Escuela Técnica Superior de Ingeniería Electrónica e Informática la Salle

Trabajo de Final de Máster

Máster Universitario en Ciencia de los Datos

MEJORA DE LA PUREZA DE SELECCIÓN EN COLISIONES SUBATÓMICAS

Alumno

Angel Berian Gonzalez

Profesor Ponente

Miriam Calvo Gómez

ACTA DE PRESENTACIÓN DEL TRABAJO DE FINAL DE MÁSTER

Reunido el Tribunal calificador el día indicado en la fecha, el alumno

D. Ángel Berián González

Expone su Trabajo de Final de Máster con la temática siguiente:

MEJORA DE LA PUREZA DE SELECCIÓN EN COLISIONES SUBATÓMICAS

Terminada la exposición y contestadas por parte del alumno las objeciones formuladas por los Señores y Señoras miembros del tribunal, estos valoran dicho Trabajo con una calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

A mis padres, Carlos y María Asunción, por enseñarme lo importante que es la educación pero sobre todo por enseñarme a ser feliz.

A mis hermanos, Carlos y Juan, quienes en la distancia me enseñan el valor de apreciar las cosas y de seguir proyectos

A mi pareja, Núria, quién conoce más que nadie el valor del sacrificio y del desarrollo personal, además de haberlo dado todo siempre sin pedir nada a cambio

A mis amigos, a todos y cada uno de ellos, por los momentos que hemos perdido en el transcurso de este máster, estos serán recuperados.

A Richard Feynmann, icono y modelo, por enseñarme la importancia de la pasión, la dedicación, la naturalidad y las ganas de transmitir conocimiento.

A Quentin, Manager de equipo en Teletech, por darme la oportunidad de adentrarme en el mundo digital careciendo de experiencia relevante.

A Heather, directora de operaciones en Teletech, por enseñarme un camino profesional basado en la analítica y hasta entonces para mí, desconocido.

A Solene, Recruiter en Criteo, por encontrarme en la maraña digital de Linkedin y abrirme las puertas a Criteo

A Giacomo, manager de ventas en Criteo, por darme la oportunidad del cambio y darme la confianza para afrontar todos los proyectos del mundo

Al equipo de ventas de España en IBM y Criteo, por enseñarme el lado humano del mundo empresarial

A Livio, Analista de Datos en Criteo, por ayudarme a poner en marcha el proceso de cambio profesional

A Eduard, Senior Manager de Analistas en Criteo, por servirme como motivación al querer demostrarte que estabas confundido conmigo

A Vanessa, eternamente, Manager de Analistas en Criteo, por darme una oportunidad que en el pasado me había sido negada

A la universidad La Salle, por aceptarme el último día de admisiones cuando ISDI canceló el programa que iba a estudiar con ellos

A Xavier y Elisabet, Director y Coordinadora del máster, por organizar este programa y por vuestra disposición en los momentos de necesidad.

A mis compañeros del máster, por las innumerables veces que me ayudasteis

A Miriam, Tutora de trabajo, por tu iniciativa, disponibilidad y seguimiento del trabajo

Finalmente, Al equipo de AX de Criteo, porque el futuro es nuestro

RESUMEN

El objetivo del presente trabajo es estudiar y comparar diferentes herramientas existentes para la selección de los datos relevantes en los estudios de desintegraciones en LHCb (CERN), concretamente la desintegración del Mesón B^0 en un mesón K^{*0} y en un fotón γ que, a su vez, el mesón K^{*0} se reconstruye a partir de un pion y un kaon, $K^+\pi^-$ o $K^-\pi^+$.

La desintegración descrita del mesón B^0 se genera con poca probabilidad, produciéndose otro tipo de desintegraciones más frecuentemente, las cuáles generan un ruido combinatorio que dificulta la separación entre mediciones correctas y mediciones de fondo. Surge por lo tanto la necesidad de encontrar algoritmos que separen correctamente la señal obtenida de lo clasificado como ruido combinatorio.

Para ello se estudian las características numéricas producidas en la desintegración, así como las relaciones físicas entre las variables. Con este conocimiento adquirido se generan modelos de inteligencia artificial basados en la tecnología de *Boosted Trees*, los cuales entrenaremos y probaremos con datos reales y artificiales generados con métodos de *Monte Carlo.* Finalmente se mide el rendimiento de los modelos con el objetivo de encontrar el algoritmo que optimice los resultados manteniendo una fiabilidad estadística que le permita ser replicable en casos similares.

SUMMARY

The objective of this thesis is to study and compare the different tools available for Data Selection in decay studies elaborated in the LHCb experiment (CERN), specifically the decay of the B meson into a K^{*0} meson and a photon γ, where the K^{*0} meson decays into a pion and a kaon, $K^+\pi^-$ or $K^-\pi^+$.

This specific decay occurs with a low probability. In a proton-proton collision, thousands of particles are produced, and other kind of B decays occur more frequently, thus resulting on a large combinatorial background in the selection of our decay of interest. It arises the need of finding specific algorithms that can distinguish between the desired signal from the background (noise).

In order to do so, we will study the numerical characteristics of the selected B candidates and its decay products, as well as the physical relationship between the variables. With this knowledge acquired we generate artificial intelligence models based on *Boosted trees* technology, where we will train and test them with both real data and data generated artificially by *Monte Carlo* models. Then, we will measure each model´s performance with the objective of maximizing the results while ensuring that those are reliable and can be applied in similar cases.

ÍNDICE

1. ANTECEDENTES Y TRABAJO REALIZADO

1. 1 Introducción al Modelo Estándar y LHCb

El modelo estándar[1] de física de partículas (ME) es una teoría física formulada en la segunda mitad del siglo XX y tiene como objetivo describir las partículas elementales de la materia y sus interacciones entre sí. Este modelo ha sido puesto a prueba en múltiples ocasiones con resultados exitosos, no obstante, se trata de una teoría incompleta dado que no aúna las implicaciones de otros modelos físicos tales como la gravedad y no tiene explicación para la materia oscura, entre muchos otros.

El ME se basa en tres familias de partículas principales en las que tanto los *quarks* como los *leptones* forman la materia existente y además se dividen en tres generaciones de la materia (fermiones), mientras que los *bosones* actúan como partículas virtuales y describen las interacciones entre las partículas mediante campos de fuerzas.

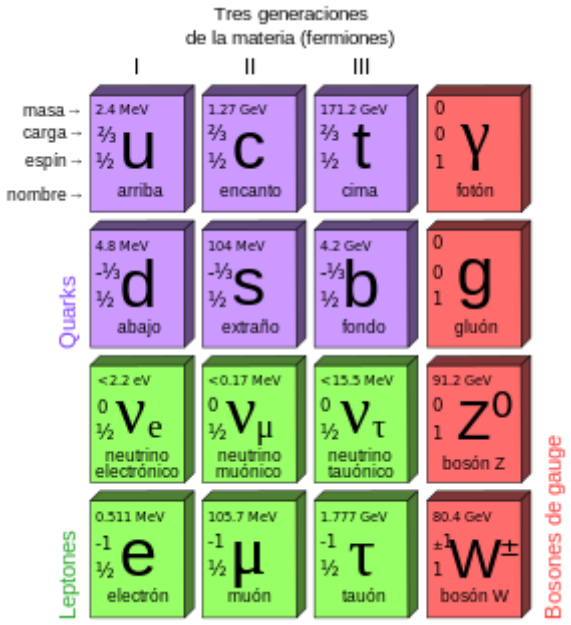

Figura 1: Partículas elementales descritas según el ME agrupando tres familias principales de Leptones, Quarks y Bosones

Mediante la combinación de *leptones* y *quarks* podemos explicar la existencia de partículas atómicas derivadas. Un ejemplo claro de este modelo es el Mesón B[2], compuesto principalmente por un *quark b* y otro quark más ligero. Sus desintegraciones y la probabilidad de cada tipo de ellas vienen descritas por el ME.

El CERN (Organización Europea de Investigación Nuclear) se encarga de realizar experimentos para corroborar las teorías de partículas formuladas, entre ellas el ME descrito anteriormente. Dentro de su organización figuran 7 subdivisiones con fines distintos y una de ellas, el LHC*b*[3] es el experimento dedicado al estudio de la violación de CP (carga y paridad) y sobre las desintegraciones raras de hadrones[4] con un *quark b*, como el caso descrito del Mesón B. El estudio de la violación de CP, relacionada con los elementos de la

matriz CKM, matriz de rotación de los estados de la masa y del *sabor de los quarks,* ayudaría a entender el porqué del exceso de materia frente a materia que observamos en el universo.

En el CERN se encuentra el Large Hadron Collider (LHC), un acelerador de 27 km de circunferencia, donde se aceleran protones a una velocidad cercana a la de la luz, alcanzando energías de 6.5 TeV. Los haces de protones se hacen colisionar en 4 puntos de interacción concretos, donde se han construido grandes detectores de partículas.

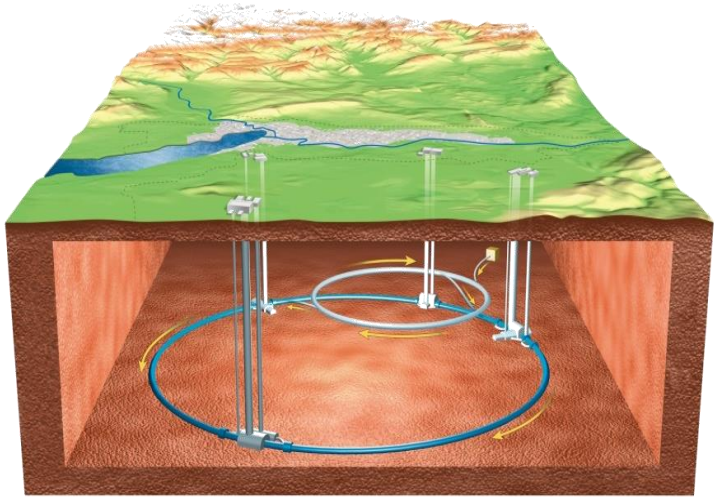

Figura 2: Esquema del LHC

A diferencia de la mayoría de los detectores de partículas, la geometría del LHC*b* consiste en un espectrómetro de brazo único que cubre ángulos pequeños con respecto al haz de protones, pero con alta precisión, dado que las partículas generadas tras la colisión toman trayectorias muy próximas al haz de protones debido al peso de las partículas bb.

