Fecha: Junio 02, 2023

Clasificación con KNN

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 En el presente reporte, se muestra el desarrollo de un modelo de clasificación. El algoritmo utilizado es KNN (k-nearest neighbors) y el objetivo es que, basado en datos de estudiantes universitarios otorgados por el dataset "Student Succes", se pueda predecir si un estudiante continuará la carrera o no. Además, también se muestran los métodos de normalización e imputación que se creyeron más adecuados para el preprocesamiento de la base datos. Finalmente se analizan los resultados del modelo, los cuales parecen indicar que KNN no es el mejor algoritmo de clasificación para esta base de datos en concreto.

PALABRAS CLAVE Normalización, Imputación, KNN, Accuracy, Recall.

I. OBJETIVO

Crear el código necesario para implementar el algoritmo KNN y utilizarlo como método de clasificación para la base de datos "Student Succes".

II. INTRODUCCIÓN

Hasta el momento en prácticas pasadas, se ha dado un repaso por las diversas técnicas de preprocessamiento de bases de datos y también se ha visto la aplicación de algoritmos de clustering. A continuación, en el presente reporte se aplicará otro método muy usado, que es el algoritmo de KNN (k-nearest neighbors).

Se sabe de antemano que, en el aprendizaje automático, existen un gran número de algoritmos que permiten realizar la tarea de clasificación, sin embargo, pocos de ellos son tan sencillos de implementar como KNN. Perteneciente al grupo de los llamados '*lazy algorithms'*, KNN es una herramienta que puede usarse para realizar clasificación y regresión y además no posee estrictamente hablando una etapa de entrenamiento (de ahí que pertenezca a tal grupo).

Este algoritmo, se basa en la idea intuitiva de que los puntos de datos con características similares, tienden a pertenecer a la misma clase. Su funcionamiento es relativamente simple, pero efectivo, de forma muy general, dado un nuevo punto de datos a clasificar, el algoritmo busca los K puntos más cercanos en el conjunto de entrenamiento y asigna al nuevo punto la clase más común entre sus vecinos. Esto se logra mediante el cálculo de distancias, generalmente utilizando la distancia euclidiana, aunque también es posible utilizar otras métricas.

La simplicidad y eficiencia del algoritmo KNN han llevado a su aplicación en una amplia gama de problemas de clasificación en diversas áreas. Ha sido utilizado con éxito en reconocimiento de patrones, minería de datos, análisis de mercado y más.

A continuación, se documenta todo el proceso y las consideraciones técnicas que se tomaron en cuenta para llevar a cabo esta práctica.

III. MARCO TEÓRICO

Como se mencionó en la sección anterior, KNN es un algoritmo relativamente simple, el cual usa criterios de distancia para clasificar datos. A continuación, se enlistan algunas características de este método, así como los pasos por los que está compuesto y algunas ventajas y desventajas de este.

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, sin embargo, como todo, tiene sus ventajas y desventajas.

Ventajas:

- Fácil de implementar.
- Adaptibilidad a cambios en los datos de entrenamiento.
- Poca cantidad de hiperparámetros a ajustar.

Desventajas:

- Sensibilidad a la alta dimensionalidad.
- Tendencia al sobreajuste.
- Exige una gran cantidad de recursos computacionales.

IV. MATERIALES Y MÉTODOS

En esta sección se repasará brevemente el preprocesamiento de los datos que se llevo a cabo, antes de la clasificación con KNN.

El dataset que se utilizó tiene por nombre “Student Succes”, posee 34 atributos y 1 más con las etiquetas. Muchos de los datos son categóricos, pero codificados (cada número significa algo según el atributo). El objetivo principal del dataset es lograr predecir el estado de un estudiante según datos específicos como el grado de estudio previo, escolaridad de los padres, PIB en ese momento, estado civil, entre otros.

A. Normalización e Imputación

Un paso inspensable antes de la creación de cualquier modelo de clasificación (además de revisar datos faltantes y conocer el significado de los atributos), es revisar el balance de las clases, pues si estas se encuentran en valores muy distintos, puede ser que el modelo no se comporte de la mejor manera. En este caso se puede observar en la siguiente gráfica, que las clases se encuentran muy desbalanceadas:

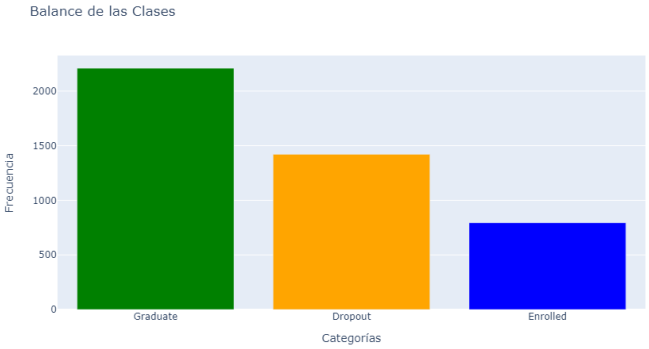

FIGURA 1. Balance de clases original del dataset.

