Universidad Nacional de San Agustín

Escuela Profesional de Ciencia de la Computación

Maestría en Ciencias de la Computación

Algoritmos y Estructura de Datos

Análisis de viabilidad crediticia mediante KD-Tree y KNN

Un enfoque de Aprendizaje Automático para evaluar el riesgo de pago de nuevos solicitantes.

Docente: Ph.D.(c) Vicente Machaca Arceda

Participantes:

Abel E. Borit Guitton
Luis A. Borit Guitton
Betzy J. Yarín Ramírez

2023

Universidad Nacional de San Agustín
Maestría en Ciencias de la Computación

Trabajo de Investigación

Abel E. Borit Guitton, Luis A. Borit Guitton, Betzy J. Yarín Ramírez

31 de agosto de 2023

Índice

1. INTRODUCCIÓN

En la actualidad, el análisis de datos y la toma de decisiones basadas en información precisa se han convertido en pilares esenciales para una variedad de campos, y la industria financiera no es una excepción. La concesión de créditos despliega un papel crucial en la economía global, pero conlleva consigo riesgos inherentes para instituciones financieras y prestamistas.

La evaluación precisa y efectiva de la solvencia crediticia de los individuos se vuelve esencial para minimizar riesgos financieros, y en este contexto, las técnicas avanzadas de análisis de datos juegan un papel crucial. En este panorama, la importancia de una evaluación crediticia sólida cobra un nuevo significado. El proceso de determinar si un solicitante es elegible para recibir un crédito debe ser respaldado por enfoques que permitan una visión integral y precisa. Aquí es donde la integración de las técnicas de Árboles KD (K-Dimensional) y el algoritmo de vecinos más cercanos (KNN) como herramientas valiosas para el análisis crediticio. Estos métodos no solo permiten una evaluación enriquecedora y precisa de la solvencia crediticia, sino que también tienen el potencial de revelar patrones y relaciones ocultas en los datos que podrían escapar a una observación superficial.

Los Árboles KD y el algoritmo KNN se alzan como herramientas cruciales por varias razones:

1. **Consideración Multidimensional:** La evaluación crediticia implica múltiples dimensiones de datos, como la edad de un individuo, crédito otorgado y si fue un buen pagador o no. Los Árboles KD permiten una segmentación multidimensional, dividiendo el espacio de características en subespacios coherentes, lo que resulta en búsquedas más eficientes y una evaluación más precisa.

2. **Identificación de Perfiles Similares:** El algoritmo KNN, al localizar vecinos cercanos en función de atributos similares, ofrece la posibilidad de identificar perfiles crediticios similares. Esto permite una evaluación más precisa al comparar la solvencia de un individuo con la de otros que comparten características similares.

3. **Detalles Ocultos y Patrones:** La combinación de Árboles KD y KNN puede revelar patrones ocultos y relaciones en los datos. Esto es fundamental en la evaluación crediticia, ya que permite descubrir tendencias en el historial crediticio que podrían no ser evidentes a simple vista.

4. **Eficiencia y Rendimiento:** Los Árboles KD y KNN trabajan en conjunto para optimizar la búsqueda de vecinos cercanos en espacios multidimensionales. Esto resulta en una evaluación crediticia más eficiente, reduciendo el tiempo y los recursos requeridos para tomar decisiones informadas.

Sin embargo, la interpretación precisa de los datos crediticios no se limita solo al uso de estas técnicas. La normalización y escalado de los atributos juegan un papel vital en el proceso.
El escalador MaxMin emerge como un aliado esencial para tratar la diversidad en la escala de los atributos. A través de la normalización, se garantiza que las diferentes características contribuyan de manera equitativa a la evaluación final, evitando distorsiones causadas por variaciones de escala.

En resumen, el análisis crediticio se beneficia enormemente de la fusión de enfoques avanzados. La combinación de Árboles KD y el algoritmo KNN brinda una evaluación multidimensional precisa y una toma de decisiones informada. A su vez, el escalador MaxMin asegura la comparabilidad y equidad en la evaluación, permitiendo que instituciones financieras y prestamistas aborden el otorgamiento de créditos con mayor confianza y eficacia en un entorno financiero cada vez más complejo y dinámico.

