árbol de decisión. Haciendo ello hemos obtenido finalmente los siguientes resultados: una certeza del 98%.

En este caso también hemos añadido unas líneas de código para la optimización del modelo mediante eliminación inversa. Para nuestros intereses todo aquel P mayor a 0,05 ha sido eliminado.