Modelos y Simulación II: Proyecto del Curso

Paz Gutierrez Kadyha, Martínez Galeano Daniel
Facultad de ingeniería, Universidad de Antioquia
Medellín, Colombia
kadyha.paz@udea.edu.co
daniel.martinezg@udea.edu.co

Abstract—En esta actividad se busca presentar todo el diseño, análisis y simulación de un sistema de predicción basado en técnicas de aprendizaje de máquina; describiendo el problema y su contexto en términos del estado del arte, especificando cada una de las etapas del desarrollo del trabajo, los modelos con sus respectivas restricciones, la metodología de validación, los resultados de las simulaciones y las conclusiones obtenidas.

***Index Terms*—predicción, clasificación, aprendizaje de máquina.**

I. COMPRENSIÓN DEL PROBLEMA DE ML

A. Descripción del problema de predicción

El problema de predicción que se aborda es sobre clasificación de ligas de videojuegos, actualmente muchos juegos dividen a los jugadores según su nivel de experiencia o habilidad, en este caso se dividen en ligas, las posibles ligas son bronce, plata, oro, platino, diamante, entre otros. Esto es similar al sistema de medallas que se usa en competencias deportivas, donde el primer lugar tiene oro, el segundo plata y el tercero bronce. En este caso se tiene un problema de clasificación, se quiere predecir la liga a la que pertenece un jugador, aprovechando datos sobre los jugadores como, la edad, total de horas jugadas, entre otros.

B. Variables de entrada

Las variables de entrada son todas de tipo numérico y se encuentran listadas a continuación:

- GameID: Un número que identifica cada juego. (Entero)
- Age: Un número con la edad de cada jugador. (Entero)
- Hours Per Week: Horas dedicadas a jugar por semana. (Entero)
- Total Hours: Total de horas jugadas. (Entero)
- APM: Acciones realizadas por minuto. (Entero)
- Select By Hotkeys: Cantidad de selecciones de unidades o edificios realizadas mediante teclas de acceso rápido realizadas por unidad de tiempo. (Real)
- Assign To Hotkeys: Número de unidades o edificios asignados a teclas de acceso rápido por unidad de tiempo. (Real)
- Unique Hotkeys: Número de teclas de acceso rápido únicas utilizadas por unidad de tiempo. (Real)
- Minimap Attacks: Número de acciones de ataque en el minimapa por unidad de tiempo. (Real)
- Minimap Right Clicks: Número de clicks con el botón derecho en el minimapa por unidad de tiempo. (Real)
- Number Of PACs: Número de PACs (Ciclo de percepción-acción) realizados por unidad de tiempo. (Real)
- Gap Between PACs: Duración media en milisegundos entre PACs. (Real)
- Action Latency: Latencia media desde el inicio de un PACs hasta su primera acción en milisegundos. (Real)
- Actions In PAC: Media de acciones dentro de cada PAC. (Real)
- Total Map Explore: Cantidad del mapa que recorre un jugador por unidad de tiempo. (Real)
- Workers Made: Número de los SCVs, drones y sondas entrenadas por unidad de tiempo. (Real)
- Unique Units Made: Unidades únicas realizadas por unidad de tiempo. (Real)
- Complex Units Made: Número de fantasmas, infestadores y altos templarios entrenados por unidad de tiempo. (Real)
- ComplexAbilitiesUsed: Uso de habilidades que requieren instrucciones específicas de orientación utilizadas por unidad de tiempo. (Real)

C. Variables a predecir

La variable a predecir es llamada League Index y se refiere a 8 niveles de habilidad que son clasificados como: Bronze, Silver, Gold, Platinum, Diamond, Master, Grand Master, y Profesional. En la base de datos cada nivel es representado con un número del 1 al 8.

