TECNOLOGIA DE DATOS MASIVOS

Nombre: Joel Pablo Pérez Camacho

Docente: Dr. Juan Gabriel Gomila

Materia: Tecnología de datos masivos

Fecha: 22-Enero-2020

INTRODUCCION

El presente trabajo plantea demostrar la manipulación y uso de los datos, para esto se realizará el manejo del mismo, en Rstudio, no obstante, se realizara predicciones en base a algunos modelos aprendidos como Random Forest, Decision Trees y Boosting, posteriormente se contrastara dichos modelos mediante Cross-Validation.

Datos

La base de datos proviene de Kaggle (https://www.kaggle.com/devinanzelmo/dota-2-matches#players.csv), de un videojuego llamado “Dota 2” , donde se recopila alrededor de 1 millón de datos que explica el comportamiento de cada jugador en relación al juego.

Existen 19 datasets, los cuales explican la habilidad del jugador, la region de la cual provienen, las habilidades de los heroes, los heroes, lista de items, lista de jugadores, pelea campal (team figth), objetivos, partidas, y una serie de datos caracteristicas distintas, en el que nos centraremos solo en alguna de ellas.

PLANTEAMIENTO DEL PROBLEMA

Para realizar un correcto planteamiento preguntamos:

¿Predecir si un jugador ganara? Es importante considerar cuando se hace la pregunta. Con mayor frecuencia, esto se hace al momento de que un partido comience, cada jugador tiene un roll especifico (suport, off, mid, hc, roaming), para predecir si un jugador ganara, se puede tomar en cuenta, el oro que gana, las muertes que tiene el jugador, los asesinatos que realiza, la experiencia del héroe, el daño a los héroes contrincantes, la vida recuperada, las asistencias en equipo al momento de realizar una campal, el nivel del héroe en relación al roll, cada parámetro tiene sus puntos fuertes y débiles, debido a que la base de datos

no especifica el roll de cada jugador, tampoco cuenta con el tipo de partida (individual, grupal, clasificatoria) , a su vez no cuenta con el MMR (Nivel de cada jugador).

OBJETIVO

Manipular los datos para construir un modelo que prediga si un jugador gana o pierde en base a las distintas variables establecidas, entrenando dicho modelo hasta obtener el menor error posible.

Visualización de datos

Primeramente se realiza la manipulación de los datos, eliminando variables innecesarias, omitiendo N/A, renombrando variables, agrupar datos, filtrar y finalmente visualizar los datos viendo la relación de los mismos.

Para esto nos planteamos supuestos muy genuinos:

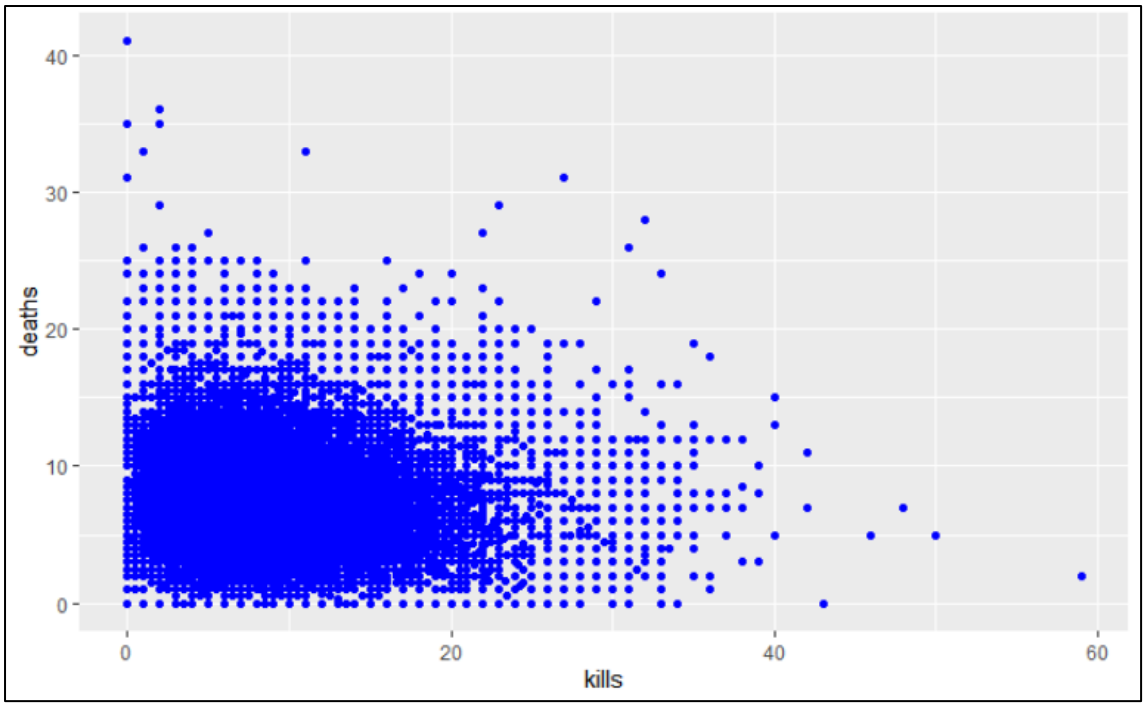
*Si un jugador asesino demasiado y muere poco tiene tendencia a ganar.

*Si un jugador gana mucho oro significa que farmea demasiado, esto conlleva a ganar mas nivel, y subsecuentemente sacar mas item y ventaja en una partida esto conlleva a llevarse la victoria.

*Si un jugador tiene muchas asistencias y mucho oro, significa que gano las campales, esto conlleva a tener el win.

Solo nos basaremos en estos supuestos, sin duda existir muchas combinaciones (basicamente una permuta de 14 variables), no obstante despues veremos modelos que priorizan variables en base a los datos establecidos.