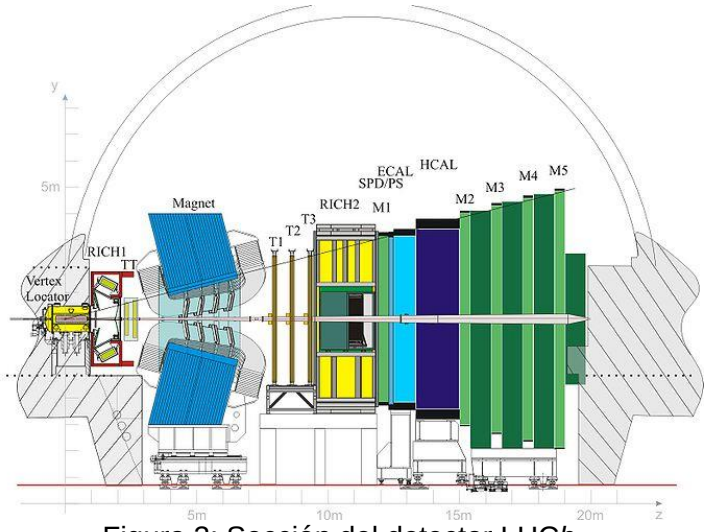

Figura 3: Sección del detector LHC*b*

Como podemos ver, en el detector hay una serie de capas de detección (hasta 14) encargadas de la medición de las trazas de partículas generadas en la colisión. Es

mediante las trazas de estas partículas por lo cuál podemos calcular directamente medidas como los puntos de desintegración (vértices), momentos lineales de las partículas y sus productos de desintegración, o de una forma indirecta, las masas resultantes de los hadrones iniciales.

En el año 2016 el Doctor Vicente José Rivés publica su tesis doctoral *"Study of b-hadron decays into two hadrons and a photon at LHCb and first observation of b-baryon radiative decays"*[5] en la cual profundizó sobre los siguientes aspectos:

- Medir la relación de frecuencias de desintegración de los procesos radiativos B0 s → φγ y Λ0 b → Λ ∗0γ con respecto a *B0 → K∗0γ*
- Medir la asimetría de CP directa para los procesos B0→ K∗0γ y Λ 0 b → Λ ∗0γ. Los estados finales estudiados se definen por las desintegraciones K∗ → K±π ∓, φ→ K+K− y Λ ∗0→ K−p (y sus complejos conjugados).

El objetivo final es participar en la misión del conocimiento formulada por el LHC*b*, no obstante, una de las problemáticas presentes en este estudio es la distinción entre lo que se define como suceso de señal y suceso de ruido.

1.2 Características de la desintegración

Según el ME de teoría de partículas podemos esperar que, de la colisión de dos haces de protones con una energía concreta, se genera un Mesón B^0 el cual se desintegra, con una probabilidad baja, en un mesón K^{*0} y en un fotón γ. A su vez, el mesón K^{*0} se reconstruye a partir de un pion y un kaon, $K^+\pi^-$ (o $K^-\pi^+$).

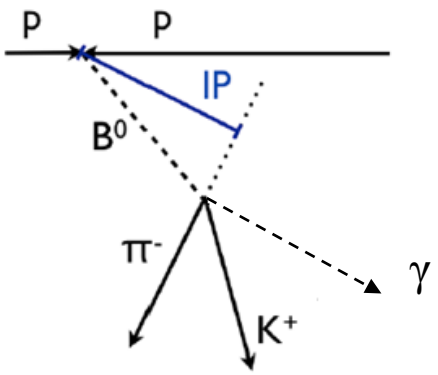

Figura 4: Representación gráfica de la desintegración del Mesón B^0

No obstante, no siempre en la reconstrucción del suceso se seleccionan los productos de la desintegración de la B^0 deseada, combinando en nuestro caso, un K^{*0} y un fotón. En ocasiones las partículas seleccionas pueden provenir de otras desintegraciones de hadrones con un quark b. El Doctor Vicente José Rivés recoge en su tesis doctoral la siguiente tabla donde se clasifican estas otras desintegraciones en dos bloques: un primer bloque de Señal y ciertas desintegraciones cuya masa del mesón B^0 reconstruido está en el rango de masa esperado, y por ello será un ruido físico indistinguible de nuestra señal, y un segundo bloque con las desintegraciones reconstruidas parcialmente (hay partículas producto de la desintegración que no han sido reconstruidas), que podrían tener una contribución relevante en el rango de baja masa del B^0 seleccionado. Finalmente, otro tipo de ruido es el combinatorio, donde se ha emparejado un mesón K^{*0} y un fotón producidos

en la colisión protón-protón aleatoriamente, sin que provengan de una misma desintegración.

	Number of events ($\times 10^6$)	Event Type
$B^0 \to K^{*0}\gamma$	12.6	11102202
$B_s^0 \to \phi\gamma$	12.6	13102202
$\Lambda_b^0 \to \Lambda^{*0}(1520)\gamma$	2.9	15102203
$\Lambda_b^0 \to \Lambda^{*0}(1670)\gamma$	5.1	15102228
$\Lambda_b^0 \to \Lambda^{*0}(1820)\gamma$	5.1	15102230
$\Lambda_b^0 \to \Lambda^{*0}(1830)\gamma$	5.1	15102240
$B^0 \to J/\psi K^{*0}$	2.0	11154001
$B^0 \to K_1^0\gamma$	1.5	11202602
$B^0 \to \rho^0\gamma$	2.9	11102222
$B^0 \to K\pi\pi^0$	1.5	11102432
$B_s^0 \to J/\psi\,\phi$	0.6	13144032
$B^+ \to D^{*0}\pi^+\pi^-\pi^0$	1.9	12265403
$B^+ \to \eta' K^+$	1.0	12203410
$B^+ \to K_1^+\gamma$	1.5	12203225
$B^+ \to K^*{}_2\gamma$	1.3	12203212
$B^+ \to K^{*+}\phi$	1.5	12103412
$B^+ \to \phi K^+\gamma$	3.1	12103202
$B^+ \to D^0\rho^+$	1.5	12165511
$B^+ \to K_1(1270)\eta$	3.1	12203410
$B^+ \to K^+K^-\pi^+\pi^0$	1.3	12103441
$B^+ \to K^+\pi^-\pi^+\pi^0$	1.5	12203203
$B^+ \to K^*\pi^+\gamma$	1.5	12203203
$B^+ \to \rho^+\rho^0$	1.2	12103401
$B^0 \to D^0(\to K^+K^-)\pi^0$	1.9	11162410
$B^0 \to D^0(\to K^+\pi^-)\pi^0$	1.9	11162400
$B^0 \to K^*\eta'$	0.9	11102441

Tabla 1: Mediciones obtenidas de la desintegración del mesón B en distintas partículas, agrupando los resultados en señal de desintegración (1º bloque) y señal de Background (Segundo Bloque)

Como comentario paralelo, se realizan jornadas de divulgación con estudiantes de institutos para colaborar en este tipo de estudios. Como ejemplo el IFCA[5], el Instituto de Física de Cantabria, realiza anualmente jornadas como estas para incentivar el conocimiento de alumnos de secundaria y que colaboren con estudios realizados por el CERN.

Hasta hace poco, lo normal es que la selección de los candidatos B^0 se basara en aplicar cortes en valores de ciertas variables cinemáticas y topológicas de los productos de desintegración (en este caso, el kaon, el pion, el fotón y el K^{*0}) y del mesón B reconstruido, comparando las distribuciones de éstas para señal y ruido. Las herramientas de machine learning resultan particularmente útiles para la clasificación de casos como el descrito, dado que buscamos identificar claramente la señal de ruido combinatorio de la señal de desintegración real.

Es importante saber diferenciar la señal de la desintegración $B^0 \to K^{*0}\gamma$ del resto de desintegraciones válidas, pero no pertenecientes a este estudio, así como del ruido combinatorio. Las distintas contribuciones de los diversos tipos de ruido se pueden estimar a partir de un ajuste a la distribución de la masa del B^0 reconstruido. Pero previamente, nos interesa reducir todo lo posible la cantidad de ruido combinatorio utilizando herramientas de machine learning, siendo precisamente éste el objetivo de este trabajo.