D. Valores faltantes

Las entradas Age, Hour per week y total hours, son valores faltantes únicamente para la liga número 8 que corresponde a la categoría professional, por la falta de información sobre dichas características, además de la poca cantidad de muestras que se tienen en esta categoría y la dificultad para completar los datos faltantes usando otros métodos, se considera adecuado eliminar la categoría Professional, conservando las 3 clases Age, Hour per week y total hours para evaluar las demás categorías.

E. Estado del arte

En el artículo Video Game Telemetry as a Critical Tool in the Study of Complex Skill Learning [1] se puede observar cómo se aborda el mismo problema que actualmente estamos abordando en este documento. En este utilizan random forest como técnica de aprendizaje, los bosques fueron creados usando la función cforest en R con 1000 árboles y 5 variables por split además, se evaluó la aleatoriedad en el algoritmo ejecutando el bosque con el 70% de muestras sin reemplazo de los datos originales, esto se hizo 25 veces.

Se evaluaron los datos haciendo divisiones entre las clases, por ejemplo clasificando a los jugadores únicamente en dos grupos, que serían novato y experto, o evitando las clases vecinas, esto puede hacerse omitiendo una clase por cada dos, esto es usar un clasificación de la siguiente manera: Bronze, Gold, Diamond, Grand Master. Al hacer esto se encontró que algunas variables presentan más importancia predictiva que otras según la forma en la que se dividieron las clases.

En el artículo Player Skill Modeling in Starcraft II [2] se afronta el mismo problema. La metodología usada es bootstrapping tomando el 70 % de los datos para el entrenamiento y el 30 % para la validación, en este artículo se evalúan 6 modelos diferentes, y allí se puede evidenciar que el que mejor resultados trae es la implementación del SVM y del Random forest.

Los resultados muestran que en promedio el 67% de las predicciones asignan a un jugador a una liga vecina. Al considerar la forma en que se asignan las ligas, que un jugador se coloque en una liga más abajo o superior a su nivel de habilidad es en realidad común, y por lo tanto tales clasificaciones erróneas son de esperar.

Adicional se tuvo en cuenta el artículo A machine learning framework for sport result prediction [3] En dónde aunque no se aborda el mismo problema, intenta predecir los resultados deportivos

y así mismo beneficiar a las casas de apuestas y medios de comunicación utilizando algoritmos con base en redes neuronales. Se utiliza cross-validation como metodología de validación y este artículo recomienda usar ROC como medida de evaluación en los casos en los que los datos están muy desequilibrados.

Entre los resultados se encontró que al evaluar modelos que predicen resultados en partidos de baloncesto, las temporadas anteriores pueden no ser relevantes para predecir partidos en temporadas futuras, esto se puede aplicar particularmente en deportes donde las fortalezas de los equipos pueden cambiar significativamente de año en año.

El último artículo que se decidió tener en cuenta es Prediction of Euroleague Games based on Supervised Classification Algorithm k-Nearest Neighbours [4] En el cual se aborda un problema de clasificación usando el método k-vecinos más cercanos (k-nn), se usó el 80% de los datos para entrenamiento y el 20% para pruebas.

Se realizaron dos pruebas con diferentes características usando el método k-nn y se encontró que la selección y la agrupación de características de entrada tiene más impacto en la precisión del modelo que el coeficiente k.

II. ENTRENAMIENTO Y EVALUACIÓN DE LOS MODELOS

A. Experimentos

La base de datos usada se llama SkillCraft1 Master Table Dataset Data Set [5], está base de datos consta de 3395 muestras con 19 atributos y con una salida que corresponde a 8 posibles clases, que se representan con los números del 1 al 8. El número de muestras por clase se lista a continuación:

- Clase 1: 167
- Clase 2: 347
- Clase 3: 553
- Clase 4: 811
- Clase 5: 806
- Clase 6: 621
- Clase 7: 35
- Clase 8: 55

Se usará la validación cruzada estratificada, usando 80% para la fase de entrenamiento, pruebas y 20% para validación. Las medidas de desempeño usadas para evaluar el sistema serán: Matthews correlation coefficient (MCC) y Balanced Accuracy (BACC), siendo Balanced Accuracy la medida principal.