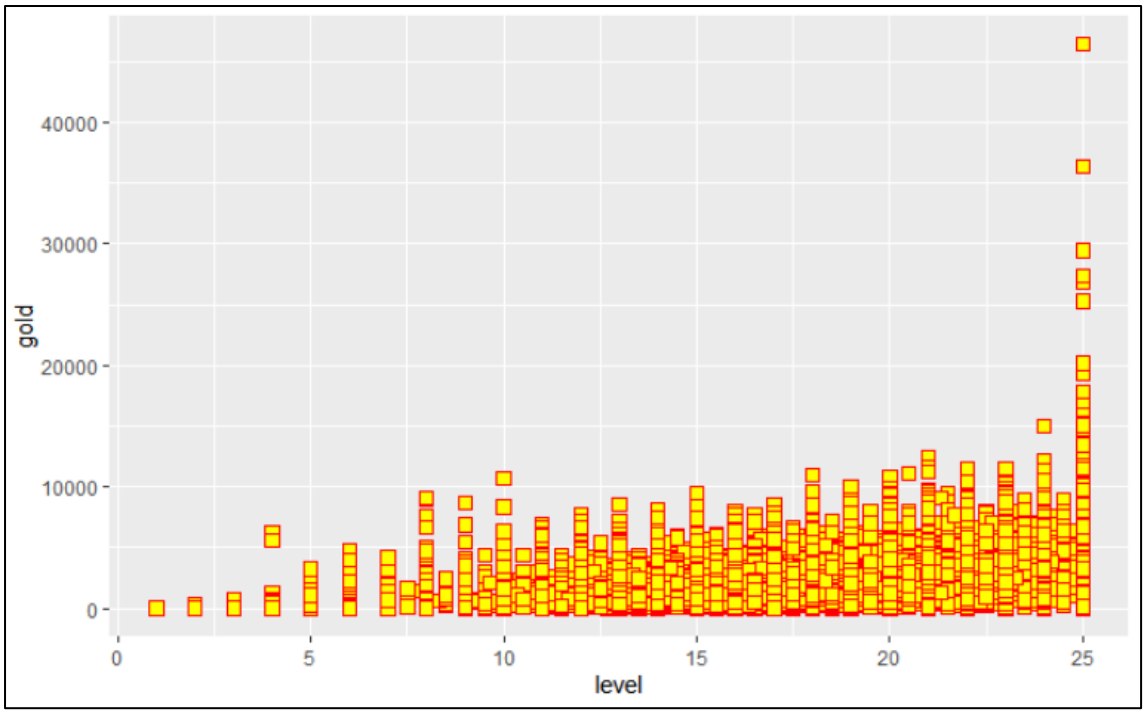
GRAFICO KILLS CONTRASTADO CON DEATHS.



Fuente: elaboración propia en base a datos dota2.

Esta relación hipotética nos plantea que generalmente los jugadores tienden a tener un intervalo de muertes de 5 a 15 y un valor aproximado de 0 a 17 asesinatos, es decir asesinan mas de lo que mueren, para aclarar el grafico se podría plantear la perdida de oro en base una de estos variables.

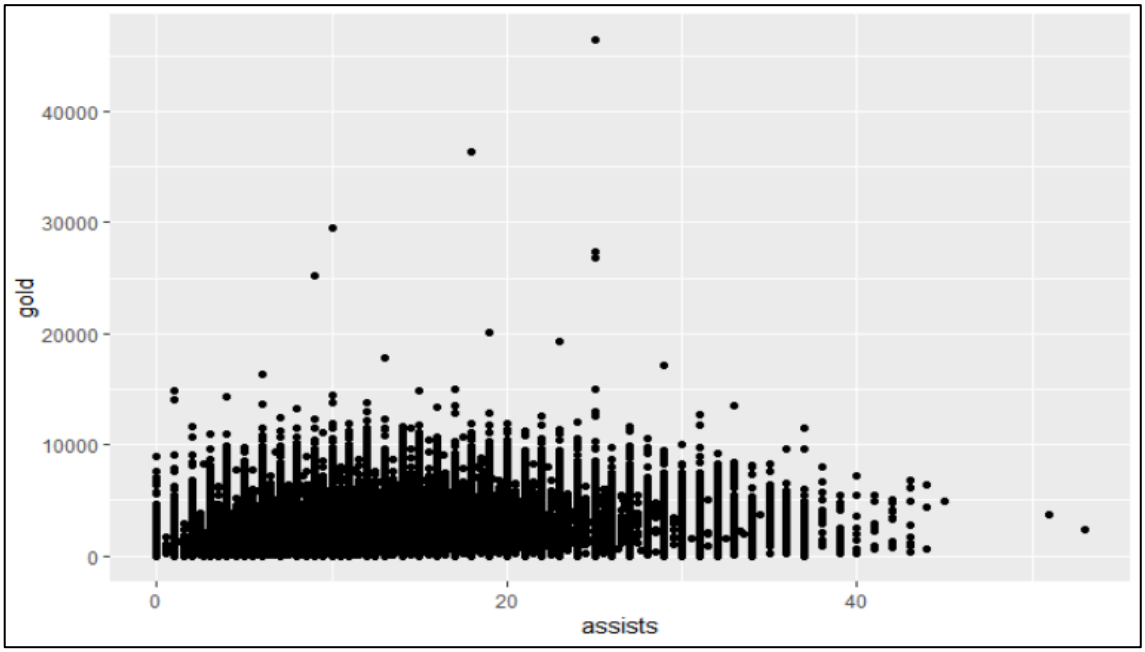
GRAFICO GOLD CONTRASTADO CON LEVEL.



FUENTE: Elaboración propia en base a datos dota 2.

Si un jugador tiene elevado nivel, quiere decir que gano mucho oro durante la partida, aunque los supuestos planteados son superficiales, eso no es una afirmación concreta, solo es visual

GRAFICO Gold contrastado con “assists”



FUENTE: Elaboración propia en base a datos de dota2.

Según el supuesto 3 (planteado anteriormente), afirmamos que estamos equivocados, el grafico señala que un jugador tiene entre 1 y 25 asistencias gana aproximadamente 10000 de oro, esto quiere decir que no necesariamente un jugador debe tener asistencias para ganar oro. Entonces esto no conlleva necesariamente a la victoria.