2. TRABAJO DE INVESTIGACIÓN

La estructura KD-Tree es una estructura multidimensional de k dimensiones. Esta permite implementar búsquedas por similitud como K Nearest Neighbor o Closest point. Adicionalmente, se usará esta estructura como un clasicador. A continuación detallamos el algoritmo:

Algorithm 1: KNN Classifier

Input: X: training data; y: object to be classified.
Output: Classification for y.
Extract features of each sample;
Build KD-Tree;
Select KNN of y in X;
Class(y) $\leftarrow$ max of classes (k closest objects);

Figura 1: Algoritmo KNN

Se usará un descriptor para tomar como entrada una muestra de la base de datos y retornar un vector de características, luego este vector representará un punto en el KD-Tree.

3. ALGORITMOS

3.1. KD-Tree

Estructura de datos muy utilizada en el campo de la informática para organizar y buscar datos en espacios de varias dimensiones, como coordenadas (x, y) o características múltiples. El K-DTree es especialmente útil para buscar puntos cercanos y para realizar búsquedas en rangos en espacios de alta dimensionalidad. [2]

1. Características:
   a) Eficiencia en la búsqueda, al seguir el camino descendente del árbol, se puede descartar rápidamente regiones que no contienen los puntos de interés.
   b) Eficiencia en alta dimensión, en comparación con enfoques de fuerza bruta, el Árbol KD puede ofrecer una mejora significativa en la eficiencia de búsqueda en espacios de alta dimensión.
   c) Exploración equilibrada, debido a su naturaleza alternante de selección de dimensiones, el Árbol KD asegura una exploración equilibrada de las diferentes dimensiones, evitando el sesgo en ciertas direcciones.
2. Funcionalidad:
   a) **Estructura de datos jerárquica:** Cada nodo representa un hiperplano de corte en una dimensión específica del espacio multidimensional.
   b) **División Recursiva:** El espacio de datos se divide recursivamente en subespacios por medio de hiperplanos ortogonales a los ejes de coordenadas, creando una estructura en forma de árbol binario.
   c) **Selección de Dimensión:** En cada nivel del árbol, se selecciona una dimensión alternante para realizar la partición.
   d) **Puntos de Bifurcación:** Divide el espacio en dos subespacios. El punto de bifurcación es elegido de manera que los puntos se distribuyan en torno a la mediana en la dimensión seleccionada.

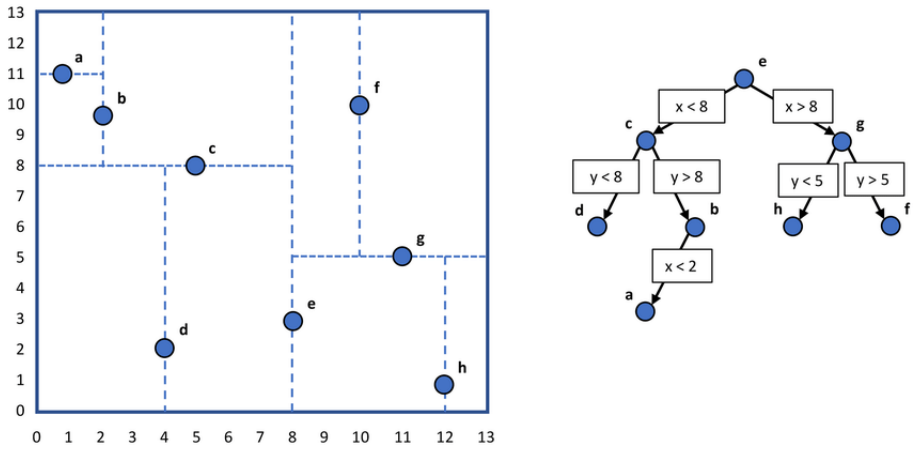

Figura 2: Visualización de Trabajo del Árbol KD

3. Inconvenientes:

   a) Deterioro en espacios de alta dimensión debido al fenómeno conocido como "maldición de la dimensionalidad". En espacios de muchas dimensiones, la mayoría de los puntos pueden estar a la misma distancia de un punto de consulta, lo que reduce la eficacia de la partición del árbol.

   b) Construcción costosa, especialmente para conjuntos de datos grandes, a medida que los datos cambian puede requerir operaciones complicadas.

   c) Distribución irregular, puede conducir a una partición desigual y subóptima del espacio, cuando los datos están agrupados de manera irregular.

