UNIVERSIDAD DE GRANADA

Tratamiento Inteligente de Datos

Práctica 5

Autor

Antonio José Muriel Gálvez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Granada, 17 de enero de 2025

Índice

Enunciado

El objetivo del proyecto es desarrollar un modelo predictivo utilizando el software KNIME, que permita evaluar el rendimiento de los jugadores de la NBA en la temporada 2021-2022. A partir de un dataset que contiene múltiples estadísticas individuales de los jugadores, se busca crear nuevas métricas de rendimiento ofensivo y defensivo, clasificarlas en categorías cualitativas, y finalmente entrenar modelos de aprendizaje automático para predecir estas categorías. La solución propuesta debe manejar el preprocesamiento de datos, la creación de nuevas variables, y la evaluación de diversos algoritmos de clasificación para identificar el modelo más adecuado.

Introducción

En el mundo del deporte profesional, la capacidad de analizar y predecir el rendimiento de los jugadores es crucial para la toma de decisiones estratégicas. La NBA, como una de las ligas de baloncesto más competitivas, recopila una gran cantidad de datos estadísticos de cada temporada. En este proyecto, se utilizará KNIME, una plataforma de análisis de datos visual, para procesar y analizar estas estadísticas, creando un flujo de trabajo que abarca desde la limpieza y transformación de datos hasta la implementación de modelos predictivos. El objetivo es diseñar un sistema que clasifique el rendimiento de los jugadores en términos ofensivos y defensivos, proporcionando insights valiosos que podrían ser utilizados para el scouting, la planificación de partidos o la gestión de equipos.

Solución Knime

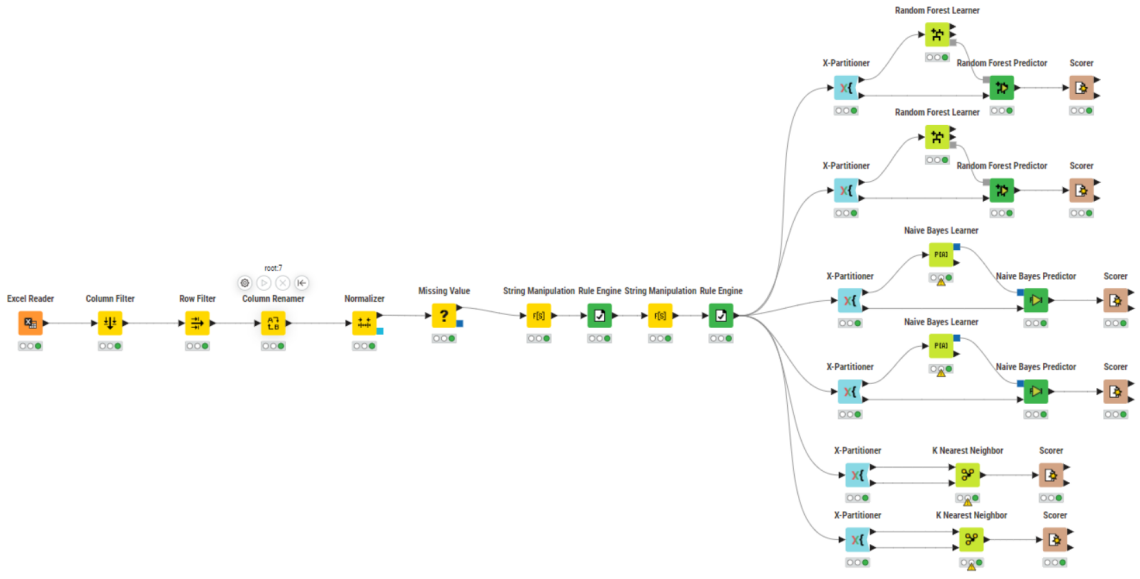


Figura 1: Solución KNIME

Nodo: Excel Reader

El nodo Excel Reader en KNIME es el punto inicial del flujo de trabajo de análisis, y su función principal es importar los datos de un archivo de Excel que contiene estadísticas detalladas de los

jugadores de la NBA de la temporada 2021-2022. El archivo cargado incluye una gran variedad de columnas que ofrecen información integral sobre el rendimiento y las características de cada jugador.