Una forma de seleccionar sucesos que corresponden a ruido combinatorio y poder identificar las características de este ruido es escoger sucesos a partir de un cierto valor de la masa reconstruida del mesón B:

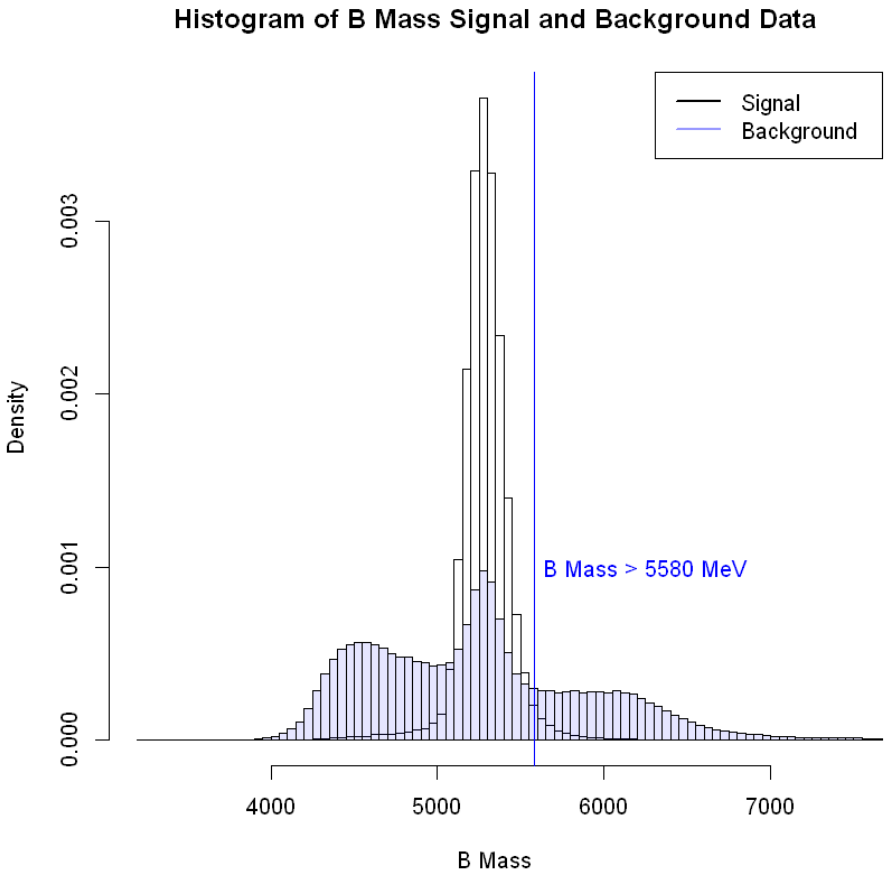

Figura 5: Histograma de frecuencias relativas de la Masa registrada del mesón B entre señal de desintegración (simulación) y datos reales (ruido y señal, en azul)

En la figura 5 apreciamos como el histograma de la señal de la masa de B tiene una forma Gaussiana con media en torno al punto de 5280 MeV, siendo esta la masa real del mesón, y una varianza muy pequeña. Para obtener esta distribución se han usado datos de $B^0 \rightarrow K^{*0}\gamma$ simulados. La distribución de color azul corresponde a datos reales seleccionados como $B^0 \rightarrow K^{*0}\gamma$. Podemos observar un pequeño pico de señal, siendo el resto ruido, registrando valores de masa entre 3500 MeV y 10300 MeV.

En la parte de baja masa, encontramos ruido combinatorio, pero también ruido de desintegraciones parcialmente reconstruidas. En cambio, la cola de background con valores de B Mass mayores a 5580 MeV, corresponde únicamente a ruido combinatorio. Por ello, se escogerá únicamente este rango de masa como ruido combinatorio para este estudio.

1.3 Trabajo realizado en clasificación de señal background sobre $B^0 \rightarrow K^{*0}\gamma$

Sobre esta desintegración concreta se han realizado múltiples herramientas de clasificación empleando machine learning en donde se ha encontrado que las que mejor resultado aportan son los clasificadores de Boosted Decision Trees, BDT.

Estas herramientas de clasificación pueden optimizarse de cara a maximizar una serie de parámetros (Que las predicciones realizadas como clasificación positiva sean siempre correctas, a riesgo que existan errores de clasificación negativa o viceversa). En este dominio, si bien profundizaremos sobre las características del dataset en los siguientes capítulos, será habitual que encontremos mayoría de la señal de background sobre la señal de desintegración obtenida. Por este motivo se selecciona el Área bajo la curva ROC[7], [8] como el parámetro a optimizar en los clasificadores.

La curva ROC, Característica Operativa de Receptor, compara el ratio de True Positive (TPR = Razón de Verdaderos Positivos) frente al ratio de falsos positivos (FPR = Razón de Falsos Positivos) también según se varía el umbral de discriminación (valor a partir del cual decidimos que un caso es un positivo). Este modelo de optimización aporta un equilibrio entre todas las variables de clasificación existentes.
\
A continuación, se muestran los resultados obtenidos en la tesis doctoral:

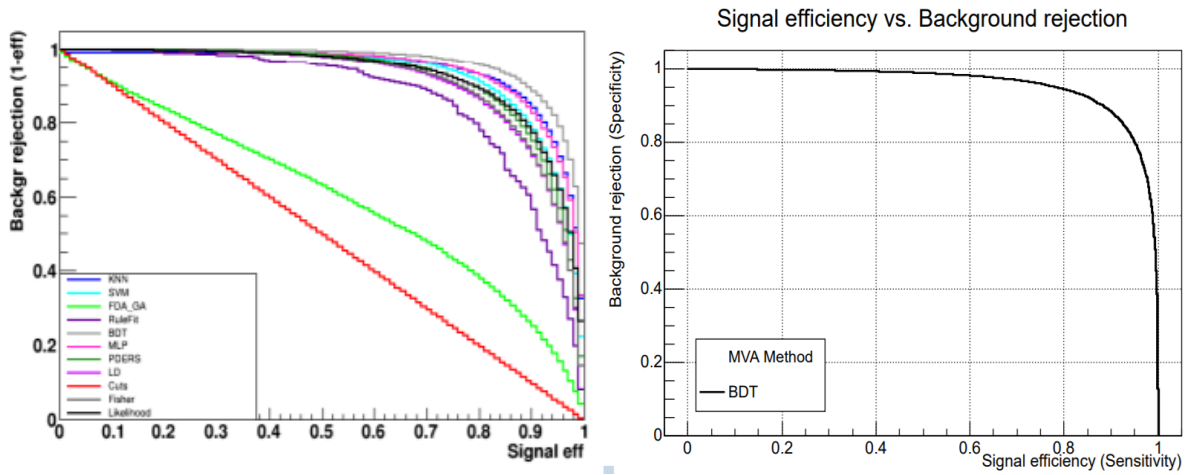

Figura 6: Curvas ROC para los distintos clasificadores utilizados y a la derecha, curva del clasificador que maximiza los resultados obtenidos (BDT)

El algoritmo de BDT empleado que maximiza los resultados obtenidos es AdaBoost, obteniendo un área bajo la curva ROC de 95.80% y utilizando un Split de los datos del 50% para entrenar el algoritmo y un 50% para hacer testing del mismo.

Para obtener estos resultados se han utilizado los siguientes parámetros:

Parameter	Value
NTrees	200
MinNodeSize	2.50%
MaxDepth	2
AdaBoostBeta	0.5
UseBaggedBoost:BaggedSamp	0.5
SeparationType	GiniIndex
nCuts	20

Tabla 3: Parámetros empleados en Adaboost para la clasificación

Como podemos ver, a pesar del resultado positivo de 95.80%, se utilizan parámetros que limitan en demasía el índice de aprendizaje del algoritmo por lo que, a simple vista, se pueden obtener mejores resultados profundizando tanto con los parámetros como con las metodologías de clasificación existentes.

2. DESCRIPCIÓN DEL DATASET Y CARACTERÍSTICAS

2.1 Descripción del dataset

Contamos con dos conjuntos de datos de similares características:

Atributo	Descripción
B_Mass	*Masa del Mesón B reconstruido a partir de los momentos y masas de K*0 y Fotón*
B_P	*Momento Lineal de la B*
B_PT	*Momento transverso de la B en el plano perpendicular al haz de protones*
B_IP	*Parámetro de impacto respecto al Primary Vertex (PV)*
B_IPChi2	*Cambio en el Chi2 del PV reconstruido si se tiene en cuenta la B reconstruida*
B_FD	*Flight Distance de la B*
B_FDChi2	*Chi2 relacionado con que vértices PV y Secondary Vertix (SV) estén separados*
B_VrtxChi2	*Chi2 del vértice de la B reconstruido*
B_Dira	*Direction Angle (Dira) de la B: El ángulo entre el momento de la B y la línea que une el PV con el SV (Desintegración de la B)*
B_DiraCos	*Coseno del B_Dira*
B_DeltaChi2OneTrack	*Cambio en Chi2 del vértice de la B si se añade una partícula más*
Kst_P	*Momento lineal del K**
Kst_PT	*Momento transverso del K* en el plano perpendicular al haz de protones*
Kst_IP	*Parámetro de impacto respecto al Primary Vertex (PV)*
Kst_IPChi2	*Cambio en el CHi2 del PV reconstruido teniendo en cuenta la reconstrucción de la B*
Kst_Helicity	*Magnitud que depende del tipo de desintegración de la B y de su spin*
Kaon_P	*Momento lineal del Kaon*
Kaon_PT	*Momento transverso del Kaon en el plano perpendicular al haz de protones*
Kaon_IP	*Parámetro de impacto respecto al Primary Vertex (PV)*
Kaon_IPChi2	*Cambio en el CHi2 del PV reconstruido teniendo en cuenta la reconstrucción de la B*
Kaon_TrackChi2	*Chi2 de la traza reconstruida*
Kaon_TrackChi2Ndof	*Chi2 de la traza reconstruida / Grados de libertad*
Pion_P	*Momento lineal del Pion*
Pion_PT	*Momento transverso del Pion en el plano perpendicular al haz de protones*
Pion_IP	*Parámetro de impacto respecto al Primary Vertex (PV)*
Pion_IPChi2	*Cambio en el CHi2 del PV reconstruido teniendo en cuenta la reconstrucción de la B*
Pion_TrackChi2	*Chi2 de la traza reconstruida*
Pion_TrackChi2Ndof	*Chi2 de la traza reconstruida / Grados de libertad*
Photon_P	*Momento del Fotón*
Photon_PT	*Momento Transverso del fotón en el plano perpendicular al haz de protones*
Photon_CL	*Nivel de Confianza*
Photon_IsPhoton	*Identificación de fotones vs piones neutros*
EventNumber	*Número de colisión registrada*
Year	Factor

Tabla 4: Descripción de los atributos de los conjuntos de datos

Podemos ver como predominan las características del bosón B y posteriormente se replican otras mismas con respecto al resto de partículas (K, Kaon, Pion y Fotón).

