IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Fecha: Junio 20, 2023

Clasificación en Dataset de Hongos

Roberto de J. Alfaro López[1]
[1]Universidad Autónoma de Querétaro, MX

Correspondiente al autor: Roberto de J. Alfaro López (e-mail: robertoalfaro14@hotmail.com).

RESUMEN El presente documento, presenta la aplicación de técnicas de aprendizaje automático para clasificar hongos según si son comestibles o venenosos. Utilizando un conjunto de datos que contiene información sobre varias características físicas de los hongos, se implementaron y evaluaron dos algoritmos de clasificación: K-Nearest Neighbors (KNN) y una red neuronal. Los modelos se entrenaron y evaluaron utilizando dos métodos de validación: validación *holdout* (el método más simple) y validación por muestreo aleatorio (en inglés *random subsampling*). Los resultados indican que ambos modelos son capaces de clasificar con alta precisión los hongos de dicha base de datos.

PALABRAS CLAVE K-Nearest Neighbors, Red Neuronal, Random Subsampling, Imputación, Codificación por frecuencia.

I. OBJETIVO

Implementar un modelo de clasificación en el "Mushroom Dataset" que distinga, de acuerdo a los atributos físicos, cuáles hongos son comestibles y cuáles son venenosos.

II. INTRODUCCIÓN

El aprendizaje automático ha demostrado ser una herramienta poderosa para tareas de clasificación y predicción en una variedad de campos. En particular, los algoritmos de aprendizaje automático pueden ser útiles para la clasificación de datos de alta dimensión donde las relaciones entre las variables pueden ser complejas y no lineales. En el presente trabajo, se explora la aplicación de dos algoritmos de aprendizaje automático, K-Nearest Neighbors (KNN) y redes neuronales, para llevar a cabo la clasificación de hongos como comestibles o venenosos.

El algoritmo KNN es un método de aprendizaje supervisado que clasifica las instancias basándose en la mayoría de votos de sus vecinos más cercanos. A pesar de su simplicidad, KNN puede ser sorprendentemente efectivo para una amplia gama de problemas de clasificación. Por otro lado, las redes neuronales son modelos computacionales inspirados en el cerebro humano que han demostrado ser especialmente eficaces para tareas de clasificación y regresión. Las redes neuronales son capaces de aprender representaciones de datos de alta dimensión y capturar relaciones complejas entre las variables.

Para examinar estos algoritmos, se utiliza un conjunto de datos que contiene información sobre varias características físicas de los hongos. Igualmente, antes de usar los modelos se llevaron a cabo varias etapas de preprocesamiento de datos. En particular, se implementa la codificación por frecuencia para las variables categóricas y la imputación de moda por clase para los valores faltantes. Estas técnicas de preprocesamiento son esenciales para preparar los datos y pueden tener un impacto significativo en el rendimiento de los algoritmos.

Por último, se analizó la robustez de los modelos usando dos técnicas de validación, validación *holdout* (una simple división en conjunto de entrenamiento y prueba) y validación por *random subsampling*.

III. MARCO TEÓRICO

A continuación, se dará un breve repaso por los algoritmos de clasificación que fueron usados en el trabajo, estos son KNN y las redes neuronales.

A. KNN

KNN es un algoritmo basado en la idea de que los puntos de datos con características similares, tienden a pertenecer a la misma clase. Un aspecto muy interesante de este, es que carece de un método de entrenamiento como tal, en su lugar utiliza un conjunto de datos ya etiquetado para realizar inferencias a partir de este. Los pasos de los que está compuesto el algoritmo son los siguientes:

1. Determinar K: En esta primera etapa se debe elegir el valor de K que mejor se ajuste a la realidad de los datos. Hay diferentes métodos para saber esto, uno de los más usados es el método del codo.
2. Cálculo de las distancias: Para cada punto del conjunto de test que se desee clasificar, se medirá la distancia a todos los demás puntos del dataset de

entrenamiento. Se puede usar cualquier medida de distancia, aunque la más común es la Euclideana.
3. Ordenar los vecinos: Se ordenan todos los vecinos del punto a clasificar según la cercanía hacia este y se eligen los K-vecinos más cercanos.
4. Clasificar: La clase a la que pertenece el punto se determinará según el voto plural (o sea según la clase a la que pertenezcan la mayoría de los vecinos).
5. Evaluación de las métricas.

Como se observa, realmente es un algoritmo fácil de implementar y muy intuitivo.

B. Redes Neuronales

Las redes neuronales artificiales son un modelo computacional inspirado en el cerebro humano, cuyo principal objetivo, es imitar su forma de aprendizaje para poder resolver problemas que resulten demasiado complejos si se usa la programación clásica.