- FULL NAME: Nombre completo del jugador.
- TEAM: Equipo al que pertenece el jugador.
- POS: Posición del jugador en el equipo (e.g., base, alero).
- AGE: Edad del jugador.
- GP (Games Played): Número de juegos jugados en la temporada.
- MPG (Minutes Per Game): Minutos jugados por partido.
- Minutes Percentage: Porcentaje de minutos jugados respecto al total posible.
- Usage Rate: Porcentaje de jugadas ofensivas donde el jugador tuvo participación.
- Turnover Rate: Tasa de pérdidas de balón en jugadas ofensivas.
- FTA (Free Throw Attempts): Intentos de tiros libres realizados.
- FT
- 2PA (Two-Point Attempts): Intentos de tiros de dos puntos realizados.
- 2P
- 3PA (Three-Point Attempts): Intentos de tiros de tres puntos realizados.
- 3P
- Effective Shooting Percentage: Porcentaje efectivo de tiro, que pondera tiros de campo y triples.
- True Shooting Percentage: Porcentaje verdadero de tiro, que incluye tiros libres, de campo y triples.
- PPG (Points Per Game): Puntos por partido anotados.
- RPG (Rebounds Per Game): Rebotes por partido capturados.
- Total Rebound Percentage: Porcentaje de rebotes totales capturados.
- APG (Assists Per Game): Asistencias por partido realizadas.
- Assist Percentage: Porcentaje de asistencias en jugadas ofensivas.
- SPG (Steals Per Game): Robos por partido realizados.
- BPG (Blocks Per Game): Bloqueos por partido realizados.
- TOPG (Turnovers Per Game): Pérdidas de balón por partido.
- Versatility Index: Índice que mide la versatilidad del jugador, considerando su capacidad en puntos, asistencias y rebotes.
- Offensive Rating: Eficiencia ofensiva individual del jugador.
- Defensive Rating: Eficiencia defensiva individual del jugador.

Nodo:Column Filter

Se configura el nodo para incluir o excluir columnas específicas. En este caso, se seleccionó manualmente la columna RANK para ser eliminada. Esta columna no contenía información relevante en ninguna de sus filas y no aportaba datos útiles para el análisis del rendimiento de los jugadores.

El resto de las columnas se conservaron porque contienen información necesaria para el análisis de rendimiento y para la creación de nuevas métricas de predicción.

Nodo: Row Filter

El nodo fue configurado para detectar y eliminar la primera fila, basada en la ausencia de datos o valores relevantes. Esta fila estaba vacía, lo que significa que no contenía datos útiles. Mantener filas vacías puede interferir con los análisis posteriores y con el entrenamiento de los modelos predictivos.

Nodo: Column Renamer

En este flujo de trabajo, el nodo Column Renamer tiene como objetivo simplificar los nombres de las columnas del dataset, eliminando las descripciones largas y dejando solo los nombres más concisos y claros. Esto facilita el manejo del dataset durante el análisis y mejora la legibilidad al trabajar con los datos.

El propósito de esta simplificación es hacer que los nombres de las columnas sean más fáciles de manejar en los pasos siguientes del flujo de trabajo. Nombres más cortos y descriptivos mejoran la claridad y eficiencia en el uso de los nodos y la interpretación de los resultados.

Nodo: Normalizer

El objetivo de este nodo es normalizar las columnas numéricas del dataset para asegurarse de que todas las variables estén en la misma escala. Esto es crucial porque las características con diferentes rangos de valores pueden sesgar los resultados del modelo y hacer que algunos atributos tengan más peso que otros durante el entrenamiento.

- Columnas Seleccionadas: Se seleccionaron todas las columnas numéricas para ser normalizadas. Esto incluye variables como MPG (Minutos por partido), FTA (Intentos de tiros libres), PPG (Puntos por partido), RPG (Rebotes por partido), entre otras métricas estadísticas.
- Método de Normalización: Se utilizó la técnica Min-Max, que reescala los valores de cada columna para que estén en un rango entre 0 y 1.

El método Min-Max fue seleccionado porque proporciona un escalado lineal directo entre 0 y 1, lo que es especialmente útil cuando las variables tienen diferentes unidades o rangos. Esto asegura que cada variable tenga la misma importancia durante el entrenamiento del modelo.

Nodo: Missing Value

El propósito principal de este nodo en el flujo de trabajo es eliminar las filas que contienen valores faltantes, asegurando que el conjunto de datos sea completo y no contenga ninguna entrada con datos

incompletos. Esto es esencial, ya que los modelos de aprendizaje automático, como los árboles de decisión o las regresiones, no pueden entrenarse correctamente si existen valores nulos.

El nodo Missing Value está configurado para eliminar cualquier fila que contenga valores faltantes en cualquier columna, tanto numérica como categórica.