Los datos obtenidos sobre la señal de desintegración son construidos mediante métodos de Monte Carlo mientras que los datos obtenidos para la señal de background incluye una muestra real con datos de señal de desintegración y datos de background. El motivo por el cuál se utiliza este método es para generar datos de similares características que puedan

servir a nuestro modelo, ya que sin estos datos tendríamos pocos datos de señal con los que entrenar al clasificador.

2.2 Características de los atributos

Como conocimiento del medio en el que nos encontramos, es importante resaltar las siguientes características de los atributos:

Tabla 5: Características numéricas de los atributos del conjunto de datos

Entre los atributos descritos, veremos posteriormente que hay uno que cobra gran importancia dentro de la clasificación: B Delta Chi 2 One Track, haciendo referencia a la longitud de la traza dejada por la partícula resultante de la desintegración.

2.3 Construcción del training set

Para la elaboración de la herramienta de clasificación, seleccionamos hacemos una limpieza de datos en base a las siguientes características:

- Filtramos por background signal B Mass mayor o igual que 5580 MeV: De esta forma aseguramos que no estamos tomando señal de desintegración real.
- Retiramos atributos referentes a número de medida y año de medida: Estos atributos no tienen ningún tipo de relación y validez numérica con respecto a la clasificación final
- Retiramos atributos referentes a las masas de los mesones B y K: Queremos que el algoritmo sepa distinguir el ruido combinatorio de la señal de desintegración por características derivadas de la colisión y no propiamente por las masas registradas. Si incluimos estos datos el algoritmo estará aprendiendo artificialmente la diferencia entre ambos conjuntos.
- Retiramos atributos referentes al fotón característico en la desintegración: Las deposiciones de energía en el calorímetro no están perfectamente reproducidas en la simulación, lo que hace que los datos no sean del todo precisos.

Finalmente obtenemos un training set con las siguientes características:

- Número de instancias: 239347
- Número de atributos: 28 numéricos
- Número de clases: 2

Atributo	Tipo de Variable
B_P	Numerica
B_PT	Numerica
B_IP	Numerica
B_IPChi2	Numerica
B_FD	Numerica
B_FDChi2	Numerica
B_VrtxChi2	Numerica
B_Dira	Numerica
B_DiraCos	Numerica
B_DeltaChi2OneTrack	Numerica
Kst_P	Numerica
Kst_PT	Numerica
Kst_IP	Numerica
Kst_IPChi2	Numerica
Kst_Helicity	Numerica
Kaon_P	Numerica
Kaon_PT	Numerica
Kaon_IP	Numerica
Kaon_IPChi2	Numerica
Kaon_TrackChi2	Numerica
Kaon_TrackChi2Ndof	Numerica
Pion_P	Numerica
Pion_PT	Numerica
Pion_IP	Numerica
Pion_IPChi2	Numerica
Pion_TrackChi2	Numerica
Pion_TrackChi2Ndof	Numerica
class	Factor

Tabla 6: Atributos finales del training set

Adicionalmente, y como vías abiertas de trabajo de cara al futuro, podrían realizarse modificaciones en los atributos para reducir computacionalmente el coste del aprendizaje. Como propuestas:

- Retirar atributos con escasa importancia en los modelos
- Utilizar modificaciones logarítmicas de algunos de los atributos empleados
- Realizar proyección sobre LDA y utilizar las 5 componentes principales.

3. VISUALIZACIÓN DEL TRAINING SET

El objetivo de este capítulo es poder confirmar las características físicas del dataset según fueron explicadas en el punto 2.2, así como conocer las principales relaciones y los atributos que más importancia tienen sobre la clasificación obtenida.

3.1 Análisis de Correlación de Variables

El estudio de la correlación dos a dos de variables nos permite identificar aquellos parámetros que tienen una dependencia lineal entre las mismas.

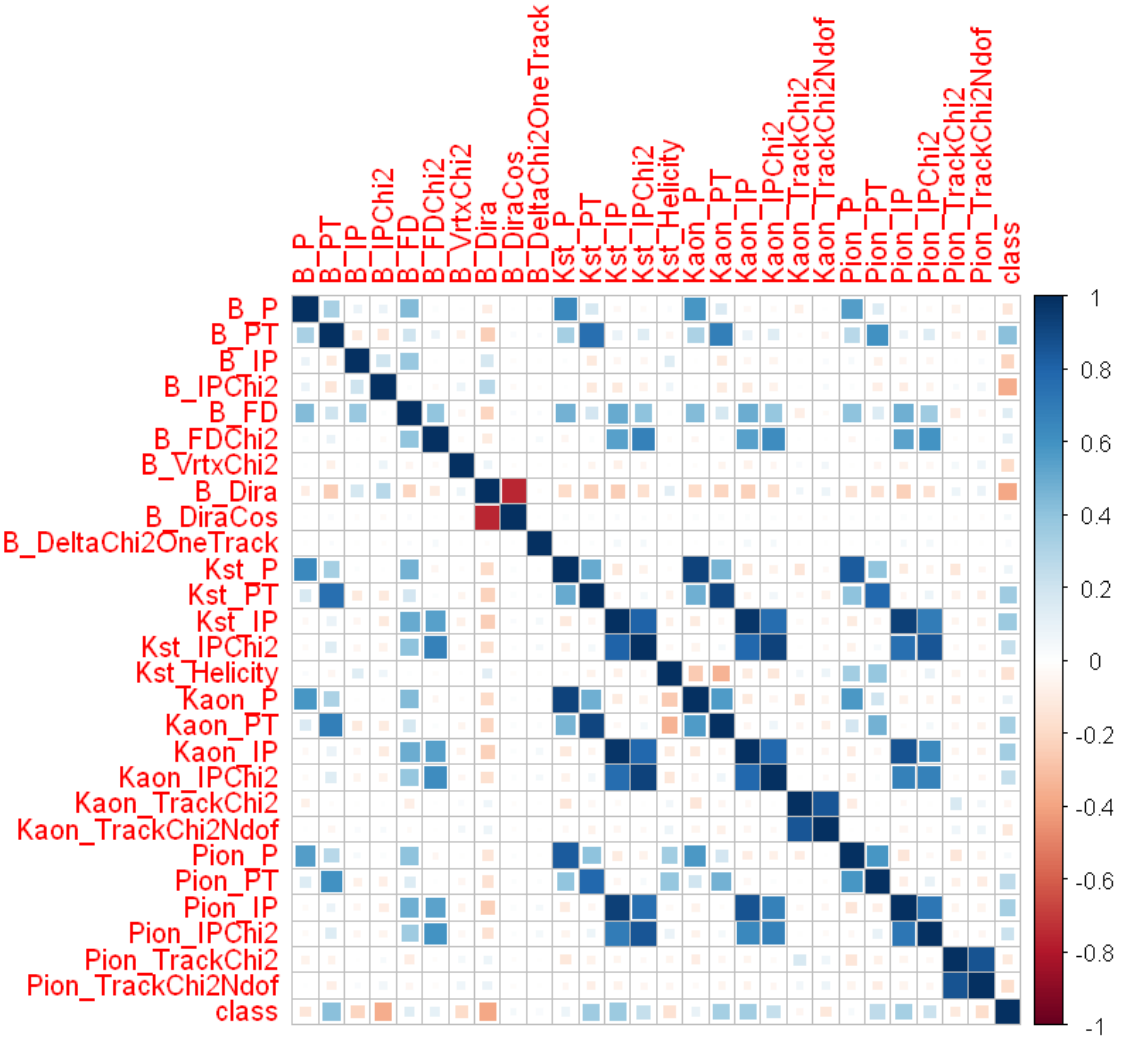

Figura 7: Matriz de Correlación de atributos del training set

Podemos identificar parámetros que tienen una relación directa dos a dos como:

- B_Dira → B_DiraCos
- Kst_IP → Kst_IPChi2, Kaon_IP, Kaon_IPChi2, Pion_IP, Pion_PIChi2

Apreciamos además que más de la mitad de las variables dentro del dataset tienen una correlación lineal con la clasificación final del dato. En base a estos parámetros podemos estimar que alcanzaremos buenos resultados con el clasificador.

Inicialmente no se toma ninguna acción sobre esta visualización, pero potencialmente podríamos realizar una reducción de componentes con las duplas de atributos correlacionados descritos anteriormente, con el objetivo de mejorar la eficiencia computacional del dataset.

En este estudio, principalmente utilizaremos Boosting Decision Trees con una profundidad de árbol controlada. Mediante esta metodología podemos mantener variables fuertemente correlacionadas como en este tipo pues este tipo de clasificadores utilizan el top X de variables que afectan directamente la clasificación final

3.2 Análisis de Principales Componentes

Este análisis nos sirve para visualizar las instancias clasificadas sobre un eje de principales componentes que separan el dataset en un plano ortogonal.