3.2. ALGORITMO K Vecinos más cercanos (K Nearest Neighbor, KNN)

Es un algoritmo simple y versátil de aprendizaje supervisado, utilizado para tareas de clasificación y regresión. Pertenece al campo del aprendizaje automático y minería de datos. [1]

1. **En clasificación,** el punto de datos dado se clasifica según la mayoría del tipo de sus vecinos. El punto de datos se asigna a la clase más frecuente entre sus k vecinos más cercanos

2. Mientras que en **regresión**, se calcula un valor promedio basado en los valores de propiedad de los vecinos más cercanos.

Tanto en clasificación como regresión, la entrada son los K ejemplos de entrenamiento más cercanos en el espacio de características; se puede asignar un peso a las contribuciones de los vecinos, de esta manera los vecinos más cercanos contribuyen más al promedio que los más distantes.

1. Características:

   a) No paramétrico (no asumen nada sobre cómo está la distribución de los datos).

   b) Se basa en la información de los puntos de datos cercanos en el espacio de características.

   c) No requiere una etapa de entrenamiento como otros algoritmos de aprendizaje supervisado para generar el modelo, utiliza todos los datos en la fase de prueba.

   d) El entrenamiento es rápido.

2. Funcionalidad:

   a) **Datos de entrenamiento:** Conjunto de datos que contiene ejemplos con características y etiquetas.

   b) **Nuevo punto:** Cuando se presenta un nuevo punto, el algoritmo busca los puntos más cercanos en función de una medida de distancia.

   c) **Medida de distancia:** Se calcula la distancia entre el nuevo punto y todos los puntos en el conjunto de entrenamiento.

d) **Selección de vecinos:** Los K puntos con las distancias más pequeñas al nuevo punto son seleccionados como "vecinos más cercanos".

e) **Clasificación o regresión:** En el caso de clasificación, se cuenta cuántos vecinos pertenecen a cada clase y se asigna la clase más común al nuevo punto. En regresión, se promedian los valores de los vecinos para predecir el valor del nuevo punto.

f) **Resultado final:** La clase asignada o el valor predicho se aplica al nuevo punto como su resultado final.

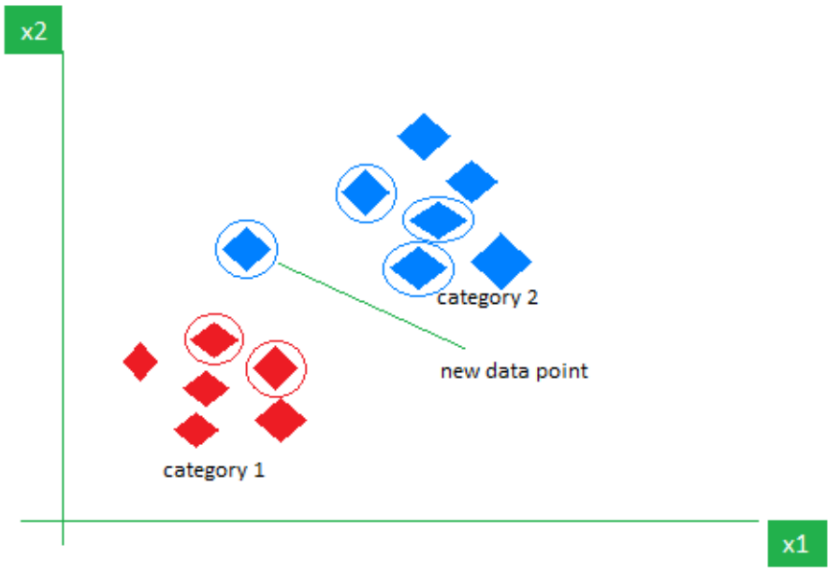

Figura 3: Visualización de Trabajo del Algoritmo KNN

3. Inconvenientes:

   a) Fuerte dependencia de los datos.

   b) Exactitud, es muy sensible al ruido y a características irrelevantes.

   c) Lentitud, puede llevar tiempo obtener respuestas cuando se tiene conjuntos de datos grandes. En la fase de prueba requiere más tiempo y recursos de memoria.

