Curso: Machine Learning Models
Catedrático: Christian
Auxiliar: Javier

Proyecto 1-Reporte

Mayco Castellanos
20210273

Reporte de Modelos de Clasificación para Predecir el Ganador de Partidos de Tenis

1. Introducción

El propósito de este reporte es presentar el proceso de desarrollo y evaluación de un modelo de clasificación para predecir el ganador de partidos de tenis. Utilizando un dataset de partidos oficiales de tenis, se exploraron dos diferentes algoritmos de machine learning, se realizó un análisis exploratorio de datos (EDA) y se optimizaron los modelos para mejorar su rendimiento. El objetivo final es determinar si es posible predecir con precisión el resultado de un partido basándose en características como el ranking de los jugadores, los puntos ATP y las probabilidades de apuestas.

En el mundo del tenis, predecir el resultado de un partido es un desafío complejo debido a la naturaleza dinámica del deporte y la variedad de factores que influyen en el rendimiento de los jugadores. Sin embargo, con el uso de los algoritmos de clasificación Kneighbors y SGD, es posible analizar patrones históricos y hacer predicciones informadas. Este proyecto tiene aplicaciones potenciales en apuestas deportivas, análisis tácticos y estrategias de entrenamiento.

2. Datos

Descripción del Dataset

El dataset utilizado contiene información sobre partidos oficiales de tenis desde el año 2000 hasta 2025. Las principales variables del dataset son las siguientes:

Variables Predictoras:

- **Rank_1, Rank_2**: Ranking de los jugadores.
- **Pts_1, Pts_2**: Puntos ATP de los jugadores.
- **Odd_1, Odd_2**: Probabilidades de apuestas para cada jugador.

  Mediante Feature Engineering:

- **Gano_el_que_tenia_mas_puntos**: Indica si el jugador con más puntos ATP ganó el partido.
- **Gano_el_que_tenia_menor_odd**: Indica si el jugador con menor odd ganó el partido.

Variable de Resultado:

- **Winner_binary**: Indica si el ganador fue Player_1 (1) o Player_2 (0).

3. Métodos

Se utilizaron visualizaciones como histogramas, gráficos de correlación y boxplots para explorar los datos.

Se realizó un análisis exploratorio de datos (EDA) para entender la distribución de las variables y detectar valores faltantes o anomalías.

Se escalaron las variables numéricas para asegurar que estuvieran en la misma escala.

Se balancearon las clases utilizando la técnica SMOTE para manejar el desbalance en la variable de resultado.

Algoritmos Utilizados

Se evaluaron dos algoritmos de clasificación:

1. **SGDClassifier**: Un modelo lineal que utiliza descenso de gradiente estocástico para la clasificación.
2. **KNeighborsClassifier**: Un modelo basado en instancias que clasifica según los "vecinos más cercanos".

Proceso de Entrenamiento y Selección de Parámetros

- **División de Datos**: Los datos se dividieron en conjuntos de entrenamiento (80%) y prueba (20%).
- **Escalado de Variables**: Se aplicó **StandardScaler** para normalizar las variables numéricas.
- **Balanceo de Clases**: Se utilizó **SMOTE** para balancear las clases en el conjunto de entrenamiento.
- **Ajuste de Hiperparámetros**: Se utilizó **Grid Search** para optimizar los hiperparámetros de ambos modelos.

4. Resultados

Rendimiento de los Modelos

SGDClassifier:

- **Accuracy**: 73%
- **Precisión**: 0.72
- **Recall**: 0.73
- **F1-Score**: 0.72

KNeighborsClassifier:

- **Accuracy**: 70%
- **Precisión**: 0.69
- **Recall**: 0.70
- **F1-Score**: 0.69

Comparación con el Baseline

- **Baseline de Mejor Ranking**: Accuracy del 65%.

Ambos modelos superaron el baseline.

Matriz de Confusión

SGDClassifier:

- Verdaderos Positivos (TP): 3200
- Falsos Positivos (FP): 700
- Falsos Negativos (FN): 800
- Verdaderos Negativos (TN): 3300

KNeighborsClassifier:

- Verdaderos Positivos (TP): 4900
- Falsos Positivos (FP): 10
- Falsos Negativos (FN): 5
- Verdaderos Negativos (TN): 4895

5. Conclusión

Hallazgos

- El **KNeighborsClassifier** en principio mostró un rendimiento muy alto, por lo que para evitar problemas de overfitting se entrenó el modelo con menos variables predictoras. Con este cambió tuvo un rendimiento de 70% de accuracy.
- El **SGDClassifier** tuvo un rendimiento aceptable (73% de accuracy), pero es más robusto y menos propenso a overfitting.
- Como producto del feature Engineering, las variables nuevas: **Gano_el_que_tenia_mas_puntos** y **Gano_el_que_tenia_menor_odd** aportaron valor al modelo, mejorando ligeramente el rendimiento.

Recomendaciones finales

1. **Evitar problemas de Overfitting**: Es importante tomar algunas medidas cuando se trabaja con **KNeighborsClassifier**. Se recomienda trabajar con subconjuntos de variables predictoras en caso tenga muchas, realizar validación cruzada, estandarización de variables y ajustar los hiperparámetros entre otras medidas.
2. **Mejorar el Balance de Clases**: Aplicar técnicas avanzadas de balanceo de clases si existe algún desbalance sobre la variable objetivo..

.