Plantear supuestos al azar de si un jugador gana o no gana en base a unos gráficos no confirma nada, para esto se construyen modelos de predicción un poco mas complejos, de los cuales utilizaremos decisión trees, random forest, boosting y finalmente los cruzaremos con croos validation.

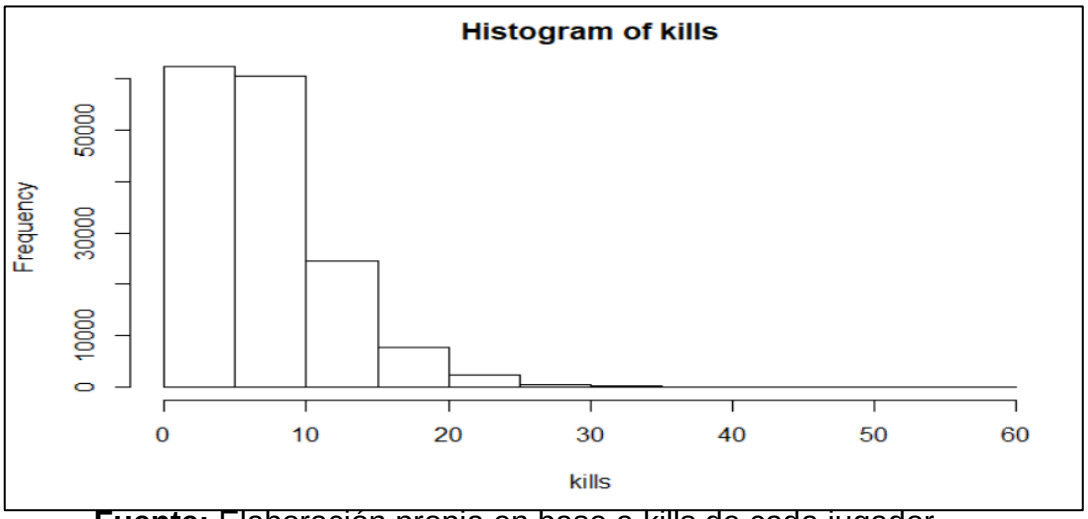
ARBOL DE DECISION (DECISION TREES)

El arbol de decisión se plantea en base a 11 variables, de las cuales supondremos un escenario hipotético:

- Si el jugador tiene muchos kills (asesinatos), conlleva a tener mayor oro, experiencia (la experiencia va ligada al nivel del héroe), asistencias de equipo, lo cual genera mayor probabilidad de ganar un partido.

Primero analizaremos cada variable por separada “kills” y “deaths”.

FRECUENCIA DE KILLS SEGÚN JUGADORES



Fuente: Elaboración propia en base a kills de cada jugador.

El anterior histograma muestra la frecuencia de kills por cada jugador, que dice de 0 a 10 se encuentra el 80% de los datos, es decir una gran mayoría de jugadores realiza asesinatos (kills), en dicho rango de valores y el restante esta entre 10 a 35.

Se crea una variable denominada High, como respuesta binaria, el cual clasifica si el jugador tiene muchos o pocos kills, en base a la media de la misma variable (kills.mean()=7).

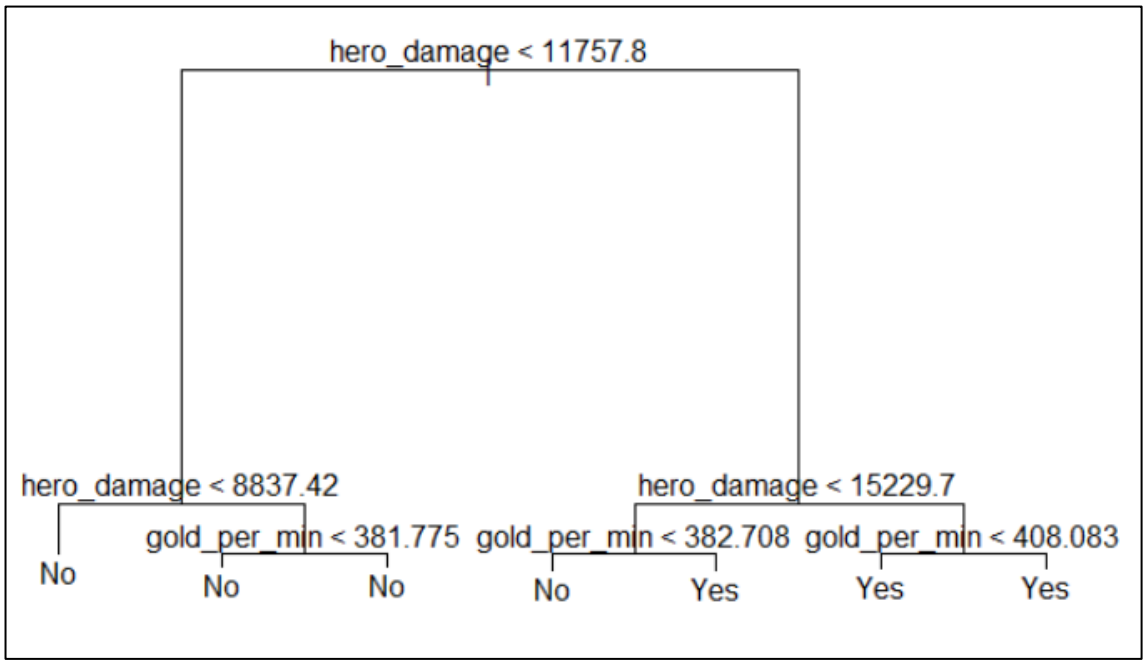
Datos de la construcción del árbol de decisión

```
Classification tree:
tree(formula = High ~ . - kills, data = datos_ajustados)
Variables actually used in tree construction:
[1] "hero_damage"  "gold_per_min"
Number of terminal nodes:  7
Residual mean deviance:  0.6779 = 107100 / 158000
Misclassification error rate: 0.1441 = 22767 / 158015
```

FUENTE: Elaboración propia en base a datos player.csv

El anterior cuadro señala que el árbol tiene un total de 7 nodos terminales, y que se emplearon como predictores las variables “hero_damage” ; “gold_per_min”, el término Residual mean deviance es simplemente la suma de cuadrados residuales dividida entre (número de observaciones - número de nodos terminales), el cual genera un valor de 0,67 (mientras menor sea mejor es el ajuste a las observaciones de entrenamiento), a su vez tenemos un error de clasificación de 0,14 el cual argumenta ser bueno (a menor error mayor certeza de predicción).