Los scripts con el análisis de la base de datos y los siguientes puntos de este documento se encuentran en el siguiente repositorio: https://github.com/Kadyha/SkillCraft1_Dataset.

B. Evaluación de modelos

1. Discriminante Cuadrático

En la figura 1 se presentan los resultados obtenidos durante la fase de validación, incluyendo los hiperparametros usados y las medidas de desempeño.

	Reg param	Balanced Accuracy train	Balanced Accuracy test	Std train	Std test	Matthews corrcoef train	Matthews corrcoef test	resta de accuracy
0	0.000	0.509235	0.310565	0.013323	0.013553	0.281347	0.182011	0.198671
1	0.001	0.457755	0.319355	0.054386	0.023381	0.279663	0.200836	0.138400
2	0.004	0.440582	0.322307	0.052040	0.025106	0.279057	0.207140	0.118275
3	0.100	0.432768	0.325029	0.048232	0.026129	0.279002	0.210445	0.107739
4	0.200	0.428313	0.327009	0.044809	0.026503	0.278585	0.213143	0.101305
5	0.300	0.426866	0.328941	0.041330	0.026799	0.278921	0.214961	0.097925
6	0.500	0.427804	0.329813	0.038515	0.026712	0.278841	0.215736	0.097991
7	0.800	0.428909	0.331438	0.036324	0.027089	0.276710	0.214314	0.097471
8	0.900	0.429589	0.335070	0.034598	0.029945	0.273557	0.212924	0.094519
9	1.000	0.415955	0.331082	0.052497	0.034691	0.254888	0.200606	0.084873

Figura 1 Resultados durante la fase de validación con discriminante cuadrático

En la figura 2 se pueden observar los resultados en el conjunto de validación usando el mejor conjunto de hiperparametros.

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.274676	0.0	0.058958

Figura 2 Mejor conjunto de hiper parámetros con Discriminante cuadrático

2. Gradient Boosting Tree

En la figura 3 se presentan los resultados obtenidos durante la fase de validación, incluyendo los hiperparametros usados y las medidas de desempeño.

	# Trees	# Max_depth	Accuracy Train	Accuracy Test	Std Train	Std Test	Matthews corrcoef train	Matthews corrcoef test	resta de accuracy
0	1.0	1.0	0.177680	0.171920	0.003807	0.007346	0.096936	0.080464	0.005760
1	1.0	4.0	0.316669	0.196852	0.018058	0.012789	0.211793	0.108071	0.119817
2	1.0	9.0	0.699239	0.226949	0.021889	0.027385	0.644347	0.126160	0.472290
3	1.0	20.0	0.991524	0.247189	0.004480	0.030998	0.989841	0.132442	0.744335
4	1.0	50.0	1.000000	0.235353	0.000000	0.036597	1.000000	0.124548	0.764647
5	1.0	75.0	1.000000	0.235917	0.000000	0.027696	1.000000	0.123427	0.764083
6	1.0	100.0	1.000000	0.234931	0.000000	0.028974	1.000000	0.118487	0.765069
7	3.0	1.0	0.224108	0.210887	0.006486	0.014921	0.158281	0.139350	0.013221
8	3.0	4.0	0.506851	0.269727	0.021224	0.029557	0.379573	0.182071	0.237124
9	3.0	9.0	0.894049	0.278890	0.014345	0.029443	0.857494	0.170806	0.615159
10	3.0	20.0	0.999945	0.272565	0.000110	0.017188	0.999884	0.172374	0.727380
11	3.0	50.0	1.000000	0.266538	0.000000	0.029437	1.000000	0.155643	0.733462
12	3.0	75.0	1.000000	0.272137	0.000000	0.032859	1.000000	0.163246	0.727863
13	3.0	100.0	1.000000	0.272781	0.000000	0.034607	1.000000	0.160974	0.727219
14	5.0	1.0	0.266311	0.232750	0.019392	0.013431	0.179364	0.152172	0.033561
15	5.0	4.0	0.586063	0.288744	0.017127	0.018404	0.452892	0.203953	0.297319
16	5.0	9.0	0.944987	0.283672	0.013113	0.033243	0.925808	0.179372	0.661315
17	5.0	20.0	1.000000	0.280385	0.000000	0.020126	1.000000	0.167558	0.719615
18	5.0	50.0	1.000000	0.281464	0.000000	0.031046	1.000000	0.167470	0.718536