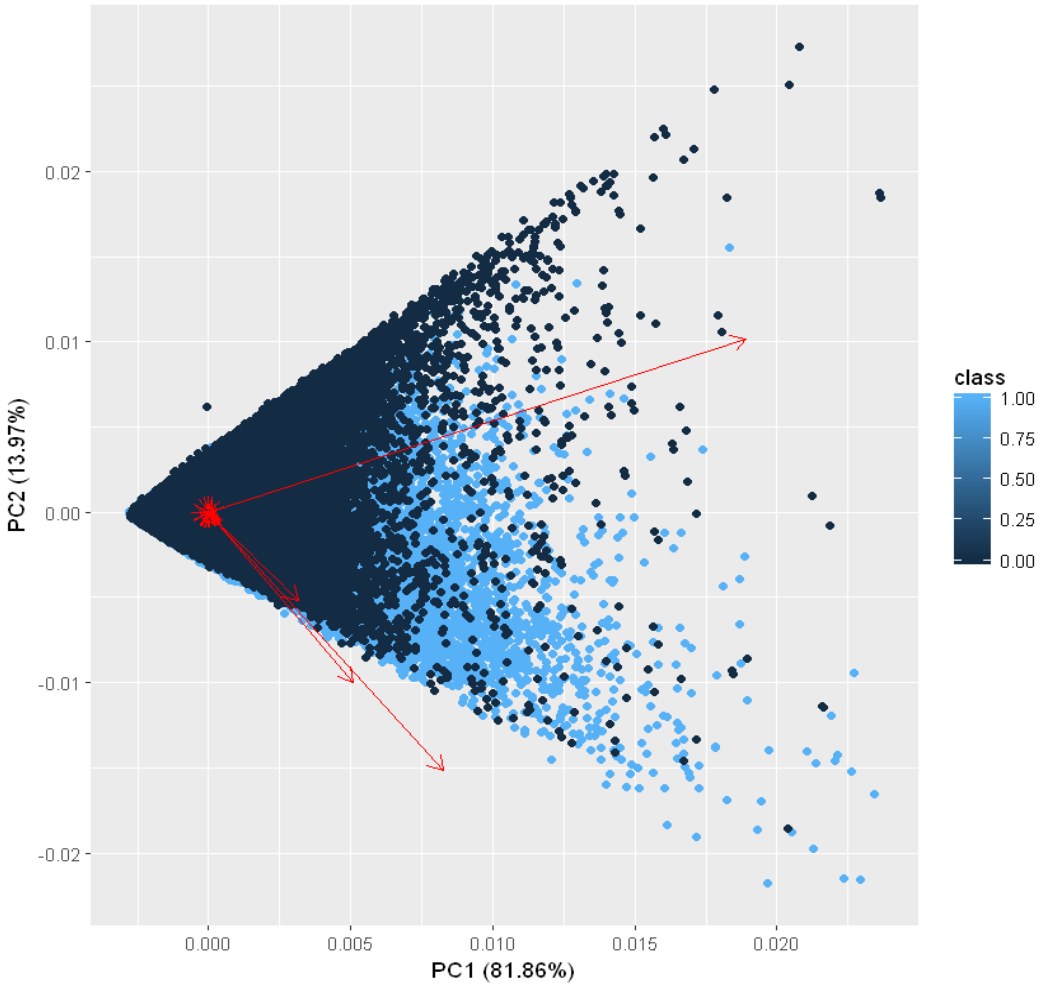

Figura 8: Visualización del training set en base a un plano de principales componentes

Por el mismo apreciamos que hay un solape evidente entre las clasificaciones en el análisis de las dos principales componentes, pero l mismo tiempo somos capaces de distinguir un segmento claro de la clase positiva cuanto mayor es la componente principal.

Si visualizamos las varianzas de las principales componentes del dataset:

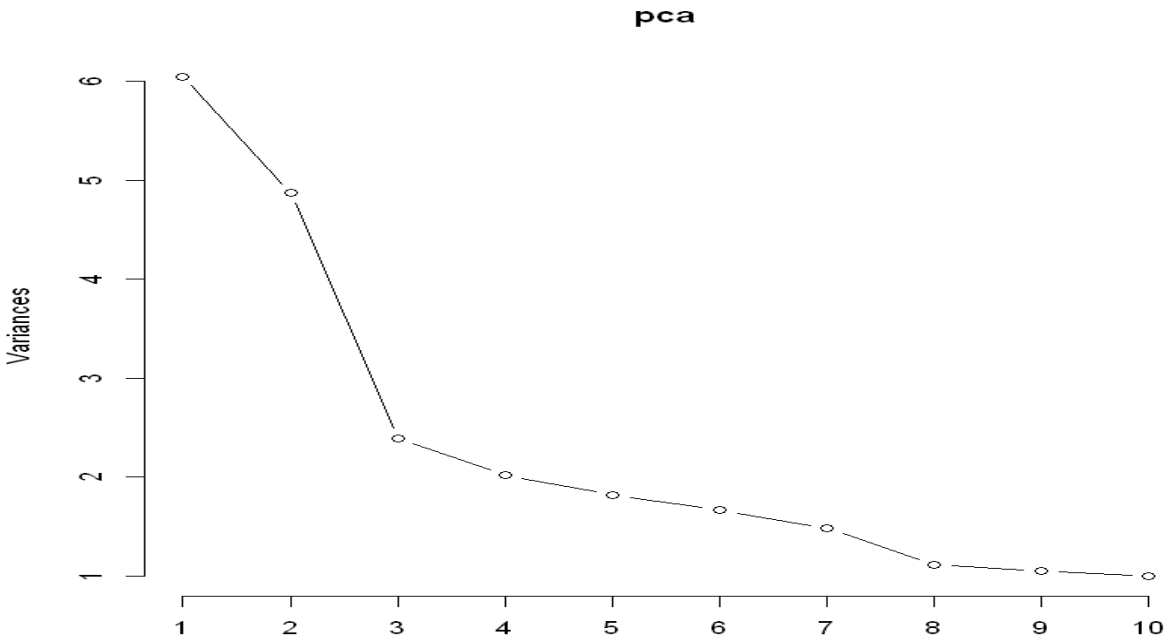

Figura 9: Visualización de la Varianza de las principales componentes en el training set

Vemos como las dos primeras componentes agrupan la mayor varianza del dataset y posteriormente, las cinco componentes principales siguientes son las que colaboran con la varianza restante.

Este ejercicio sirve para entender la varianza del dataset global pero adicionalmente, con el objetivo de reducir la dimensionalidad del conjunto de datos y reducir así el tiempo computacional, podrían seleccionarse las proyecciones de los datos sobre las siete principales componentes, reduciendo así el número de dimensiones de 28 a 7 asegurando la varianza del conjunto de datos. No obstante, este trabajo se realiza con el objetivo exploratorio de maximizar el área bajo la curva ROC en detrimento de mejora en tiempos computacionales, pero este puede ser una vía de desarrollo futura.

3.3 Visualización de principales atributos en Decision Trees

El desarrollo de nuestra herramienta de Machine Learning se basa en métodos de Boosted Trees. Por esta razón y dentro del marco de análisis exploratorio de datos, realizamos un árbol de decisión para identificar cuáles de los atributos del dataset tienen un mayor impacto sobre la clasificación final de las instancias y posteriormente, centrar el análisis exploratorio sobre las mismas.

Para ello utilizamos un clasificador de decisión tres, utilizando un test – training set de cross-validation y una profundidad de árbol máxima de 4 niveles obteniendo los siguientes resultados.

```
Confusion Matrix and Statistics

          Reference
Prediction      0      1
         0  65206  18163
         1  12764 143214

               ROC Area : 0.8619
               Accuracy : 0.8708
                 95% CI : (0.8694, 0.8721)
    No Information Rate : 0.6742
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.711
 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.8363
            Specificity : 0.8874
         Pos Pred Value : 0.7821
         Neg Pred Value : 0.9182
             Prevalence : 0.3258
         Detection Rate : 0.2724
   Detection Prevalence : 0.3483
```

Tabla 7: Resultados del clasificador de Decision Trees

Como podemos ver existe una separación evidente entre los datos pues, utilizando una de las herramientas de clasificación más básicas de Machine learning, nos encontramos con una precisión en la clasificación del 87.08% y un área bajo la curva ROC del 86.19%.

Si visualizamos el decisión tree obtenido:

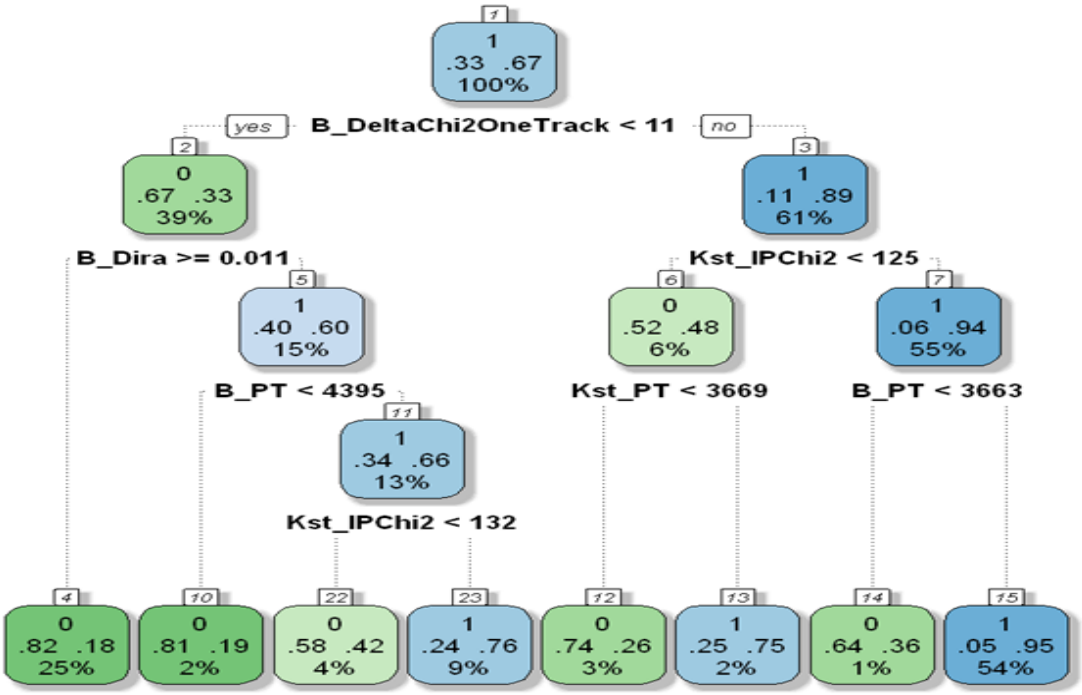

Figura 10: Representación Gráfica del Decision Tree

Con este decisión tree obtenemos una precisión relativamente elevada teniendo en cuenta que únicamente se utilizan 5 atributos del dataset para lograr la clasificación final, siendo estos:

- B_DeltaChi2OneTrack
- B_Dira
- Kst_IPChi2
- B_PT
- Kst_PT

El atributo principal por el surge la clasificación primaria es B_DeltaChi2OneTrack, representando el cambio en el Chi2 si se añade una partícula más. Recordamos que, si hacemos una lectura física de los atributos, este parámetro debe tener un valor alto pues de lo contrario nos encontramos con que la traza resultante es producida por la propia desintegración de la B y, por lo tanto, representante de ruido combinatorio.
Haciendo una lectura del propio árbol en la Figura 8 nos encontramos que las instancias con un valor de B_DeltaChi2OneTrack superior a 11 tienen una clasificación de señal y de todas estas, se logra una precisión en la clasificación del 78.80%, únicamente empleando los parámetros mencionados anteriormente. De una forma similar nos encontramos con Kst_IPChi2. Basándonos en el modelo de colisión descrito buscamos valores altos de este parámetro, representando que el Kaon no se ha producido en la colisión protón-protón. En este caso, el valor umbral seleccionado en el árbol es de 125. Cuando se producen ambos eventos (B_DeltaChi2OneTrack superior a 11 y Kst_IPChi2 superior a 125), nuestro modelo indica que estadísticamente se tratará de una instancia representativa de señal de desintegración con una probabilidad del 98.18%.

3.4 Histogramas de las principales características del dataset y training set

El objetivo de este apartado es entender la distribución del conjunto de datos sobre las clases del dataset, así como la magnitud de estas y cómo varían las clases en función de los atributos seleccionados.

3.4.1 Histograma de B Mass sobre los datasets de señal y de background:

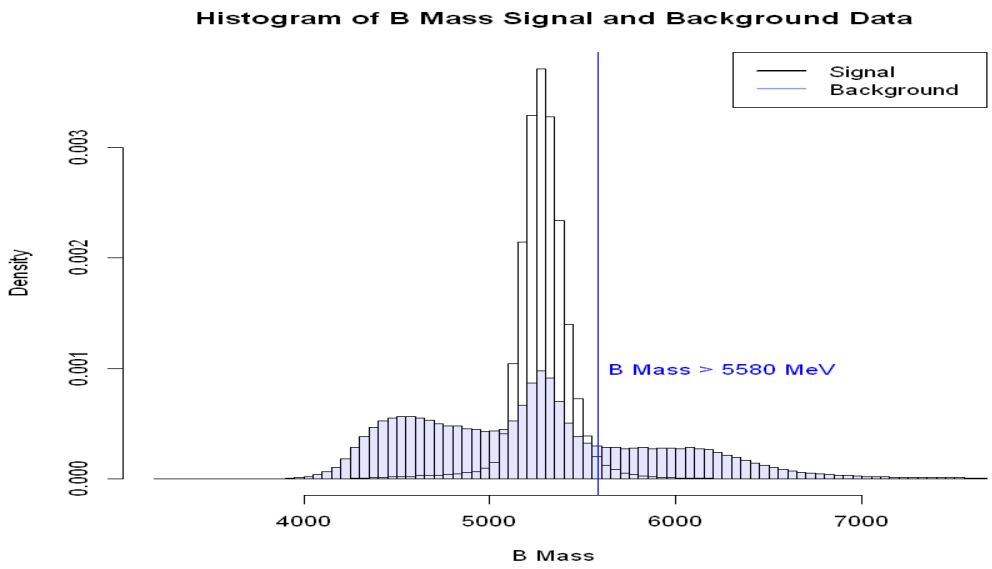

Figura 11: Histograma de frecuencias de B Mass

El histograma mostrado en la figura superior encontramos dos conjuntos de datos:

- La señal de B Mass generada artificialmente mediante métodos de Monte Carlo.
- El conjunto de mediciones reales en las que se incluyen varias desintegraciones, así como la señal de background debida a ruido combinatorio.

El método de Monte Carlo es un método de generación de datos aleatorios de una forma no determinista que permite generar muestras de datos cumpliendo unas ciertas características siguiendo una distribución aleatoria, manteniendo por lo tanto una calidad en la aleatoriedad de los datos mientras mantiene unas características básicas como media y varianza del conjunto de datos. El uso de estos métodos proviene precisamente del trabajo realizado en el estudio de la bomba atómica para generar muestras de datos representativas de procesos subatómicos.

Atendiendo a la señal de background de datos reales (En azul): Tal y como podemos ver se producen dos picos en la distribución, uno en torno a 5280 MeV, procedente de la desintegración deseada del mesón B. Finalmente se aprecia una agrupación de medidas en la parte alta de la masa, procedente del ruido combinatorio descrito en los puntos anteriores.

Atendiendo a la señal de datos generados artificialmente: Se aprecia que los datos siguen una distribución normal con centro en un valor de B Mass de 5280 MeV, siendo este el peso aproximado del mesón B y coincidiendo con el pico de muestras registradas en datos reales.

3.4.2 Histograma de Kst Mass sobre los datasets de señal y de background:

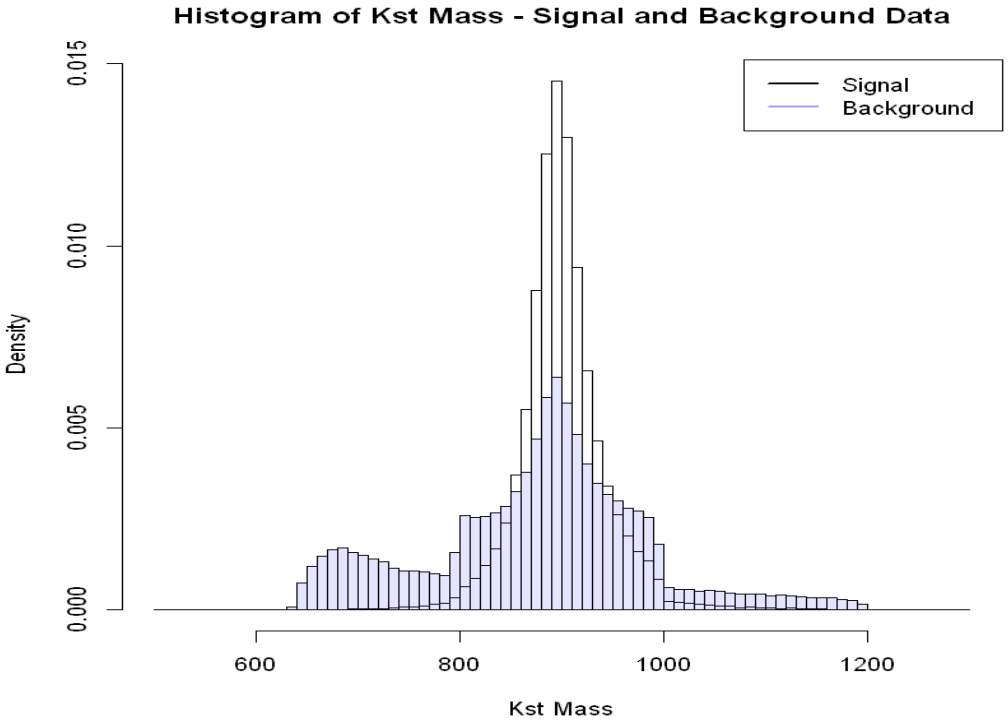

Figura 12: Histograma de frecuencias de K* Mass

De una forma similar a como sucedía en el histograma de B Mass nos encontramos con que la señal de background con datos reales tiene una variación muy alta, sin llegar a ser tan alta como en el caso del histograma de B Mass. Adicionalmente nos encontramos con un pico más reducido en la zona de masas bajas de K*.

En este caso no hacemos ningún tipo de filtro o transformación sobre el dataset para mejorar la precisión del algoritmo.