Su simplicidad y eficacia en la resolución de varios problemas, lo convierten en una herramienta esencial en el conjunto de técnicas de análisis de datos. [3]

3.3. Escalador MaxMin

Es una técnica de normalización, que ajusta los valores de las características a un rango específico. Cuando se aplica al contexto de KNN (vecinos más cercanos) y KD-Tree (Árbol KD), esta técnica puede mejorar la eficacia y la equidad de la evaluación en problemas de búsqueda y análisis en espacios multidimensionales.

4. IMPLEMENTACIÓN

En el siguiente enlace Repositorio GitHub se podrá visualizar toda la implementación realizada y también un archivo README.md propio del proyecto y repositorio.

En el siguiente enlace Video de YouTube se podrá visualizar el video subido en YouTube para una mejor explicación del proyecto.

Para la implementación del KNN y KD-Tree, importamos las bibliotecas necesarias 1.

Algorithm 1 Importación de Bibliotecas

```
importar PANDAS como PD
importar NUMPY como NP
importar MATPLOTLIB.PYPLOT como PLT
```

En el Código 2, podemos observar la implementación del KD-Tree.

Algorithm 2 KD-Tree

```
function ConstruirKDTree(puntos, profundidad)
    if longitud(puntos) = 0 then
        return Nulo
    end if
    eje ← profundidad mód longitud(puntos[0])
    Ordenar puntos por el valor en el eje-ésimo eje
    mediana ← longitud(puntos) dividido 2
    nodo ← nuevo KDNode(puntos[mediana], eje)
    nodo.izquierda ← ConstruirKDTree(puntos[0 : mediana], profundidad + 1)
    nodo.derecha ← ConstruirKDTree(puntos[mediana + 1 : longitud(puntos)],
profundidad + 1)
    return nodo
end function
```

En el Código 3, podemos observar la implementación del KNN.

Algorithm 3 K-Nearest Neighbors

function BUSCARKNN($nodo, punto, k$)
  function BÚSQUEDARECURSIVA($nodo, punto, k, cercanos$)
    if $nodo$ = Nulo **then**
      return
    end if
    $distancia \leftarrow$ norma(punto − nodo.punto)
    if longitud($cercanos$) $< k$ **then**
      Agregar (distancia, nodo.punto) a $cercanos$
      Ordenar $cercanos$ por la distancia
    else if distancia < cercanos[−1][0] **then**
      Eliminar el último elemento de $cercanos$
      Agregar (distancia, nodo.punto) a $cercanos$
      Ordenar $cercanos$ por la distancia
    end if
    $distancia\_eje \leftarrow$ punto[nodo.eje] − nodo.punto[nodo.eje]
    $nodo\_cerca, nodo\_lejano \leftarrow$ elegir entre (nodo.izquierda, nodo.derecha) y (nodo.derecha, nodo.izquierda) según $distancia\_eje \leq 0$
    BÚSQUEDARECURSIVA($nodo\_cerca, punto, k, cercanos$)
    if |distancia_eje| < cercanos[−1][0] **then**
      BÚSQUEDARECURSIVA($nodo\_lejano, punto, k, cercanos$)
    end if
  end function
  $puntos\_cercanos \leftarrow$ []
  BÚSQUEDARECURSIVA($nodo, punto, k, puntos\_cercanos$)
  return [punto para distancia, punto en puntos_cercanos]
end function