Árbol de decisión (Decisión trees)



FUENTE: Elaboración propia en base a players.

El árbol de decisión tomo en cuenta como primer y segundo nodo la variable “hero_damage” el cual mide el daño realizado a héroes enemigos, señalando que es un predictor muy importante. Por ejemplo el árbol predijo que si el daño a héroe (hero_damage) es mayor a 11757.8 y estos a su vez tienen un daño mayor a 15229.7, obtendrán 408.083 de oro por minuto, lo cual significaría que tendrían mayor proporción de kills (sin tomar en cuenta oro de crep).

Árbol de decisión (detallado)

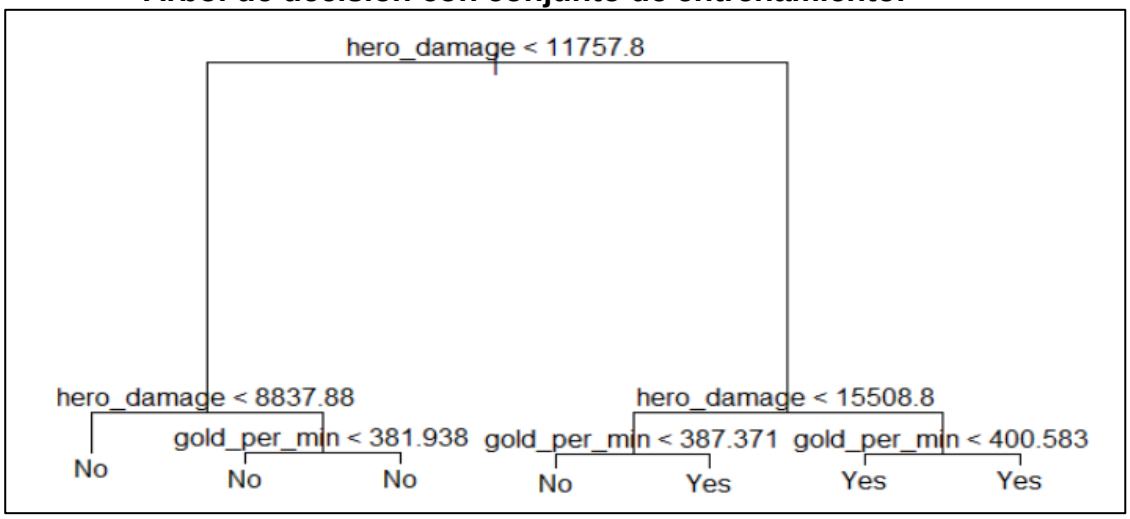
```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 158015 215500 No ( 0.57443 0.42557 )
   2) hero_damage < 11757.8 84804  58890 No ( 0.88966 0.11034 )
     4) hero_damage < 8837.42 50427  11920 No ( 0.97464 0.02536 ) *
     5) hero_damage > 8837.42 34377  37490 No ( 0.76502 0.23498 )
      10) gold_per_min < 381.775 17298  13170 No ( 0.87299 0.12701 ) *
      11) gold_per_min > 381.775 17079  21990 No ( 0.65566 0.34434 ) *
   3) hero_damage > 11757.8 73211  75120 Yes ( 0.20929 0.79071 )
     6) hero_damage < 15229.7 32334  42670 Yes ( 0.37162 0.62838 )
      12) gold_per_min < 382.708 8900  11920 No ( 0.60742 0.39258 ) *
      13) gold_per_min > 382.708 23434  27880 Yes ( 0.28207 0.71793 ) *
     7) hero_damage > 15229.7 40877  22970 Yes ( 0.08088 0.91912 )
      14) gold_per_min < 408.083 6345   7443 Yes ( 0.27329 0.72671 ) *
      15) gold_per_min > 408.083 34532  12780 Yes ( 0.04552 0.95448 ) *
```

FUENTE: Elaboración propia en base a datos player.csv.

El anterior cuadro muestra el criterio de división de cada nodo, el número de observaciones que hay en esa rama (antes de dividirse), la deviance, la predicción promedio de esa rama (en el caso de clasificación se muestra el grupo más frecuente) y la proporción de cada grupo. Cuando se trata de nodo terminal, se indica con un asterisco.

Ahora creemos una división del conjunto de entrenamiento y prueba (110.610,47.405) de las 158.015 observaciones, haga crecer el árbol en el conjunto de entrenamiento y evalúe su desempeño en el conjunto de prueba.

Árbol de decisión con conjunto de entrenamiento.



FUENTE: Elaboración propia en base a conjunto de entrenamiento.

Podemos observar que el árbol no tiene cambios radicales en comparación al original, los valores de las ramas tienen un cambio numérico débil.

Predicción

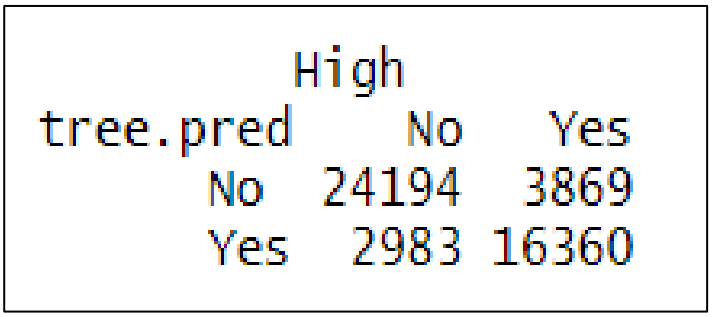
	High	
tree.pred	No	Yes
No	24431	4081
Yes	2779	16114

FUENTE: Elaboración propia en base a A.D. entrenado

Para realizar una síntesis de la certeza en la predicción realizamos una pequeña operación (24431+16114)/(24431+4081+2779+16114)= 0.8552 es equivalente a 85.85% de certeza en la predicción

Con la finalidad de reducir la varianza y minimizar el error se somete al proceso pruning o también conocido como “podado”

Predicción con pruning



	High	
tree.pred	No	Yes
No	24194	3869
Yes	2983	16360

FUENTE: Elaboración propia en base a pruning.