Se optó por eliminar las filas con valores faltantes en lugar de imputar los valores. Esto se debe a que los valores faltantes son pocos y su eliminación no debería afectar significativamente el tamaño del conjunto de datos. Además, algunas columnas contienen información crítica para los modelos predictivos, y no es recomendable realizar una imputación arbitraria en estos casos.

Nodo: String Manipulation

En este flujo de trabajo, el nodo String Manipulation se utiliza para crear dos nuevas columnas que representen el rendimiento ofensivo y defensivo de los jugadores de la NBA. Estas columnas se calculan mediante fórmulas basadas en las estadísticas de rendimiento de los jugadores. La creación de estas nuevas métricas es fundamental para poder predecir el rendimiento global de cada jugador, utilizando las métricas individuales ya presentes en el dataset.

- Rendimiento Ofensivo: Calculado a partir de varias estadísticas ofensivas, como los juegos jugados (GP), minutos jugados por partido (MPG), porcentaje de uso (Usage Rate), tiros de campo, y otras métricas de eficiencia ofensiva.
- Rendimiento Defensivo: Calculado a partir de estadísticas defensivas como los rebotes (RPG), robos (SPG), bloqueos (BPG), entre otros.

Fórmula de Rendimiento Ofensivo:

```
string(
    (0.15 * toDouble($GP$)) +
    (0.2 * toDouble($MPG$)) +
    (0.2 * toDouble($USG%$)) +
    (0.05 * toDouble($FTA$)) +
    (0.05 * toDouble($FT%$)) +
    (0.1 * toDouble($2P%$)) +
    (0.1 * toDouble($3PA$)) +
    (0.1 * toDouble($3P%$)) +
    (0.15 * toDouble($eFG%$)) +
    (0.15 * toDouble($TS%$)) +
    (0.25 * toDouble($ORTG$))
)
```

Fórmula de Rendimiento Defensivo:

```
string(
    (0.2 * toDouble($MPG$)) +
    (0.2 * toDouble($RPG$)) +
    (0.15 * toDouble($SPG$)) +
    (0.15 * toDouble($BPG$)) +
    (0.1 * toDouble($TOPG$)) +
    (0.2 * toDouble($TRB%$)) +
    (0.25 * toDouble($DRTG$))
)
```

nodo String Manipulation es crucial porque permite transformar las estadísticas existentes en nuevas métricas que resumen el rendimiento global de los jugadores en áreas clave (ofensiva y defensiva). Estas métricas pueden ser utilizadas más tarde en el modelo para predecir el rendimiento general de los jugadores, lo que es el objetivo final del análisis.

Nodo: Rule Engine

El nodo Rule Engine se utiliza para convertir las métricas numéricas de rendimiento ofensivo y defensivo (calculadas previamente en el nodo String Manipulation) en categorías cualitativas. Estas categorías son útiles para interpretar el rendimiento de los jugadores de manera más comprensible, ya que convierten las métricas numéricas en clases como .ExcelenteÇ, "Bueno", "Promedioz "Bajo".

Reglas para Rendimiento Ofensivo:

```
$Rendimiento_Ofensivo$ > 0.65 => "Excelente"
$Rendimiento_Ofensivo$ > 0.50 => "Bueno"
$Rendimiento_Ofensivo$ > 0.30 => "Promedio"
TRUE => "Bajo"
```

Reglas para Rendimiento Defensivo:

```
$Rendimiento_Defensivo$ > 0.80 => "Excelente"
$Rendimiento_Defensivo$ > 0.65 => "Bueno"
$Rendimiento_Defensivo$ > 0.45 => "Promedio"
TRUE => "Bajo"
```

Se eligieron umbrales específicos para clasificar a los jugadores según su rendimiento. Estas categorías cualitativas permiten interpretar de forma sencilla el rendimiento de los jugadores, sin necesidad de revisar las métricas numéricas detalladas. Además, estas clases categóricas son más fáciles de manejar al momento de realizar análisis posteriores o cuando se entrenan modelos de clasificación.

Nodo: X-Partitioner

Divide el conjunto de datos en particiones que serán utilizadas para entrenamiento y validación del modelo. En este flujo de trabajo, se opta por la validación cruzada para evaluar el rendimiento de los modelos predictivos de forma más precisa.

Se configuró el nodo para realizar 10 particiones de validación cruzada.