5. RESULTADOS

5.1. Distribución de pagadores y deudores

Según la edad y monto del crédito

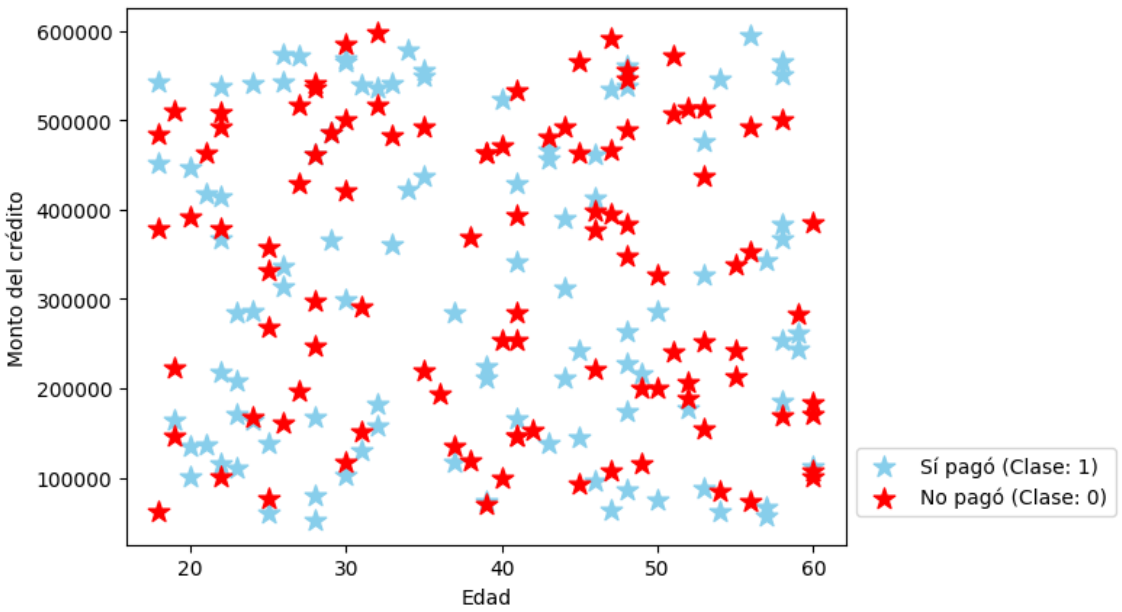

Figura 4: Gráfica de Dispersión de clientes pagadores vs deudores

5.2. Inclusión de un Nuevo Solicitante +

En el análisis de pagadores y deudores

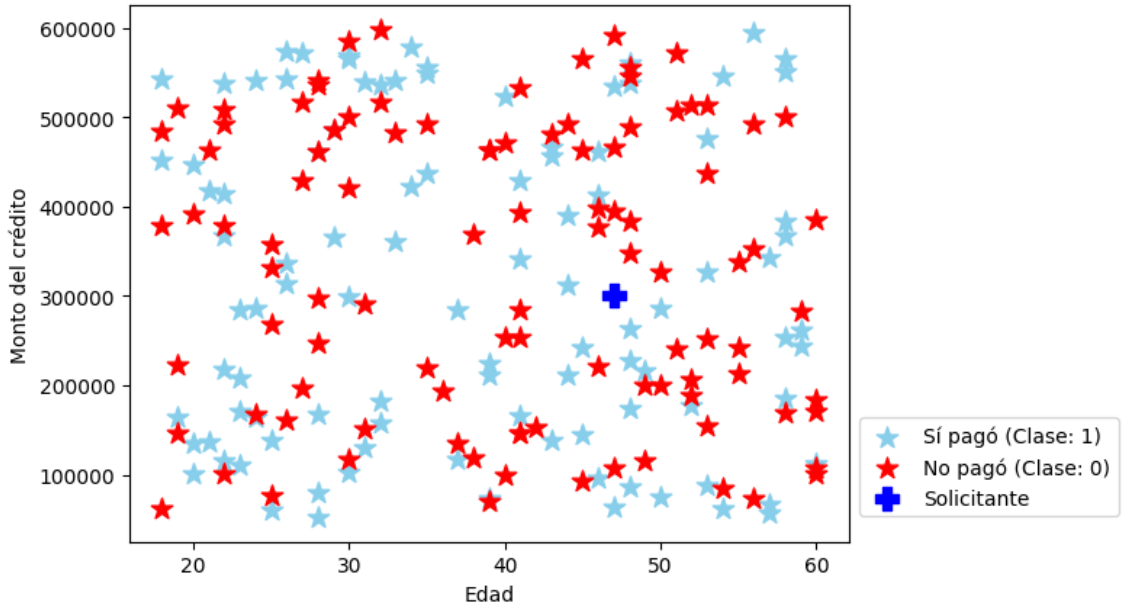

Figura 5: Gráfica de Dispersión con el Nuevo Solicitante

5.3. Mapa de Calor para Predicción de Pago de Créditos

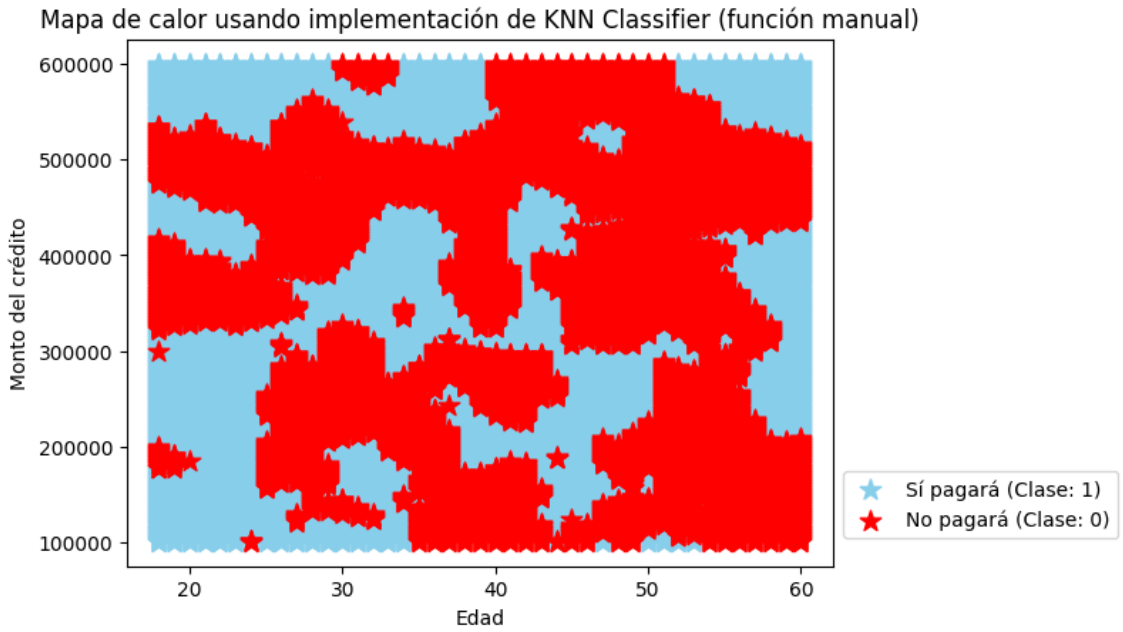

Figura 6: Distribución de Predicciones de Pago de Créditos

5.4. Funcionamiento del Modelo en términos de métricas

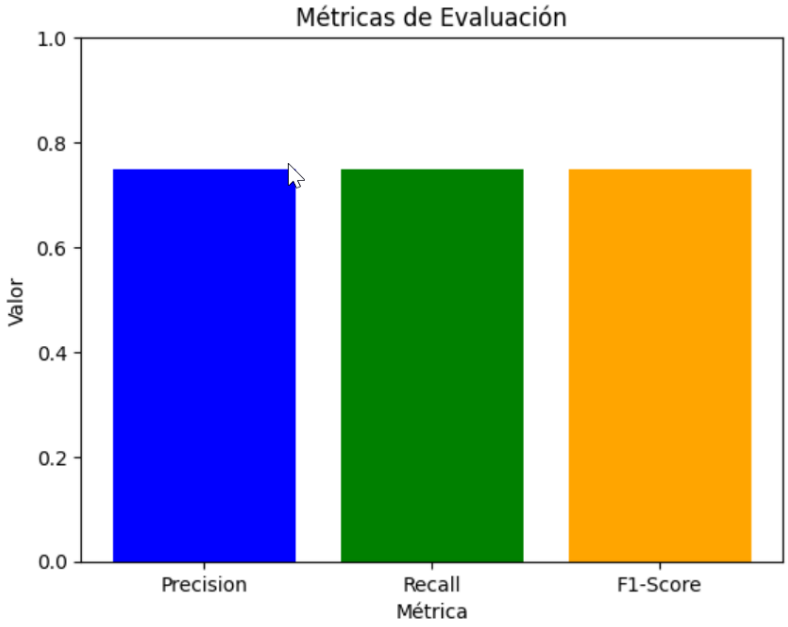

Figura 7: Gráfico de barras del desempeño del Modelo

El resultado de las métricas depende de la evaluación del modelo y de los datos utilizados para el entrenamiento y prueba. Cada una se calcula utilizando diferentes fórmulas. F1-Score depende de las primeras métricas y busca encontrar un equilibrio entre ellas.
El tener valores iguales en las métricas es un indicio que el modelo está obteniedo resultados consistentes, también se le atribuye a la sencillez del modelo.

6. CONCLUSIONES

1. **K-Nearest Neighbors (KNN) como algoritmo de clasificación:** El código implementa el algoritmo K-Nearest Neighbors (KNN) para la clasificación. KNN es un algoritmo simple pero efectivo que se basa en encontrar los vecinos más cercanos a un nuevo punto para predecir su clase. En este caso, se utilizó la distancia euclidiana como métrica de distancia.