Los resultados de pruning es relativo al árbol original, esto significa que el error no varía, ya que nos dio los mismos valores, esto solo puede significar dos cosas, que las variables a explicar contienen información inadecuada y el modelo se ajusta perfectamente a la predicción o que las variables toman formas empíricas al momento de realizar pruning (es mientras mas grande mejor el podado).

RANDOM FOREST AND BOOSTING

Para trabajar con este modelo se utilizará 1000 muestras aleatorias (se tomo el valor de 1000 en base a la capacidad que soporta el ordenador, memoria RAM).

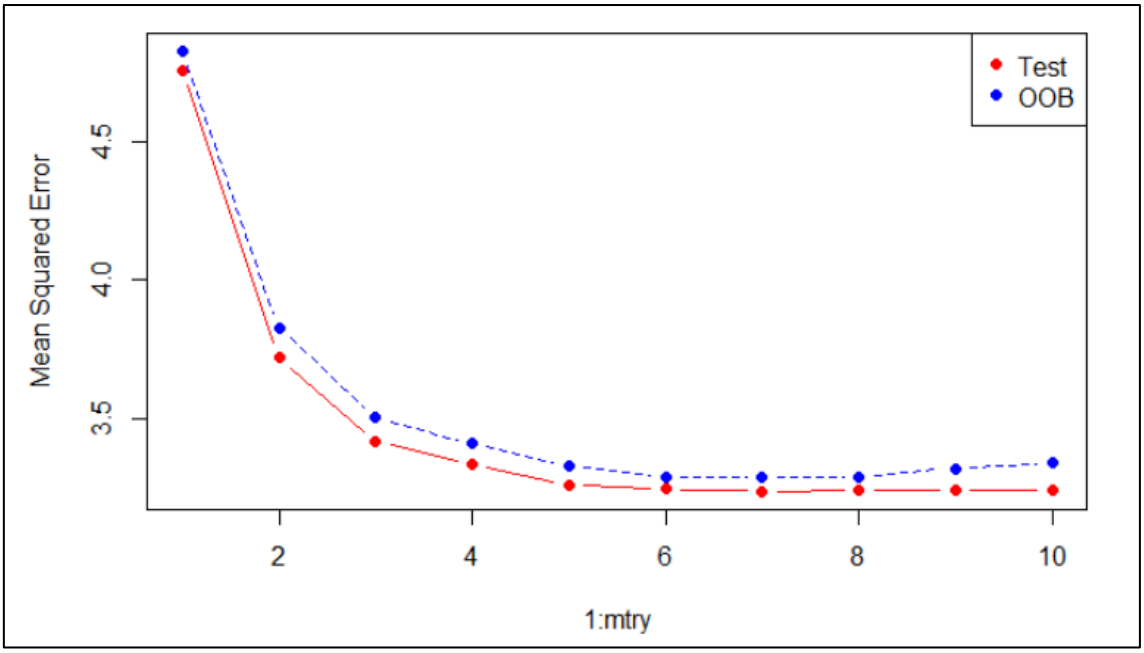
Random Forest

```
Call:
 randomForest(formula = kills ~ ., data = datos_ajustados, subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 3.397798
                    % Var explained: 85.48
```

FUENTE: Elaboración propia en base a datos players.csv.

El numero de arboles generados es de 500, tiene 4 splits (cortes o ramificaciones), el error es del 3.39% con una explicación del 85% en relación con la varianza.

Out of Bag



FUENTE: Elaboración propia en base al modelo out of bag.

En el grafico anterior podemos señalar que la curva de error de prueba cae por debajo de la curva de OOB, el cual indica que estas estimaciones están basadas en sus mismos datos Y por lo tanto tienen sus propios errores. La curva “mtry” se comienzan a estandarizar en 3.

BOOSTING

Boosting es otra estrategia de ensemble que se puede emplear con un amplio grupo de métodos de statistical learning, entre ellos los árboles de predicción. La idea detrás de boosting es ajustar, de forma secuencial, múltiples weak learners (modelos sencillos que predicen solo ligeramente mejor que lo esperado por azar). Cada nuevo modelo emplea información del modelo anterior para aprender de sus errores, mejorando iteración a iteración.

Aplicando Boosting al modelo tenemos los siguientes resultados:

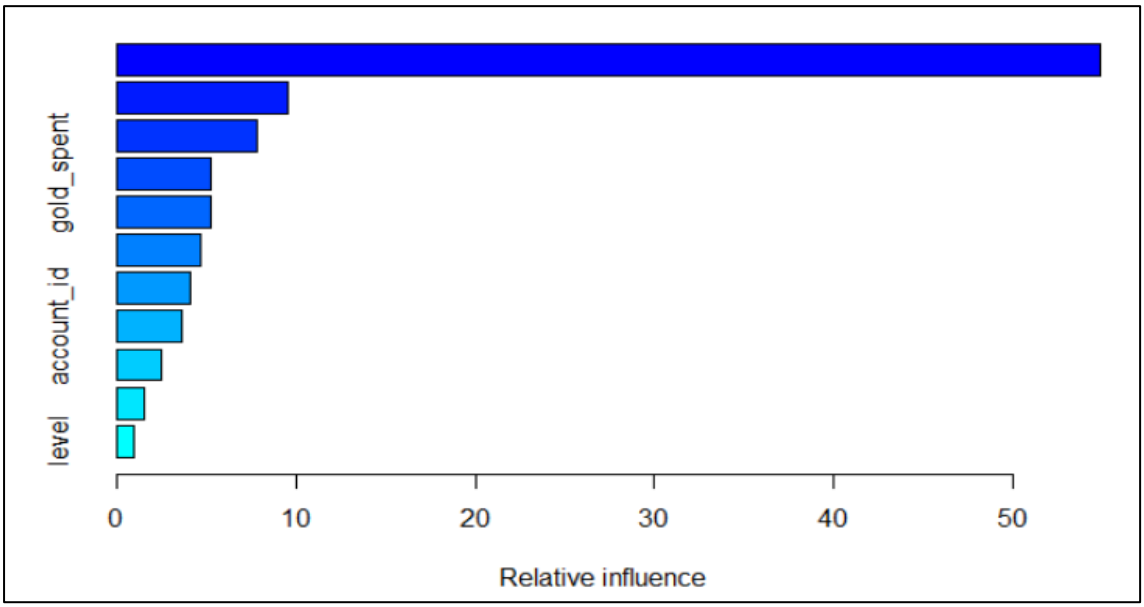
Análisis de variables

	var <fctr>	rel.inf <dbl>
hero_damage	hero_damage	54.8856701
xp_hero	xp_hero	9.5339537
gold_per_min	gold_per_min	7.8110579
gold_spent	gold_spent	5.2585992
xp_per_min	xp_per_min	5.2191251
last_hits	last_hits	4.6423131
assists	assists	4.0856282
account_id	account_id	3.6639554
deaths	deaths	2.4768894
denies	denies	1.5042998

FUENTE: Elaboración propia en base a players.csv.

Contrastando con el modelo anterior (las variables que explicaban más eran hero_damage y gold_per_min), este modelo señala que hero_damage (54.88%) y xp_hero (9.5%), aportan más información al momento de predecir.

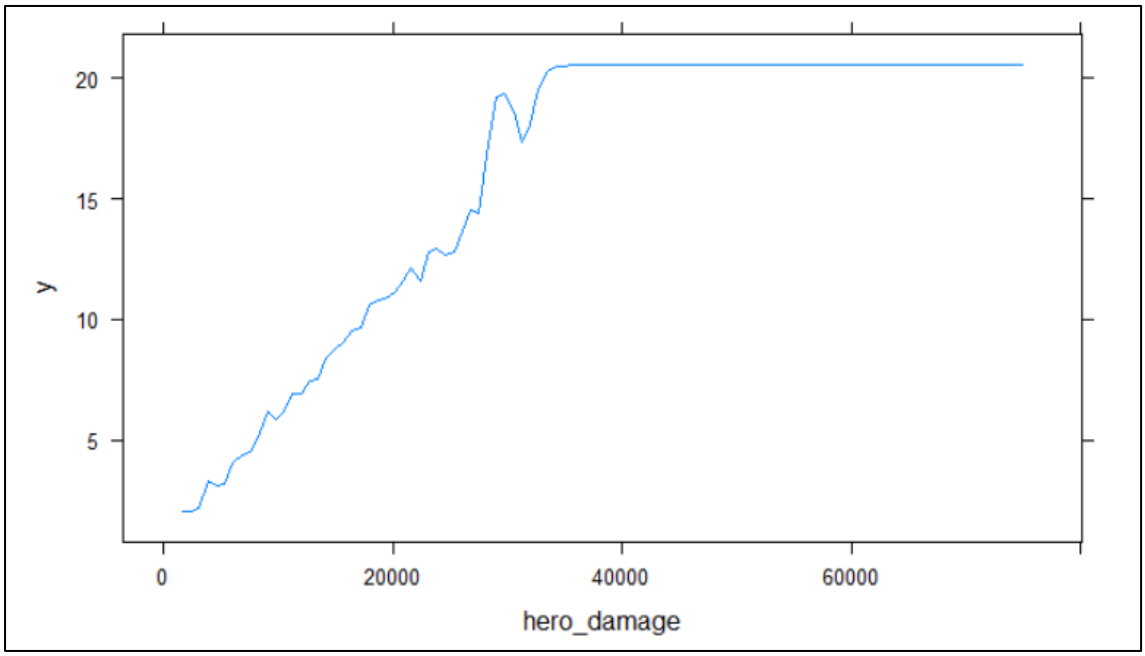
Análisis de influencia en relación a las variables



FUENTE: Elaboración propia en base a datos.

La influencia relativa de las variables se encuentra entre gold_spent que es el oro perdido ya sea por muertes al enemigo y account_id el cual no señala nada más que la dirección del jugador IP (es decir IP de jugador).