Figura 3 Resultados durante la fase de validación con Gradient Boosting Tree

En la figura 4 se pueden observar los resultados en el conjunto de validación usando el mejor conjunto de hiperparametros.

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.280526	0.0	0.196506

Figura 4 Mejor conjunto de hiper parámetros con Gradient Boosting Tree

3. Redes Neuronales Artificiales

En la figura 5 se presentan los resultados obtenidos durante la fase de validación, incluyendo los hiperparametros usados y las medidas de desempeño.

	# Hidden layers	# Neurons	Accuracy Train	Accuracy Test	Std Train	Std Test	Matthews corrcoef train	Matthews corrcoef test	resta de accuracy
0	1.0	1.0	0.263220	0.257065	0.020221	0.020150	0.221967	0.209154	0.006155
1	1.0	3.0	0.323235	0.314730	0.005718	0.010322	0.284775	0.269469	0.008504
2	1.0	5.0	0.335931	0.321338	0.018180	0.014023	0.292451	0.262807	0.014593
3	1.0	10.0	0.341853	0.325233	0.011770	0.018392	0.291242	0.258160	0.016620
4	1.0	20.0	0.362427	0.332923	0.007205	0.007753	0.300975	0.255553	0.029504
5	1.0	50.0	0.357968	0.341354	0.005258	0.019020	0.294496	0.264765	0.016615
6	1.0	100.0	0.357330	0.340727	0.006915	0.012339	0.296786	0.271027	0.016604
7	1.0	500.0	0.376162	0.325933	0.043651	0.009522	0.282500	0.244705	0.050229
8	2.0	1.0	0.247298	0.244150	0.020876	0.022033	0.206688	0.199966	0.003148
9	2.0	3.0	0.320955	0.313977	0.009994	0.012115	0.281486	0.267772	0.006978
10	2.0	5.0	0.326086	0.316931	0.014369	0.007735	0.281553	0.266265	0.009154
11	2.0	10.0	0.347102	0.333923	0.015871	0.010723	0.295178	0.271843	0.013179
12	2.0	20.0	0.329942	0.323779	0.008055	0.011753	0.284125	0.270668	0.006163
13	2.0	50.0	0.338757	0.325815	0.018754	0.018266	0.283975	0.260041	0.012941
14	2.0	100.0	0.334992	0.324894	0.008508	0.005034	0.278369	0.257725	0.010098
15	2.0	500.0	0.341794	0.328857	0.007742	0.009340	0.274750	0.256206	0.012937
16	3.0	1.0	0.223856	0.218982	0.051081	0.048765	0.159522	0.148679	0.004874
17	3.0	3.0	0.320830	0.311128	0.009133	0.003318	0.278863	0.260512	0.009702

Figura 5 Resultados durante la fase de validación con Redes Neuronales Artificiales

En la figura 6 se pueden observar los resultados en el conjunto de validación usando el mejor conjunto de hiperparametros.