2. **Preparación y procesamiento de datos:** Los datos se leen de un archivo CSV y se filtran para separar a los clientes que cumplieron y no cumplieron con el pago del crédito. Las características relevantes (edad y monto del crédito) se seleccionan y se escalan para que todas las características tengan el mismo peso.

3. **Evaluación de Riesgo Crediticio:** La implementación demuestra cómo se puede utilizar un enfoque de vecinos más cercanos (KNN) y un árbol KD para evaluar el riesgo crediticio de nuevos solicitantes. Esta técnica permite clasificar a los solicitantes en función de su similitud con los clientes existentes en términos de edad y monto de crédito.

4. **Importancia del KD-Tree:** La estructura KD-Tree se utiliza para organizar eficientemente los datos y acelerar la búsqueda de vecinos cercanos. Esto es especialmente útil cuando el conjunto de datos es grande, ya que mejora el tiempo de búsqueda y hace que la clasificación de nuevos solicitantes sea más rápida.

5. **Importancia de los Datos de Entrenamiento:** La calidad y representatividad de los datos de entrenamiento influyen en la precisión del modelo. Una base de datos con un número suficiente de ejemplos y una distribución equilibrada de clases puede conducir a decisiones más confiables. La inclusión de más atributos y datos de diferentes fuentes también puede mejorar la precisión.

6. **Toma de Decisiones Basada en Vecinos Cercanos:** La clasificación de un nuevo solicitante se basa en la categoría de sus vecinos más cercanos. En este caso, se utiliza un valor de k=3 para determinar la clase. Esta decisión se basa en la proporción de vecinos cercanos que pertenecen a la clase "Sí pagó", lo que influye en la decisión final.

7. **Visualización de Resultados:** La representación gráfica de los clientes buenos y malos, así como del nuevo solicitante en el espacio de características (edad y monto de crédito), proporciona una visión clara de cómo se toman las decisiones de clasificación. Esta visualización permite entender mejor cómo se separan las clases en función de estas características.

En general, la implementación demuestra cómo un enfoque de aprendizaje automático basado en KNN y KD-Tree puede ser utilizado para la evaluación crediticia. Sin embargo, es importante recordar que este enfoque es una simplificación y que una implementación más completa requeriría pruebas exhaustivas y validación en conjuntos de datos más grandes y diversos.

Referencias

[1] Ethem Alpaydin. *Introduction to Machine Learning.* 2014.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* 2006.

[3] George T. Heineman, Gary Pollice, and Stanley Selkow. *Algorithms in a Nutshell.* 2009.