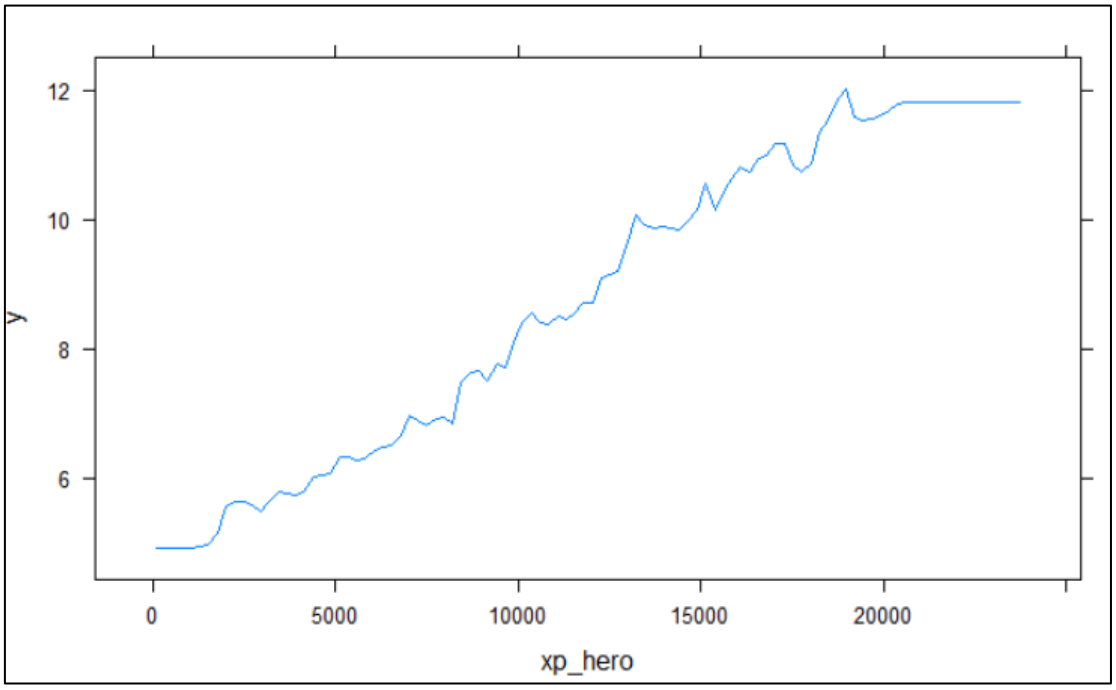
Grafico de hero_damage



FUENTE: Elaboracion propia en base a la variable hero_damage

El grafico anterior muestra la tendencia del daño a los héroes enemigos, el cual va desde 0 hasta 60.000, quiere decir que a mayor daño realizado, mayor la probabilidad de realizar kills, la tendencia aumenta hasta 20 kills, sin considerar el roll del jugador, la probabilidad de matar o no matar al final no dependerá del mismo (ya que la variable dependiente es killS en el eje Y).

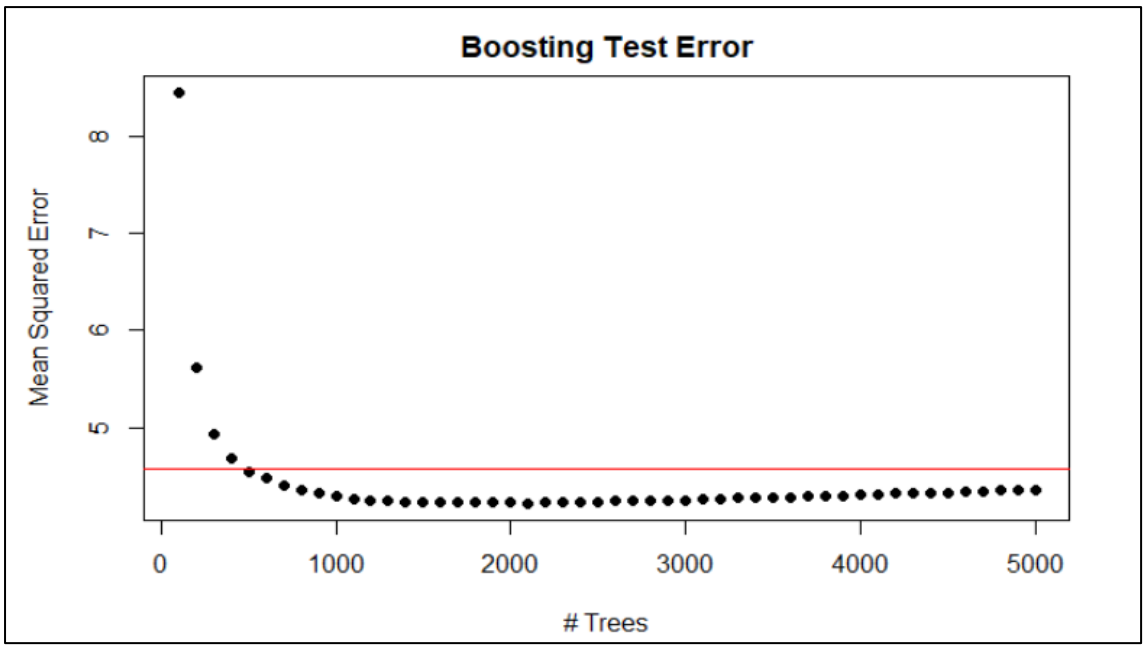
GRAFICO DE XP_HERO



FUENTE: Elaboración propia en base a datos.

La segunda variable explicativa, señala que mientras mayor kills consiga el jugador mayor será la experiencia del héroe. O viceversa la clara tendencia de matar genera experiencia al héroe (ya sea en campal o muerte offlaner).

Comparación con el modelo Random Forest

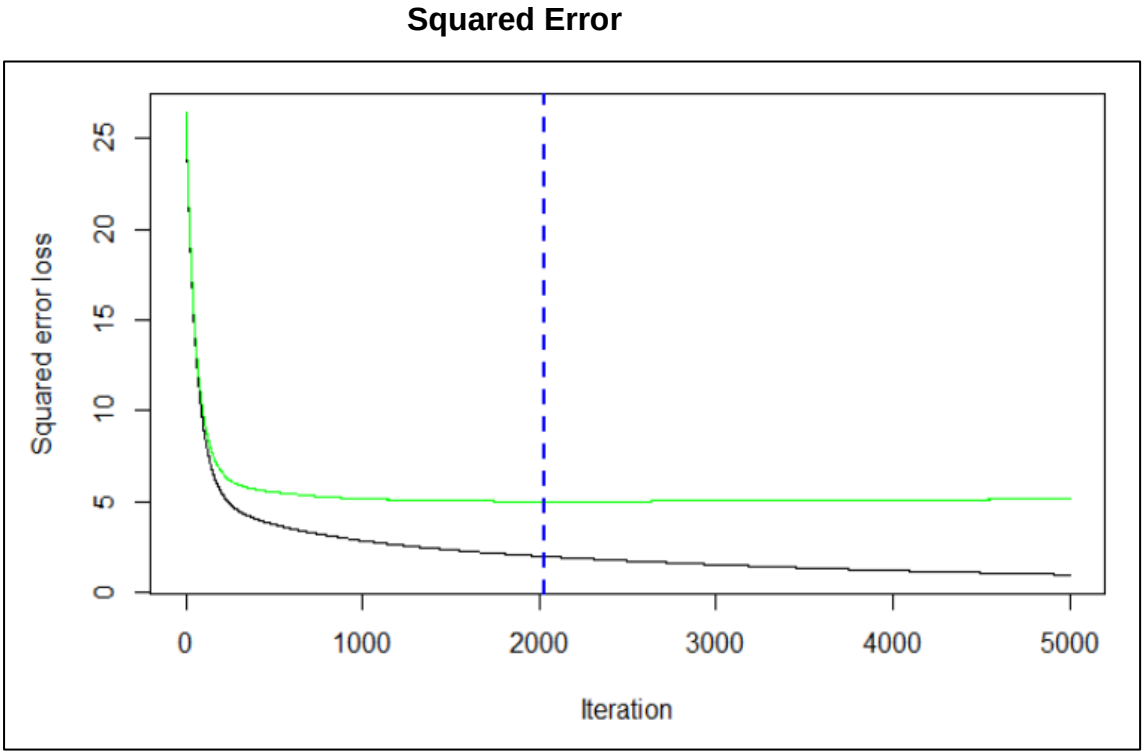


FUENTE: Elaboracion propia en base a datos preliminares.