3.4.3 Histograma de frecuencias del logaritmo de B_DeltaChi2OneTrack en el training set

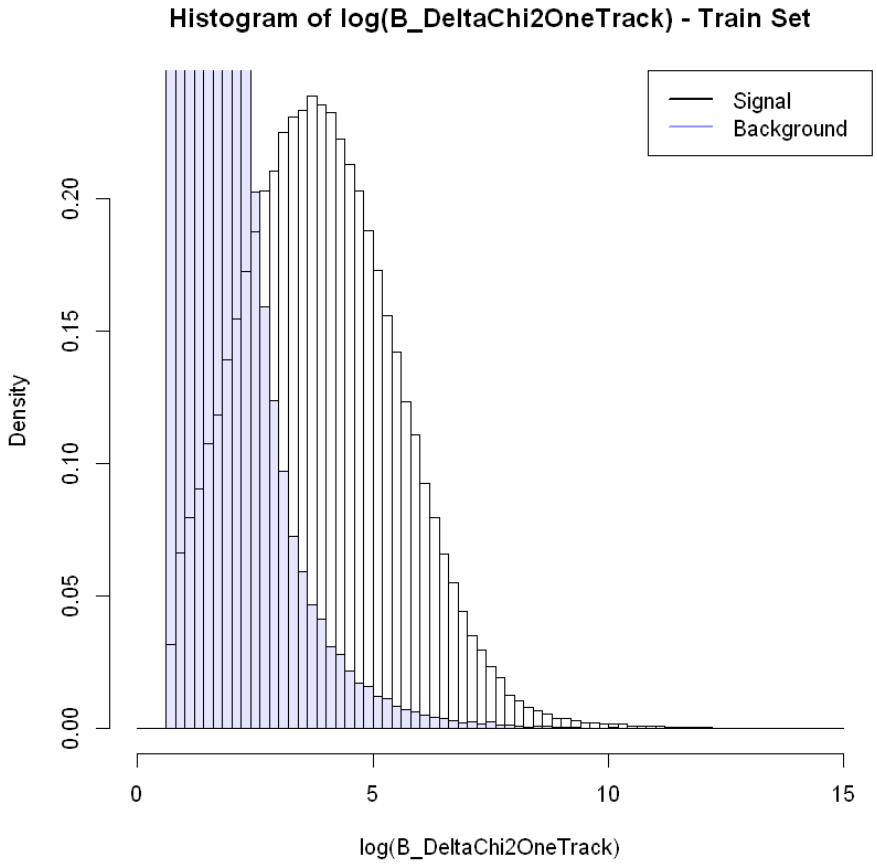

Figura 13: Histograma de frecuencias del logaritmo de B_DeltaChi2OneTrack

En este caso, con el objetivo de mejorar la visualización del histograma, decidimos aplicar el logaritmo de las mediciones en vez de visualizar las propias mediciones. El motivo de esta acción es porque existe una distribución de los datos a lo largo de varios órdenes de magnitud y se pierde capacidad de identificación en la propia visualización.

En este caso nos encontramos que la mayoría de los datos seleccionados como señal de background para nuestro training set parece seguir una distribución de Poisson con un lambda reducido, resultando en una acumulación de probabilidades para los valores pequeños del logaritmo de B_DeltaChi2OneTrack y una reducción casi exponencial con valores superiores a 3. Por contrario, atendiendo a valores de la señal, nos encontramos con una distribución más elevada con valores más altos.

Se aprecia por lo tanto que se tratan de dos distribuciones muy distintas y que, tal y como hemos definido la variable por parámetros físicos, así como los parámetros detectados por el clasificador de Decision Tree, cuanto mayor sea el valor de B_DeltaChi2OneTrack, más probable será que nos encontremos con una señal de desintegración y no de una señal de ruido combinatorio.

3.4.3 Histograma de frecuencias del logaritmo de Kst_IPChi2 en el training set

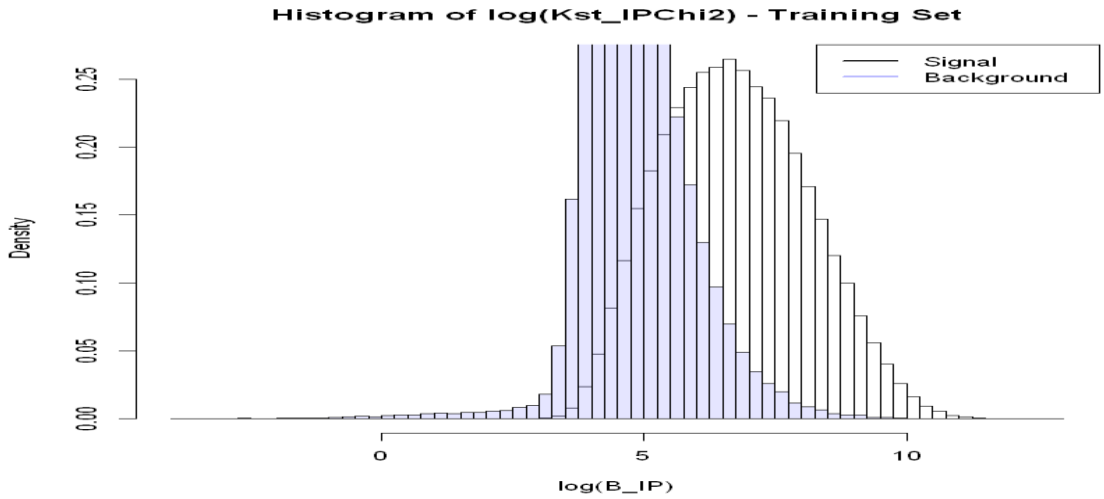

Figura 14: Histograma de frecuencias del logaritmo de Kst_IPChi2 sobre el training set

Aquí nos encontramos con un caso similar al descrito con B_DeltaChi2OneTrack, en el que tenemos una alta variación en los órdenes de magnitud de las medidas. Por esta razón utilizamos de nuevo el logaritmo de la variable para visualizar el histograma de frecuencias.
De nuevo, nos encontramos con un punto en el cuál tenemos una gran diferencia en la forma de las distribuciones y que, cuanto más alto sea la medición de Kst_IPChi2, más probable será que nos encontremos con una medición de señal y no con una de background.

3.4.3 Histograma de frecuencias del logaritmo de B_IP en el training set

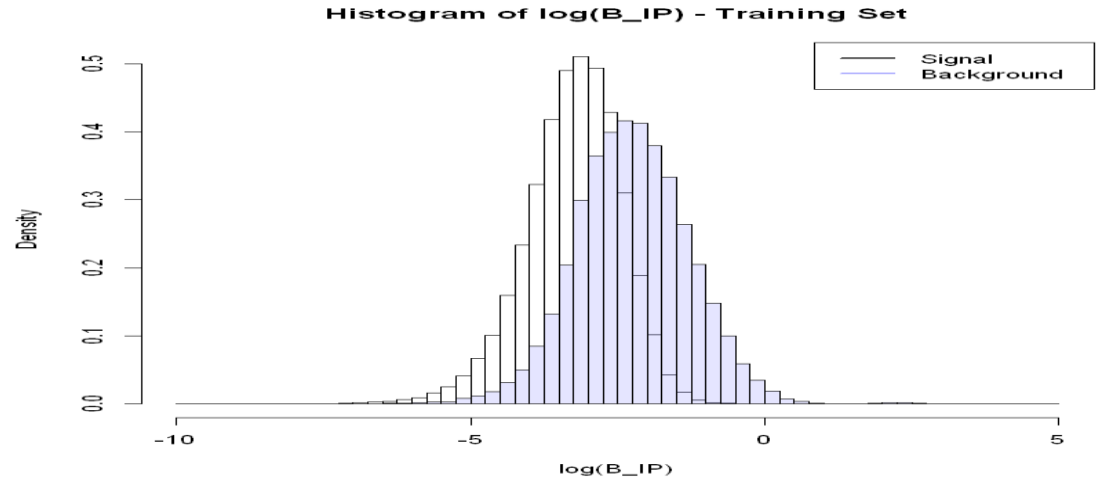

Figura 15: Histograma de frecuencias del logaritmo de B_IP sobre el training set

Las distribuciones de B_IP con respecto a las clases no difieren tanto la una de la otra. Si bien los datos de señal tienen unos valores más pequeños, existe una zona de solape grande entre ambas distribuciones. Es por este motivo por el cual nuestro clasificador inicial no seleccionó esta variable para distinguir ambas clases.

4. MODELOS DE MACHINE LEARNING

4.1 Introducción a las Tecnologías Utilizadas

A lo largo de este punto procedemos a enumerar e introducirlas distintas tecnologías empleadas para elaborar la clasificación, así como explicar la estructura de boosting de los clasificadores finales

4.1.1 Lenguajes de programación.

- **R**[9]

El lenguaje de programación R es un entorno con enfoque sobre análisis estadístico open source. Se trata de uno de los lenguajes más extendidos en el ámbito estadístico y tiene aplicaciones particulares sobre en la investigación biométrica, bioinformática, matemática financiera y sobre la minería de datos, debido a la agilidad computacional.

El paquete básico de R, en su última versión estable 3.4.4, se compone de funciones preprogramadas por la cuáles se pueden hacer transformación de datos, cálculo estadístico, visualización de datos o machine learning entre muchas otras aplicaciones. Además, dado el carácter open source del lenguaje existen “librerías” generadas por la comunidad con múltiples aplicaciones ad hoc.

Decidimos utilizar R como lenguaje de programación pues es uno de los lenguajes más completos dentro del ámbito de machine learning y veremos que tiene múltiples librerías específicas de machine learning que emplearemos para generar la clasificación de nuestros datos.

4.1.2 Entornos de Desarrollo Integrado

- **R Studio**[9]

R Studio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R. Se trata de una interfaz diseñada en exclusiva para el lenguaje R, lo que facilita un formato automático del código, así como un salto rápido a la documentación de cada función y al uso de estas.

R Studio además ofrece la posibilidad de usar Notebooks, siendo éstos una interfaz de código en los que se integran celdas de código, outputs de cada celda y anotaciones dentro del mismo documento. Resulta especialmente útil para documentar y anotar en los progresos realizados en el trabajo

- **Project Jupyter: Jupyter Notebook**[10]

Project Jupyter es una organización sin ánimo de lucro creada para “desarrollar software open-source con open estándares y servicios para computación interactiva, con el objetivo de dar soporte a docenas de lenguajes de programación”. Inicialmente se creó para dar soporte a tres lenguajes principales, conformando las siglas del acrónimo Jupyter: Julia, Python y R.