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.347663	0.0	0.283631

Figura 6 Mejor conjunto de hiper parámetros con Redes Neuronales Artificiales

4. Máquinas de Soporte Vectorial

En la figura 7 se presentan los resultados obtenidos durante la fase de validación, incluyendo los hiperparametros usados y las medidas de desempeño.

	kernel	gamma	param_reg	Accuracy Train	Accuracy Test	Std Train	Std Test	% de vectores de soporte	Matthews corrcoef train	Matthews corrcoef test	resta de accuracy
0	linear	0.001	0.1	0.346768	0.318856	0.008639	0.016826	97.715296	0.293501	0.256493	0.027912
1	linear	0.001	1.0	0.361529	0.329974	0.004421	0.022066	97.465646	0.302328	0.258997	0.031555
2	linear	0.001	2.0	0.371976	0.328989	0.011794	0.020904	97.390708	0.302696	0.256719	0.042987
3	linear	0.001	5.0	0.372407	0.331681	0.015397	0.020778	97.465609	0.300605	0.259240	0.040726
4	linear	0.001	10.0	0.373592	0.329567	0.014165	0.022917	97.478121	0.301774	0.257437	0.044025
5	linear	0.010	0.1	0.346768	0.318856	0.008639	0.016826	97.715296	0.293501	0.256493	0.027912
6	linear	0.010	1.0	0.361529	0.329974	0.004421	0.022066	97.465646	0.302328	0.258997	0.031555
7	linear	0.010	2.0	0.371976	0.328989	0.011794	0.020904	97.390708	0.302696	0.256719	0.042987
8	linear	0.010	5.0	0.372407	0.331681	0.015397	0.020778	97.465609	0.300605	0.259240	0.040726
9	linear	0.010	10.0	0.373592	0.329567	0.014165	0.022917	97.478121	0.301774	0.257437	0.044025
10	rbf	0.001	0.1	0.143894	0.143301	0.000798	0.000444	100.000000	0.011146	0.004966	0.000593
11	rbf	0.001	1.0	0.233368	0.225849	0.006022	0.016645	99.775281	0.200402	0.184067	0.007519
12	rbf	0.001	2.0	0.257447	0.242071	0.008344	0.013917	99.338330	0.225240	0.200044	0.015376
13	rbf	0.001	5.0	0.290744	0.275491	0.002530	0.023028	98.676629	0.252063	0.225634	0.015253
14	rbf	0.001	10.0	0.314011	0.293490	0.007825	0.016503	98.364516	0.270289	0.238917	0.020521
15	rbf	0.010	0.1	0.227650	0.220411	0.005116	0.016096	99.937575	0.193093	0.176710	0.007239
16	rbf	0.010	1.0	0.319888	0.290699	0.005342	0.031784	98.526854	0.283581	0.238304	0.029189
17	rbf	0.010	2.0	0.352671	0.302196	0.015381	0.021100	98.177278	0.314863	0.245991	0.050476

Figura 7 Resultados durante la fase de validación utilizando Máquinas de Soporte Vectorial

En la figura 8 se pueden observar los resultados en el conjunto de validación usando el mejor conjunto de hiperparametros.

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.320197	0.0	0.272434

Figura 8: Mejor conjunto de hiper parámetros utilizando Máquinas de Soporte Vectorial

III. SELECCIÓN/EXTRACCIÓN DE CARACTERÍSTICAS

A. Análisis de características

El análisis individual de características se realizó con una matriz de correlación, como se puede observar en la figura 9.