Según el test error observamos que Boosting tiene menor error en comparación con Random forest (la línea roja denota el error de forest), esto es debido a que el algoritmo aprendió demasiado, tanto que puede conllevar al OVERFITING, que es el sobreajuste, esta es una desventaja de ajustar demasiado el modelo, para que esto no ocurra se establece los ntrees y ramas a las cuales puede llegar el modelo. El modelo es bueno tiene un error considerable al momento de predecir.

CROSS VALIDATION

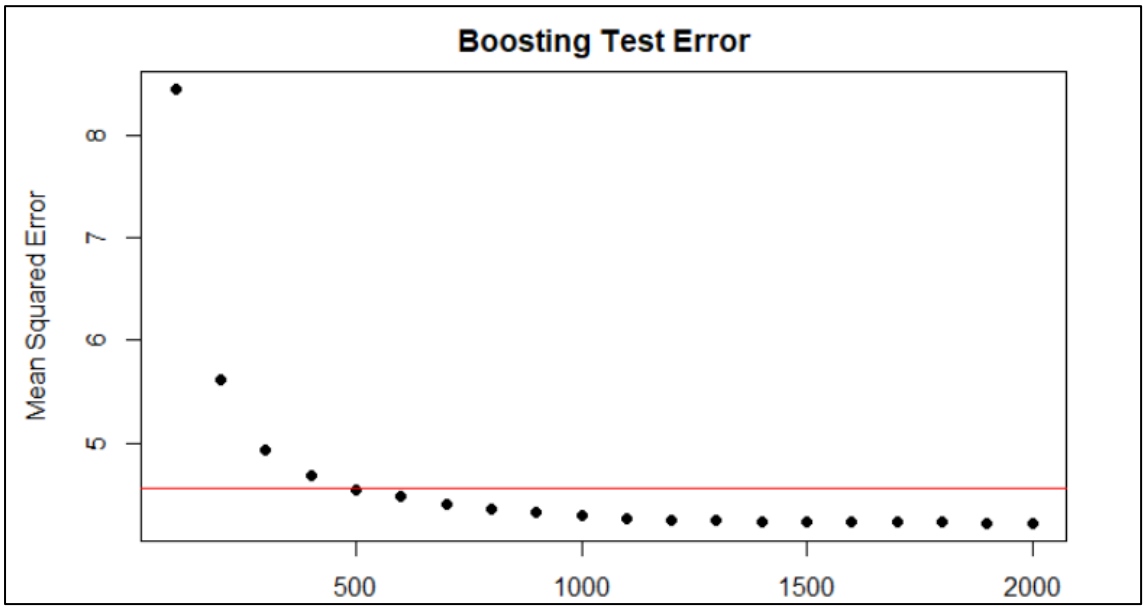
Una vez analizados los modelos, procedemos a realizar el análisis de validación cruzada de los distintos modelos (los parámetros se encuentran en el código R).



FUENTE: Elaboración propia en base a modelos preliminares.

El grafico anterior muestra que de las 5000 iteraciones solo se requieren 2000 es decir, señala la relación del error y lo que el modelo aprende según se acumulen los datos de entrenamiento o las ramas.

Validacion Cruzada (cruce de modelos)



FUENTE: Elaboración propia en base a validación cruzada.

Finalmente confirmamos que Boosting se ajusta mejor a los datos en relación a Random Forest, no obstante, la diferencia de erro es mínima en comparación a Decision Trees que en comparación con estos dos modelos tiene una diferencia de error del 11% (14$ error decision tres- 3% boosting y Random), afirmando que entrena mejor, se ajusta bastante bien a los datos la desventaja es que si añadimos nuevos agentes a este modelo, puede producir OVERFITING ya que el modelo se ajusto lo suficiente.

CONCLUSIONES

Los datos se manipulan en base a las expectativas creadas (preguntas), es decir que los datos se manejan de acuerdo con un objetivo general, y en relación a esto realizar mutates, arrages, filters, groupbys y una diversidad de funciones que puedan utilizar. Para el presente caso se realizo la manipulación en general. Sin interiorizar en las funciones más complejas.

El modelo de Decision Trees (arboles de decision), predice con un error del 14.41%, toma en cuenta las variables damage_hero y gold_per_min como variables explicativas. No consigue mejorar sus estimaciones con pruning, ya

que no mejora la predicción. No Obstante el modelo no es malo, la predicción indica que realizando mayor daño a héroes y ganando un significativo nivel de oro (gold) existe mayor probabilidad de ganar.

El modelo Random Forest explica el 85.5% de la varianza de los datos, tiene un error del 3.39%, se ajusta bien al modelo. Se toma como variable independiente (kills) debido a que suponemos que mientras mas muertes se realiza en una partida existe mayor probabilidad de ganar, random forest kills es una variable significativa en relacion a decisión trees.

Boosting toma como variables hero damage que explica el 54.88% y hero_exp que explica el 9.5%, tiene un menor error a comparación de Random forest, aproximando un 2.8% (dato obtenido del grafico).

Según Cross Validation, afirmamos que el modelo mejor ajustado es de Boosting, sin menos preciar Random Forest, la diferencia de errores es mínima.

El análisis de los datos esta en base al criterio del investigador, las predicciones son muy análogas, ya que para predecir correctamente, se deben tomar mas variables las cuales no se tomaron en cuenta como el MMR de cada jugador o las horas jugadas, la experiencia del jugador en base al juego, la clasificación de cada jugador (player).