Debido a esto se llevó a cabo la imputación por el método de la ruleta, revisando que las distribuciones permanecieran lo más parecidas posibles.

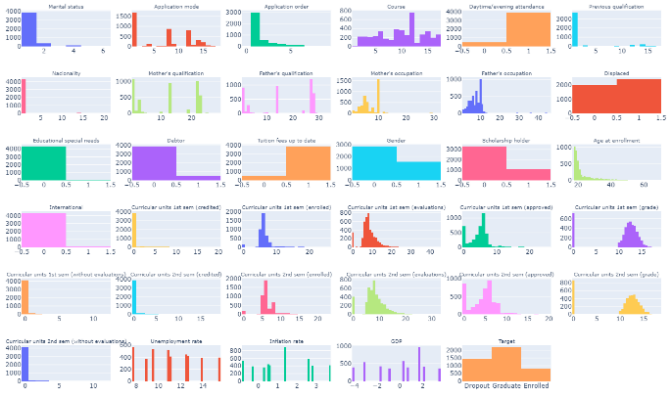

FIGURA 2. Distribuciones originales del dataset.

FIGURA 3. Distribuciones del dataset posterior a la imputación.

Se puede observar que la imputación se hizo correctamente.

Por último, en esta sección, también se llevó a cabo la normalización de los datos entre 0 y 1, y utilizando el método min-max. El resultado fueron casi las mismas distribuciones que las antes vistas:

FIGURA 4. Distribuciones después de normalizar.

B. Reducción de Dimensionalidad

El algoritmo KNN resulta sencillo de implementar y en muchas ocasiones sus resultados son muy buenos. Sin embargo, entre sus desventajas se encuentra que ante datos con altas dimensionalidades, puede que sus resultados no sean los esperados. Es por esto que se decidió reducir la dimensionalidad del dataset, de 34 atributos a sólo 3, más el atributo de etiquetas. El algoritmo de reducción que se utilizó fue PCA.

V. PSEUDOCÓDIGO

Para la implementación de KNN se creó una clase en Python, la cual corresponde al siguiente 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
```

VI. RESULTADOS

Para elegir el mejor número K, se comparó el accuracy obtenido con distintos valores desde k=1 hasta k=25, los resultados se muestran en la siguiente gráfica:

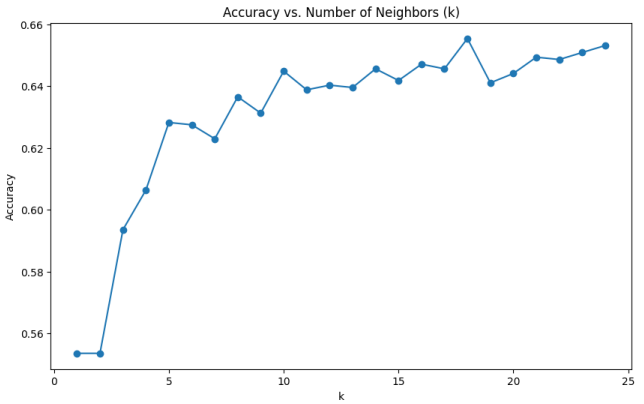

FIGURA 5. Elección del mejor valor de K.

Se puede observar que el mejor accuracy se alcanzó en K=18, por lo que ese será el número de vecinos más cercanos a considerar.

Para evaluar el modelo, primero se dividió el conjunto de datos en 80% entrenamiento y 20% test. Los resultados del test se evaluaron usando las métricas accuracy, recall y F1-score, cuyos valores se muestran en la tabla 1.

MÉTRICAS DE EVALUACIÓN	
Accuracy	**0.6553**
Recall	**0.6553**
F1-score	**0.6562**

Tabla1. Resultados del algoritmo KNN para clasificación.

VII. CONCLUSIONES

Como se puede observar, los resultados anteriores no fueron precisamente los mejores, pues realmente el modelo cuenta con métricas bastante bajas, sólo un poco mejor que un tiro de moneda. Es cierto que podrían implementarse algunos métodos que probablemente llevaran a mejor término los resultados (por ejemplo proponer un criterio de elección en caso de que la cantidad de vecinos sea casi la misma). Sin embargo, se considera que realmente los datos tienen una gran cantidad de dimensiones en este dataset y que al aplicar PCA de 34 a 3, puede estar perdiéndose alguna información importante. Además, los datos en general están poco relacionados, por lo que quizá algún algoritmo como redes neuronales, podrían ser más adecuados para este problema.

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.