Toda red neuronal está compuesta de pequeños nodos llamados "neuronas", cada una de estas neuronas se conecta con otras intercambiando señales; así, cuando la red es estimulada (o sea que recibe ciertos valores de entrada) la red será capaz de generar una salida. Las conexiones que hay entre cada una de estas neuronas, forman lo que se les llama 'capas', y puede haber redes de una sola o con cientos de ellas; además, si una red tiene tres o más capas, se dice que tiene una capa de entrada, una o varias capas ocultas y una capa de salida.

La forma en que las redes neuronales aprenden es modificando la fuerza que hay entre los enlaces de cada neurona (a estos valores se les llama "pesos") y estableciendo ciertas "reglas" para la salida de cada una de ellas (llamadas función de activación y valor umbral); de esta forma, los distintos pesos que existen entre una misma entrada X1 y diferentes neuronas, hará que el valor cambie para cada una de ellas, del mismo modo, la función de activación y el valor umbral, calcularán una salida adecuada que será el valor de entrada para las neuronas que se encuentren más adelante en la red. Al proceso que consiste en modificar los pesos de las conexiones entre las neuronas, se le llama 'entrenamiento', y hay varias maneras de hacerlo, aunque el método más usado es el algoritmo 'backpropagation' o 'propagación hacia atrás', que permite entrenar fácilmente a redes de grandes dimensiones.

IV. MATERIALES Y MÉTODOS

Los datos para la clasificación se tomaron del "*Mushroom Dataset*", el cual contiene información de los atributos físicos de varios hongos, junto con sus etiquetas para conocer si este es venonoso o comestible. Este dataset posee 23 atributos (contando la clase) y 8123 instancias.

Por otra parte, antes de aplicar los algoritmos de clasificación, se realizaron varias etapas de preprocesamiento de datos para garantizar que los datos estuvieran en un formato adecuado para el análisis.

Primero, se utilizó el método de imputación de "moda por clase" para tratar los valores faltantes en el conjunto de datos. Este método calcula la moda (el valor más común) para cada clase y luego imputa los valores faltantes en la columna con la moda de su respectiva clase. Este enfoque es particularmente adecuado para este conjunto de datos, ya que los atributos de este son categóricos, sin embargo, como se verá más adelante, la moda fue la misma para ambas clases (sólo había un atributo con valores faltantes).

Por último, se implementó el método de codificación por frecuencia. Este método transforma las variables categóricas en numéricas al reemplazar cada valor en la columna por su frecuencia relativa en el conjunto de datos. Dado que todas las características en el "Mushroom Dataset" son categóricas, este método de codificación era esencial para los algoritmos que se usarían, pues estos requieren entradas numéricas (KNN y redes neuronales). Además, la codificación por frecuencia puede ayudar a preservar la información sobre la distribución de los datos, que puede ser valiosa para la clasificación.

V. PSEUDOCÓDIGO

ALGORITMO 1: KNN

```
class KNN:
    K = (k1, k2, …, kn)  # num vecinos
    C = (c1, c2, …, ck)  # clases
    D = (d1, d2, …, dn)  # dataframe
    T = (t1, t2, …, tn)    # dataframe de test
    i = num iteraciones

    c1, c2, …, ck ∈ D

  for i = 1 hasta i = I:
      foreach d ∈ D:
          argminDistance(d, t)
          d -> k
       end
      foreach k ∈ K:
          c = majority(d ∈ D)
      end
    end
  end
 return C
```

ALGORITMO 2: RANDOM SUBSAMPLING

```
class RandomSubsampling:
    D = (d1, d2, ..., dn)
    I = num iteraciones
    T = tamaño del conjunto de prueba

    for i = 1 hasta i = I:
        TestIndices = randomSample(D, T)
        TrainIndices = randomSample(D, len(D) - T)

        TrainSet = D[TrainIndices]
        TestSet = D[TestIndices]

        classifier.train(TrainSet)

        Predictions = classifier.predict(TestSet)
        Metrics = calculateMetrics(TestSet, Predictions)
    end
```

```
    AverageMetrics = calculateAverage(Metrics)

    return AverageMetrics
end
```

VI. RESULTADOS

A. Parámetros e Hiperparámetros

Como se comento anteriormente, se usaron dos modelos de clasificación, el primero basado en KNN y el segundo una red neuronal.

Es bien sabido que KNN casi no tiene hiperparámetros a configurar, por lo que basta decir que se usó un valor de K igual a 5.

Por su parte la red neuronal tenía la siguiente estructura:

- 22 entradas (una por atributo) conectadas a 150 neuronas.
- Función de activación ReLU y conexión a otra capa de 100 neuronas.
- Una última función de activación ReLU y una salida de 2 neuronas (una por cada clase).

Por último, se entrenó con un *batch size* de 32, optimizador Adam y 100 épocas.

B. Validación Holdout

Primero se mostrarán los resultados de cada modelo usando la técnica de validación más básica "*holdout*"; esta consiste en dividir los datos en entrenamiento y prueba (en este ejemplo 80-20 respectivamente) entrenar el modelo y evaluarlo en el conjunto de test aplicando las métricas que se requieran, esto se hace una sola vez.

