Implementación de una técnica de aprendizaje máquina sin el uso de un framework.

Imanol Muñiz Ramirez A01701713

Contents

[Introducción 1](#_Toc207026572)

[Contexto 1](#_Toc207026573)

[Problema 2](#_Toc207026574)

[Objetivo 2](#_Toc207026575)

[Solución 2](#_Toc207026576)

[Extracción de los datos 2](#_Toc207026577)

[Transformación 3](#_Toc207026578)

[Carga 3](#_Toc207026579)

[Resultados 3](#_Toc207026580)

[Conclusiones 3](#_Toc207026581)

# Introducción

## Contexto

Teamfight Tactics es un juego online de 8 jugadores que consiste en construir el equipo más fuerte para derrotar al de los demás. Cada ronda se simula el enfrentamiento y obtienes monedas de acuerdo con los resultados. Entre las rondas puedes comprar personajes llamados campeones y formar sinergias entre ellas. Existen campeones con costos que van desde una moneda hasta cinco, siendo generalmente los más costosos los que tienen un mayor potencial. Cada campeón además de contar con una habilidad única también cuenta con estadísticas base que van acordes a su costo y rol. Por ejemplo, un personaje diseñado para resistir el daño enemigo puede contar con más puntos de vida que el que esta diseñado para hacer daño, pero también entre campeones con mismo rol pero diferente costo, suele tener mayores atributos el que cuesta más.

## Problema

Riot Games frecuentemente está diseñando las próximas versiones de su juego Teamfight Tactics (TFT) cambiando campeones y mecánicas. Dado que cada campeón tiene atributos únicos, suele ser complicado ajustarlos para el costo en el que tiene que encajar o viceversa. A veces se lanza el nuevo set o versión con ciertos campeones que resultan injustamente más poderosos provocando desequilibrios que afectan la variedad de opciones y consecuentemente la jugabilidad.

## Objetivo

Desarrollar una herramienta de machine learning que nos permita asignar el costo de un campeón con base en sus estadísticas base.

# Solución

El archivo MachineLearning.py en esta misma carpeta contiene el código que obtiene los parámetros ““ de una función para calcular el costo de cada campeón según sus estadísticas base. Esta función la obtenemos por el método de regresión lineal, utilizando como modelo la función dónde representa el costo del campeón y las cada una de las estadísticas base de este. Para calcular el error utilizamos el promedio del error al cuadrado (MSE) entre los datos reales y la predicción del modelo. Para conocer la dirección en que deben ajustarse los parámetros tras cada época usamos el algoritmo de gradiente descendiente.

El entrenamiento y las pruebas se hacen por medio de validación cruzada tomando subconjuntos de “x” instancias para minimizar el sesgo que podría ocasionar entrenar y probar con el mismo conjunto de datos. De esta forma evitamos caer en que el modelo solo haya “memorizado” y lo confundamos con que funciona a la perfección.

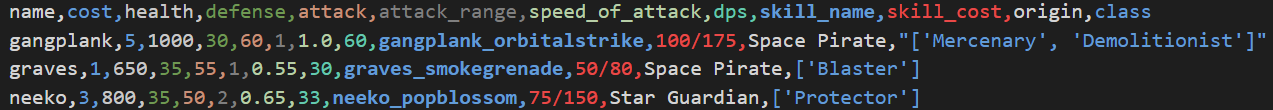
## Extracción de los datos

Los datos se descargaron de Kaggle a través de la siguiente URL, compartidos por un miembro de la comunidad.

<https://www.kaggle.com/datasets/gyejr95/league-of-legends-tftteamfight-tacticschampion?select=TFT_Champion_CurrentVersion.csv>

Los datos son tabulares. Las columnas son principalmente numéricas, aunque también hay textos. Son 52 instancias, una por cada campeón.

Columnas con ejemplos:



## Transformación

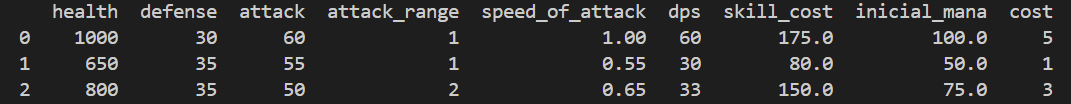
Primero fue necesario eliminar las columnas que no eran relevantes o el tipo de dato no era manejable para este algoritmo. Eliminamos name, class, origin y skill\_name.

En el caso de la columna skill\_cost observamos que el tipo de dato es un string que contiene dos enteros separados por una diagonal. Esta columna representa con cuanta energía o maná inicia el combate y cuánta necesita el personaje para ejecutar su habilidad. Un campeón que requiera una cantidad muy alta de maná para lanzar una habilidad o que inicie el combate con muy poca cantidad, podría repercutir en qué tan poderosa es esa unidad. Por lo tanto, esta columna nos es relevante para nuestro objetivo. A partir de esta creamos dos columnas llamadas inicial\_mana y skill\_cost.

Revisando las instancias observamos que hay ciertos campeones que no utilizan maná para lanzar su habilidad. Por lo tanto, para estas unidades este atributo lo convertiremos a cero para que de esta forma solo repercuta el resto de sus estadísticas.



Por otra parte, nuestra columna objetivo (cost) la ponemos al final de la tabla para manejarla más fácilmente. Nos queda algo así:



Si observamos el conjunto de datos podemos observar un último problema. Existen atributos que pueden ser 1000 y otros que van entre 0 y 1. Al trabajar con valores grandes y potencias ocasiona desbordamientos de variables y que sea más complicado encontrar los hiper parámetros adecuados resultando en que el modelo no converja. Para solucionar este problema aplicamos un escalamiento a todas las variables para convertirlas en números entre 0 y 1.

## Carga

Los datos utilizados se encuentran disponibles en el repositorio en el archivo TFT\_Champion\_CurrentVersion.csv: <https://github.com/ViejoAgrio/Machine-Learning>

## Resultados

A screen shot of a computer program

AI-generated content may be incorrect.

Promedio de error: 0.5818

Entrenamiento del bloque 0 con epochs igual 1000 y learnign rate igual a 0.1.

A blue line graph with numbers

AI-generated content may be incorrect.

Matriz de confusión.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Clase | 1 | 2 | 3 | 4 | 5 |
| 1 | 2 | 2 | 0 | 0 | 0 |
| 2 | 9 | 8 | 2 | 1 | 0 |
| 3 | 1 | 2 | 9 | 4 | 0 |
| 4 | 0 | 0 | 1 | 4 | 3 |
| 5 | 0 | 0 | 0 | 0 | 3 |
| 6 | 0 | 0 | 0 | 0 | 1 |
| Total | 12 | 12 | 12 | 9 | 7 |

# Conclusiones

Pendientes: matriz de confusión automática

Transformación en otro archivo

Plot de cada una de las x contra y.