Nodos Random Forest Learner / Naive Bayes Learner / K Nearest Neighbor Learner

Estos nodos son utilizados para entrenar tres modelos de aprendizaje automático diferentes: Random Forest, Naive Bayes y K Nearest Neighbor (KNN). En este flujo de trabajo, se han creado dos ramas para cada modelo, una para predecir el Rendimiento Ofensivo y otra para predecir el Rendimiento Defensivo. La idea es evaluar cómo cada modelo predice el rendimiento de los jugadores en estas dos áreas clave.

- Random Forest Learner: El Random Forest es un modelo de aprendizaje supervisado basado en múltiples árboles de decisión. Este modelo crea varios árboles de decisión durante el proceso de entrenamiento y luego promedia los resultados para mejorar la precisión y reducir el sobreajuste.
- Naive Bayes Learner: El Naive Bayes es un modelo probabilístico basado en el teorema de Bayes, que supone que las características son independientes entre sí. Es un modelo sencillo pero eficaz, especialmente útil cuando las relaciones entre las características no son demasiado complejas.
- K Nearest Neighbor (KNN) Learner: El modelo KNN se basa en la idea de que los puntos de datos similares están cerca unos de otros en el espacio de características. En este caso, el modelo predice el rendimiento de un jugador en función de los jugadores más cercanos en el espacio de características.

Se han excluido las columnas Full NAME, TEAM, POS, AGE, y la columna correspondiente a la métrica de rendimiento que no se está prediciendo (es decir, si se predice el rendimiento ofensivo, se excluye la columna de rendimiento defensivo, y viceversa). Esto asegura que las variables relevantes para la predicción del rendimiento específico sean las que se utilicen en el modelo, sin incluir datos irrelevantes.

Nodos: Random Forest Predictor / Naive Bayes Predictor / K Nearest Neighbor Predictor

Estos nodos se utilizan para hacer predicciones con los modelos entrenados en el paso anterior. El Random Forest Predictor, Naive Bayes Predictor, y K Nearest Neighbor Predictor aplican los modelos previamente entrenados a los datos de prueba para predecir las categorías de rendimiento ofensivo y defensivo de los jugadores.

- Random Forest Predictor: Este nodo aplica el modelo Random Forest entrenado para hacer predicciones sobre las categorías de rendimiento ofensivo o defensivo para los datos de prueba.
- Naive Bayes Predictor: Similar al Random Forest Predictor, este nodo utiliza el modelo Naive Bayes entrenado para realizar predicciones sobre los datos de prueba.
- K Nearest Neighbor Predictor: Este nodo utiliza el modelo K Nearest Neighbor para hacer predicciones basadas en las distancias a los puntos más cercanos en el conjunto de datos de prueba.

Scorer:

Las matrices de confusión proporcionan una visión detallada de las predicciones y permiten evaluar el rendimiento de cada modelo en función de cómo clasifica las instancias en las categorías definidas ("Bueno", "Promedio", .ᴱˣcelente", "Bajo"). A continuación se desglosa cada matriz para los tres modelos en las tareas de predicción del rendimiento ofensivo y defensivo.

Random Forest

Rendimiento Ofensivo:

El modelo Random Forest para la predicción del rendimiento ofensivo muestra una fuerte precisión en predecir la categoría Bueno (44 aciertos). La categoría Excelente también tiene un buen rendimiento con 6 aciertos, pero la categoría Bajo tiene 1 acierto erróneo que se confunde con Promedio.

Rows: 4 | Columns: 4 — Table | Statistics

#	RowID	Bueno (Number (integer))	Promedio (Number (integer))	Excelente (Number (integer))	Bajo (Number (integer))
1	Bueno	44	0	0	0
2	Prome	0	15	0	0
3	Excele	0	0	6	0
4	Bajo	0	1	0	1

Figura 2: Random Forest Ofensivo

Rendimiento Defensivo:

El modelo Random Forest para el rendimiento defensivo tiene un desempeño sólido. Predice correctamente la categoría Excelente (17 aciertos). Las categorías Bueno y Promedio muestran una distribución razonable, aunque algunas predicciones de Bueno se confunden con Promedio.

Rows: 4 | Columns: 4 — Table | Statistics

#	RowID	Bueno (Number (integer))	Promedio (Number (integer))	Excelente (Number (integer))	Bajo (Number (integer))
1	Bueno	36	0	0	0
2	Prome	1	11	0	1
3	Excele	0	0	17	0
4	Bajo	0	1	0	0

Figura 3: Random Forest Ofensivo

Naive Bayes

Rendimiento Ofensivo:

El modelo Naive Bayes muestra una buena precisión en la predicción de la categoría Bueno (40 aciertos). Sin embargo, también hay algo de confusión, especialmente en las categorías Promedio y Excelente, que se mezclan con los otros. La categoría Bajo tiene 2 aciertos, lo que muestra que el modelo tiene algo de dificultad para clasificar correctamente en esta categoría.