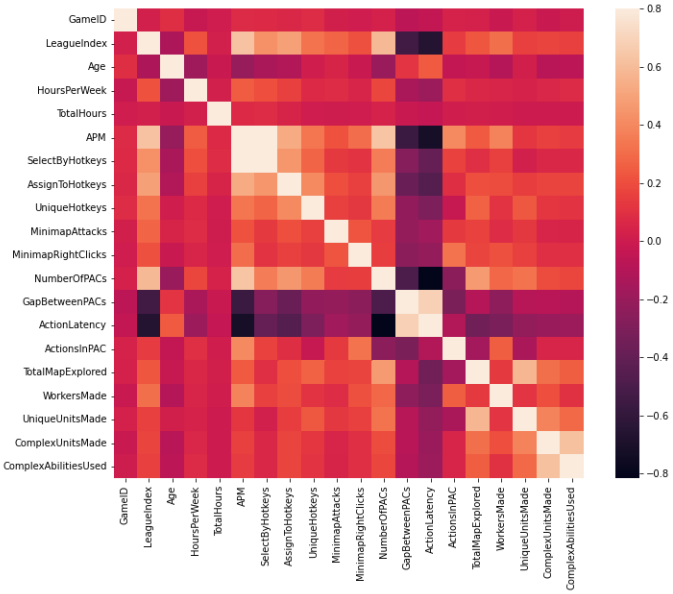

Figura 9 matriz de correlación

Estos datos también se pueden ver de forma numérica y fueron usados para saber qué características son candidatas a ser eliminadas.

Con el índice de fisher se encontró que las candidatas a ser eliminadas son en primer lugar Game ID, ya que es la variable que menos aporta al modelo y esto tiene sentido ya que se trata de un identificador único para cada juego, la segunda candidata es APM, y la tercera es ActionsInPac, todos estos resultados se pueden apreciar en la Figura 10.

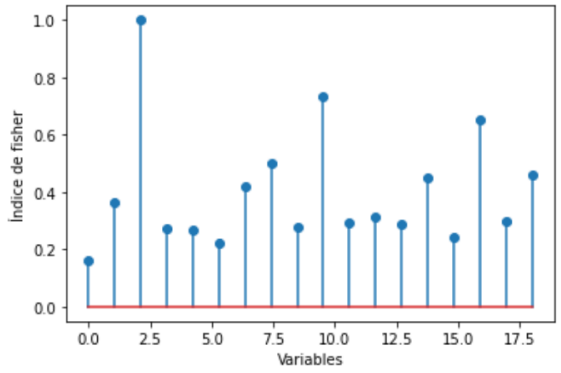

Figura 10 Índice de Fisher

B. Selección de características

Para la selección de características se eligió el método de búsqueda secuencial hacia adelante ya que se evaluaron ambos y en el caso del método de búsqueda secuencial hacia atrás no avanzó, es decir solo evaluó una vez con todas las carácteristicas y se detuvo, mientras que el otro recorrió todas las características por duplas, arrojando un modelo equivalente con menos columnas.

Los 3 mejores modelos fueron redes neuronales artificiales, la máquina de soporte vectorial y gradient boosting tree.

	1	2	3	4	5	6
feature_idx	(11,)	(2, 11)	(2, 6, 11)	(1, 2, 6, 11)	(1, 2, 3, 6, 11)	(1, 2, 3, 6, 7, 11)
cv_scores	[0.37378277153558054]	[0.39400749063670411]	[0.41011235955056178]	[0.41423220973782771]	[0.42059925093632959]	[0.42846441947565544]
avg_score	0.373783	0.394007	0.410112	0.414232	0.420599	0.428464
feature_names	(11,)	(2, 11)	(2, 6, 11)	(1, 2, 6, 11)	(1, 2, 3, 6, 11)	(1, 2, 3, 6, 7, 11)

	7	8	9	10	11	12
feature_idx	(1, 2, 3, 6, 7, 11, 15)	(1, 2, 3, 6, 7, 10, 11, 15)	(1, 2, 3, 6, 7, 10, 11, 14, 15)	(1, 2, 3, 6, 7, 10, 11, 12, 14, 15)	(1, 2, 3, 6, 7, 10, 11, 12, 13, 14, 15)	(1, 2, 3, 6, 7, 8, 10, 11, 12, 13, 14, 15)
cv_scores	[0.43146067415730334]	[0.43820224719101125]	[0.43895131086142322]	[0.44868913857677903]	[0.46966292134831461]	[0.44644194756554306]
avg_score	0.431461	0.438202	0.438951	0.448689	0.469663	0.446442
feature_names	(1, 2, 3, 6, 7, 11, 15)	(1, 2, 3, 6, 7, 10, 11, 15)	(1, 2, 3, 6, 7, 10, 11, 14, 15)	(1, 2, 3, 6, 7, 10, 11, 12, 14, 15)	(1, 2, 3, 6, 7, 10, 11, 12, 13, 14, 15)	(1, 2, 3, 6, 7, 8, 10, 11, 12, 13, 14, 15)