Las métricas de clasificación obtenidas por el modelo KNN fueron las siguientes:

TABLE I
MÉTRICAS KNN VALIDACIÓN HOLDOUT

Métrica	Valor
Accuracy	1.0
F1 score	1.0
Recall	1.0

Por su parte, las métricas obtenidas por la red neuronal fueron:

TABLE II
MÉTRICAS RED NEURONAL VALIDACIÓN HOLDOUT

Métrica	Valor
Accuracy	1.0
F1 score	1.0
Recall	1.0

Se puede observar que aparentemente la clasificación es perfecta usando ambos métodos.

C. Validación Random Subsampling

Por último, se muestran las métricas al usar validación por "*random subsampling*". Esta técnica de validación, consiste en dividir el conjunto de datos igualmente en entrenamiento y prueba, pero en este caso, el proceso de evaluación se repite N cantidad de veces usando en cada iteración diferentes divisiones de los datos. A diferencia de otros métodos de validación iterativos (como la validación cruzada) aquí los datos pueden estar repetidos en el conjunto de datos de entrenamiento y prueba.

La cantidad de iteraciones usadas para la validación en ambos modelos fue de 50.

Las métricas de clasificación obtenidas por el modelo KNN usando fueron las siguientes:

TABLE I
MÉTRICAS KNN VALIDACIÓN RANDOM SUBSAMPLING

Métrica	Valor
Accuracy	0.999963054187192
F1 score	0.9999629665524574
Recall	0.9999612403100775

Los resultados en cada iteración y para cada métrica, se observan a continuación:

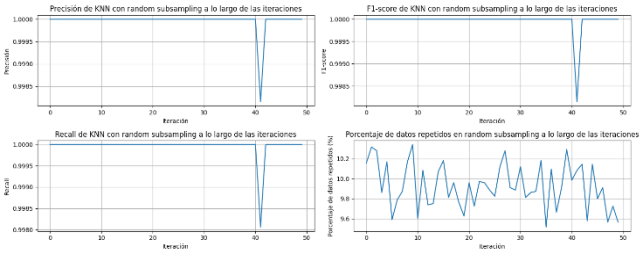

FIGURA 1. Cambio de las métricas en cada iteración para KNN

Por su parte, las métricas obtenidas por la red neuronal fueron:

TABLE II
MÉTRICAS RED NEURONAL VALIDACIÓN RANDOM SUBSAMPLING

Métrica	Valor
Accuracy	0.9999137931034483
F1 score	0.9999136226408559
Recall	0.9999103418390848

Los cambios en cada iteración también se muestran a continuación, se puede observar que a pesar de tener métricas muy parecidas con KNN, las caídas en estas son más frecuentes usando la red neuronal, aunque por la escala se pueden considerar con poca importancia.

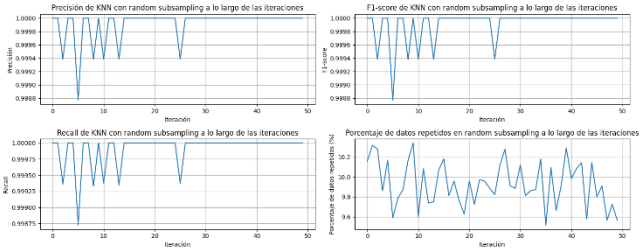

FIGURA 2. Cambio de las métricas en cada iteración para la red neuronal

Al igual que en la sección anterior, las métricas son realmente parecidas.

VII. CONCLUSIONES

Gracias a los resultados que se han presentado, se puede observar que, al menos para esta tarea, el algoritmo KNN es igual de bueno realizando clasificación como las redes neuronales, las cuales son más complejas tanto de implementar como de entender y entrenar. Este trabajo es el claro ejemplo de que lo más complejo no siempre es lo mejor, ni lo que se debe utilizar. No sólo porque se llega al mismo resultado (incluso muy ligeramente mejores), sino que también ahorra mucho tiempo, ya que al aplicar validación por random subsampling, el algoritmo KNN se demoraba unos minutos, mientras que las redes neuronales tardaban una cantidad significativa de tiempo (casi 50 minutos).

Por último, caben destacar los métodos de validación usados, al menos para esta tarea las cosas se mantuvieron muy parecidas, pero se observa que efectivamente, nuestro accuracy y demás métricas no son de 1.0 como lo mostraba la validación simple, sino que puede variar por más perfecto que parezca. Con random subsampling se pudo observar que efectivamente los modelos pueden equivocarse y que es importante usar algún método de validación iterativo de tal manera que se pueda saber de mejor manera qué esperar de ellos.

BIBLIOGRAFÍA

[1] M. A. Aceves Fernandez, Inteligencia Artificial para Programadores con Prisa. Independently Published, 2021.

[2] F. Berzal, Redes Neuronales & Deep Learning, vol. 1, 2019.