Rows: 4 | Columns: 4 — Table | Statistics

#	RowID	Bueno (Number (integer))	Promedio (Number (integer))	Excelente (Number (integer))	Bajo (Number (integer))
1	Bueno	40	5	5	2
2	Prome	1	9	0	0
3	Excele	0	0	3	0
4	Bajo	0	0	0	2

Figura 4: Naive Bayes Learner Ofensivo

Rendimiento Defensivo:

Para el rendimiento defensivo, Naive Bayes muestra una buena capacidad para predecir Excelente (16 aciertos) y Bueno (23 aciertos). Sin embargo, las categorías Promedio y Bajo tienen una mayor confusión, ya que hay predicciones incorrectas en Bajo.

Rows: 4 | Columns: 4

Table | Statistics

#	RowID	Bueno (Number (integer))	Promedio (Number (integer))	Excelente (Number (integer))	Bajo (Number (integer))
1	Bueno	23	5	1	4
2	Prome	1	5	0	2
3	Excele	4	0	16	5
4	Bajo	0	0	0	1

Figura 5: Naive Bayes Defensivo

K-Nearest Neighbor

Rendimiento Ofensivo:

Interpretación: El modelo KNN para el rendimiento ofensivo tiene un desempeño impresionante, especialmente en la categoría Bueno (103 aciertos). Sin embargo, también muestra cierta confusión en Promedio, con 13 aciertos incorrectos en esta categoría.

Rows: 4 | Columns: 4

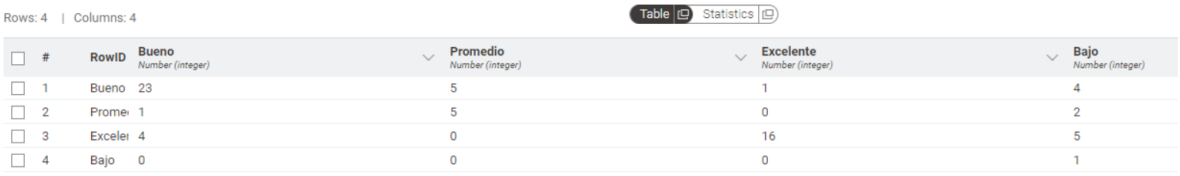
Table | Statistics

#	RowID	Bueno (Number (integer))	Promedio (Number (integer))	Excelente (Number (integer))	Bajo (Number (integer))
1	Bueno	103	2	1	0
2	Prome	7	13	0	0
3	Excele	3	0	2	0
4	Bajo	0	1	0	1

Figura 6: K Nearest Ofensivo

Rendimiento Defensivo:

Para el rendimiento defensivo, KNN muestra una fuerte predicción en la categoría Bueno (60 aciertos) y en Excelente (14 aciertos). Las categorías Promedio y Bajo presentan algunas confusiones, pero el modelo sigue siendo bastante preciso en general.



Rows: 4 | Columns: 4

Table | Statistics

#	RowID	Bueno (Number (integer))	Promedio (Number (integer))	Excelente (Number (integer))	Bajo (Number (integer))
1	Bueno	23	5	1	4
2	Prome	1	5	0	2
3	Excele	4	0	16	5
4	Bajo	0	0	0	1

Figura 7: Naive Bayes Defensivo

Conclusión

Las matrices de confusión proporcionan una visión clara del desempeño de cada modelo. Aquí hay algunos puntos clave:

- Random Forest: Este modelo muestra un buen rendimiento, especialmente en la clasificación de Bueno y Excelente, con pocos errores en las predicciones.
- Naive Bayes: Aunque tiene un buen rendimiento general, presenta más confusión en las predicciones entre categorías cercanas, especialmente para Bajo.

- K-Nearest Neighbor (KNN): Este modelo tiene una excelente capacidad para predecir Bueno, pero sufre más de confusión en las categorías de Promedio.

El Random Forest parece ser el modelo con mayor precisión en términos de clasificación correcta, pero es útil comparar los tres para entender cuál se adapta mejor a los datos y cuál es más adecuado para predecir el rendimiento ofensivo y defensivo de los jugadores.