	13	14	15	16	17	18
feature_idx	(1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)	(0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)	(0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...	(0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...	(0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14...	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...
cv_scores	[0.44269662921348313]	[0.4490636704119851]	[0.43595505617977526]	[0.44119850187265921]	[0.47790262172284644]	[0.45280898876404496]
avg_score	0.442697	0.449064	0.435955	0.441199	0.477903	0.452809
feature_names	(1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)	(0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)	(0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...	(0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...	(0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14...	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...

Figura 11 Búsqueda secuencial hacia adelante

	18
avg_score	0.463296
cv_scores	[0.4632958801498127]
feature_idx	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...
feature_names	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...

Figura 12 Búsqueda secuencial hacia atrás

Luego de analizar los resultados obtenidos de la búsqueda secuencial hacia adelante, se obtuvo que el mejor conjunto de características es el siguiente (1, 2, 3, 6, 7, 10, 11, 12, 13, 14, 15) por lo anterior este es el conjunto que se utiliza para evaluarlo con el modelo de redes neuronales, SVM y GBT.

Los resultados al evaluar los mejores modelos se pueden ver en las figuras 13, 14 y 15.

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.155383	0.0	0.031032

Figura 13 Evaluación con el modelo de redes neuronales

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.151292	0.0	0.017009

Figura 14 Evaluación con el modelo de SVM

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.151292	0.0	0.01744

Figura 15 Evaluación con el modelo GBT

C. Extracción de características

La extracción de características tienen como objetivo generar un nuevo subconjunto de características que contengan la mayor parte de información de las características originales, para este artículo se utiliza PCA (Análisis de componentes principales) la cual es una técnica que busca reducir la dimensión del conjunto de variables manteniendo el máximo nivel de variación de las variables. además resulta adecuado en vista de que al reducir la dimensión de las

variables, esta acción no afecta el rendimiento final del sistema.

Para elegir el número de componentes principales en el que se proyectarán los datos, se ha decidido utilizar el porcentaje de varianza acumulada, esto quiere decir que se seleccionan los primeros componentes que cubran el 79% de la varianza total, se estableció este criterio debido a que PCA organiza los componentes de forma descendente de acuerdo a los valores propios, por ende los componentes que estén en los últimos lugares van a tener poca influencia o presentan poca variabilidad de los datos.

	NUM_VAR	% VAR ACUM
0	1.0	0.267940
1	2.0	0.383177
2	3.0	0.468253
3	4.0	0.539354
4	5.0	0.599351
5	6.0	0.654980
6	7.0	0.708274
7	8.0	0.754808
8	9.0	0.796717
9	10.0	0.836685
10	11.0	0.875124
11	12.0	0.907932
12	13.0	0.936226
13	14.0	0.959549
14	15.0	0.978889
15	16.0	0.992888
16	17.0	0.999096
17	18.0	1.000000

Figura 16 Resultados de prueba de PCA

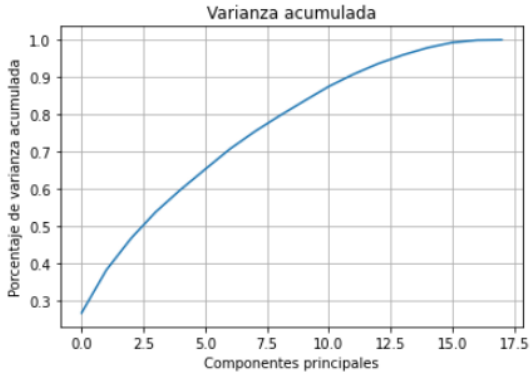

Figura 17 Curva de varianza

Como se puede observar en las figuras 16 y 17 si realizamos los experimentos con las 9 primeras características, podemos cubrir un 79% de la varianza total y de esta manera estamos disminuyendo en 9 la cantidad de características de nuestro sistema y por ende volviéndolo más eficiente, También se puede observar que si se aumenta el número de características a más de 9, el porcentaje de cobertura de la varianza total se ve afectado pero en una escala muy pequeña.

Los resultados al evaluar los mejores modelos se pueden ver en las figuras 18, 19 y 20.

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.141679	0.0	-0.016615

Figura 18 Evaluación con el modelo de redes neuronales

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.140542	0.0	-0.007091

Figura 19 Evaluación con el modelo de SVM

	Balanced Accuracy test	Std test	Matthews corrcoef test
0	0.140542	0.0	-0.006661

Figura 20 Evaluación con el modelo GBT

D. Discusión

Luego de evaluar los modelos GBT, SVM y redes neuronales con las características filtradas con por medio del algoritmo de selección hacia adelante y el PCA se llegó a la conclusión de que las características filtradas por medio del algoritmo de selección hacia adelante arrojan valores más acertados a la hora de implementar los modelos antes descritos pero aún así, si fueran comparados con los resultados antes de filtrar las características, se puede afirmar que el mejor resultado se encuentra cuando se ejecuta con todas las características.

Afrontando este problema se tuvieron varios inconvenientes, entre los cuales se puede resaltar el hecho de que al modelo le cuesta distinguir entre clases cercanas, ya que generalmente este tipo de jugadores pueden jugar entre ellos, por lo que tienen en general el mismo tipo de características como la dedicación de horas semanales, edad, etc.

Esto hace que el clasificador se confunda y tome las clases cercanas como si fueran la que se pretende encontrar.

Este problema va muy encaminado con lo que se menciona en el artículo de Avontuur et al [2] mencionado en el estado del arte, el cual plantea como una posible solución el agrupamiento de los niveles más cercanos, ya que según el mismo documento, es normal que el algoritmo se equivoque clasificando una entrada en una clase vecina, ya que son pocas las diferencias por ejemplo entre un jugador bronce y plata o entre uno oro y platino. Es por esto que agrupar clases para predecir por ejemplo si un jugador es novato o avanzado, se puede reducir el error y aumentar la precisión del algoritmo.

REFERENCIAS

[1] Thompson JJ, Blair MR, Chen L, Henrey AJ, 2013, Video Game Telemetry as a Critical Tool in the Study of Complex Skill Learning. PLOS ONE 8(9): e75129. Disponible en: https://doi.org/10.1371/journal.pone.0075129

[2] T. Avontuur, P. Spronck, and M. van Zaanen, "Player Skill Modeling in Starcraft II", AIIDE, vol. 9, no. 1, pp. 2-8, Jun. 2021. Disponible en: https://ojs.aaai.org/index.php/AIIDE/article/view/12682

[3] Bunker, R. and Thabtah, F., 2021. A machine learning framework for sport result prediction. [online] ScienceDirect. Disponible en: https://www.sciencedirect.com/science/article/pii/S2210832enviar doc y enlace al repo al correo717301485

[4] Horvat, T., Job, J. and Medved, V., 2018. Prediction of Euroleague Games based on Supervised Classification Algorithm k-Nearest Neighbours. [online] Scitepress.org. Disponible en: https://www.scitepress.org/Papers/2018/68935/68935.pdf

[5] Blair, M., Thompson, J., Henrey, A. and Chen, B., 2013. UCI Machine Learning Repository: SkillCraft1 Master Table Dataset Data Set. [online] Archive.ics.uci.edu. Disponible en: https://archive.ics.uci.edu/ml/datasets/SkillCraft1+enviar doc y enlace al repo al correoMaster+Table+Dataset