Uno de sus productos son los denominados “Jupyter Notebook”. Se trata de una interfaz basada en entorno web con una funcionalidad similar a los notebooks de R Studio: Documentar los procesos de un código, así como generar los outputs de cada celda de código.
Estos notebooks pueden generar outputs en distintos formatos open standard como HTML, LaTeX, PdF o muchos otros y por este motivo son un estándar en la comunidad de Data Science y machine learning

4.1.3 Librerías y Software de Machine Learning

- **Caret Libary**[11]

Classification And Regression Training (Caret) es un paquete de R en el que se definen una serie de funciones que dan soporte al proceso creativo de modelos de datos predictivos. Incluye funciones como:

- Separación de los datos
- Preprocesamiento de datos
- Selección de atributos
- Ajuste de modelos utilizando sampling de datos
- Estimación de importancia de los atributos

Basaremos el estudio de predicción de nuestro dataset con esta librería, la cual da soporte a más de 200 modelos de machine learning distintos. En los siguientes puntos enumeramos y describimos la totalidad de los algoritmos empleados en este trabajo.

- **H2o.io**[12]

H2O es un software open-source empleado en big data analytics cuyas principales características es la elaboración de miles de modelos, así como el descubrimiento de patrones ocultos dentro de los datos.

Este software funciona con lenguajes de programación de R y Python, pero tiene la ventaja de funcionar en un espacio virtual escrito en Java. Esto le permite que, aunque el software y las funciones se generen desde código R, la computación se realice en un espacio paralelo en Java, generando así resultados con una velocidad computacional mucho más elevada que con la librería Caret por sí sola.

Actualmente se trata de uno de los software líder en el mercado en el ámbito de Data Science y Machine learning. La consultora tecnológica Garnter ha realizado este año un estudio de las principales empresas en este ámbito, en donde H2O se posiciona como la empresa más visionara del sector[13], hecho que le ayuda a posicionarse entre las empresas líderes como SAS, KNME o Rapidminer entre otras.

Figura 16: Cuadrante mágico de Gartner para herramientas de Machine learning y Data Science (2018)

En este trabajo utilizaremos h2o sucintamente pues tiene la posibilidad de integrarse como función de entrenamiento en la librería caret, pero como vías abiertas proponemos el uso e investigación de esta solución con el objetivo de reducir los tiempos computacionales.

4.1.4 Algoritmos de Machine Learning – Boosted Trees

Recordamos que nuestro trabajo se centra en la predicción de señal o background de cada una de las instancias dependiendo de las características de los datos. Se trata por lo tanto de un trabajo de clasificación de instancias y por lo tanto emplearemos algoritmos de clasificación.

Por el trabajo realizado por José Vicente en su tesis vemos que los algoritmos que mejor resultado le han dado son los Gradient Boosted Trees por lo que centramos la investigación del trabajo en identificar los modelos GBT más potentes en la actualidad.

- **XGBoost**[15]

Se trata de una librería de machine learning open-source adaptada a lenguajes de programación C++, Java, Python y R. La misión del proyecto busca aportar “Librerías de Gradient Boosting (GBM, GBRT, GBDT) escalables y distribuidas”

Fue creado en el año 2014 y si bien 4 años en el ámbito de machine learning pueden suponer una desventaja competitiva, este algoritmo se mantiene robusto en su aplicación y consigue excelentes resultados en la actualidad. Tanto es así que ha ganado varios premios emitidos por entidades de Machine Learning y ha sido una de las soluciones más utilizadas en el sitio web kaggle, donde se realizan competiciones de Data Science y Machine learning a nivel mundial. Como ejemplo y como proximidad del ámbito de aplicación, XGBoost fue la solución ganadora del Kaggle Challenge del Bosón de Higgs[16].

Gracias al éxito de XGBoost se utiliza en la actualidad en el CERN[17] en otros departamentos con aplicación en la física de altas energías, por lo que a priori parece un algoritmo adecuado para nuestro dominio.

Finalmente se trata de un modelo que tiene como objetivo minimizar el posible overfitting del modelo en base a parameter tuning.

- **Catboost**[18]

Se trata de una librería de machine learning open-source adaptada a lenguajes de programación C++, Java, Python y R.

El algoritmo fue publicado en noviembre de 2017 por la empresa rusa Yandex, logrando unos resultados muy positivos en los benchmarks estándar de Machine Learning.

Entre sus características encontramos:

  - Robustez: Sin necesidad de optimizar multitud de parámetros
  - Tratamiento de variables categóricas: Puede gestionar variables categóricas mientras que otras herramientas de la competencia no pueden.
  - Posibilidad de optimizar en base a funciones de pérdida personalizadas.

Si bien nuestro dataset carece de variables categóricas, CATBoost se utiliza actualmente en el CERN[19] por lo que a priori, tal y como sucedía con XGBoost, parte como uno de los algoritmos más interesantes a utilizar en este trabajo.

- **Adaboost**[20]

Adaptive Boosting (AdaBoost) es un algoritmo de clasificación publicado en 2003 y sirvió de base a los algoritmos de Boosted tres.

Se basa en generar *weak learners* como es el decision tree generado en el punto anterior y aprender de los errores cometidos en cada uno de ellos para generar nuevos *weak learners* que conozcan este error y se adapten al mismo, generando un *strong learner* con reiteraciones del proceso. Esta práctica se denomina boosting en el ámbito de machine learning.

Una de las mayores problemáticas de este algoritmo es que, de llegar a repetirse demasiadas veces el proceso puede generarse un sobre entrenamiento del clasificador, haciendo que nuestra herramienta carezca de validez estadística fuera de nuestro dominio de datos.

Este algoritmo fue el ganador dentro de la selección de algoritmos empleado en el trabajo previo realizado en la tesis doctoral.

- **GBM**[21]

Gradient Boosting Machine (GBM) es una herramienta de gradient boosting utilizada en ámbitos de regresión y clasificación estadística. De una forma similar a como sucede en Adaboost, este método utiliza *weak learners* reiteradamente para generar un *strong learner*, optimizando una función objetivo de pérdida diferenciable.

Como aplicaciones comerciales, este algoritmo es utilizado en motores de búsqueda como Yahoo y Yandex, sirviendo de base de investigación al ya mencionado CATBoost.

- **Adabag**[22]

Adaptive Bagging (Adabag) es una versión alternativa del ya mencionado Adaboost, en el que en vez de basarse en el boosting de *weak learners* utiliza una estrategia de Bootstrap Aggregating (Bagging), en donde se repiten varias iteraciones del *weak learner,* pero no se tiene en cuenta el error cometido en la iteración anterior, sino que del dataset se generan nuevos subconjuntos de datos basados en los existentes y se produce un “empaquetado” de todos los modelos.

4.1.5 Algoritmos de Machine Learning – Otros Algoritmos

- **KNN**[23]

K Nearest Neirghbours (KNN) es un algoritmo de machine learning de clusterización. En este se produce un aprendizaje no supervisado de los datos, por lo que no tiene en cuenta la clasificación inicial de las instancias.

El objetivo del algoritmo es encontrar un número de centroides K definido de antemano, de forma que los datos se agrupen alrededor de estos centroides y minimizando las distancias existentes con los mismos. Este tipo de algoritmos tienen una gran aplicación sobre conjuntos de datos que no se conozca el dominio dado que encuentra relaciones entre los datos que a priori pueden no ser evidentes.

Puede utilizarse adicionalmente como herramienta de clasificación, buscando aquellas características comunes en los datos dentro de las clases. Por lo general y salvo que no exista una separación evidente entre las clases no suele ser un algoritmo muy popular como algoritmo de clasificación.

- **Treebag**[24]

Tree Bagging (Treebag) es un algoritmo muy similar a Adabag, en el que se utilizan decisión trees como *weak learners* y posteriormente se realiza un bagging de los resultados para mejorar la función objetivo. En este caso difiere de Adabag por los parámetros empleados dentro del tuning del algoritmo, pero en esencia tienen un comportamiento similar.

- **RandomForest**[25]

Random Forest es un algoritmo de bagging utilizado para la clasificación de instancias similar a Adabag en el concepto de utilizar decision trees y de hacer bagging de los mismos, pero en este caso se utilizan valores generados aleatoriamente para hacer el tuning de cada decision tree y se promedia los resultados finales obtenidos, de forma que no existe una correlación de los árboles y teóricamente reduce las posibilidades de hacer un overfitting sobre el modelo, no obstante no lo llega a conseguir y se producen casos de overfitting severo en grupos de datos poco uniformes o ruidosos.

4.2 Estructura del modelo

El objetivo inicial de nuestro trabajo será evaluar independientemente todos los algoritmos mencionados en el punto anterior de forma independiente y comparar los resultados obtenidos como referencia.

Posteriormente y una vez evaluado los resultados individuales de los algoritmos realizamos una estructura de 3 capas definida a continuación:

4.2.1 Clasificación individual de los distintos clasificadores

El objetivo de esta capa es generar distintos algoritmos con parámetros de tuning distintos dado que varios de los clasificadores parten de principios similares, para evitar que muestren resultados de clasificación similar.

Figura 17: Representación gráfica de los 6 clasificadores utilizados finalmente

Una vez generado las predicciones de los clasificadores encontraremos que habrá una serie de instancias que todos los clasificadores identifican correctamente mientras que habrá un número de instancias que son correctamente clasificadas por unos y no por otros. El objetivo por lo tanto es hacer un bagging de los clasificadores obtenidos y generar un dataset con el conjunto de predicciones para cada clasificador.

En este punto evaluaremos la posibilidad de descartar algunos de los clasificadores por posible overfitting de la muestra.

4.2.2 Feature Engineering

El objetivo de esta capa es analizar el nuevo dataset de predicciones generado por la primera capa y generar artificialmente atributos que favorezcan la reducción de ruido del dataset y por lo tanto favorezcan el resultado final de la clasificación.

En este punto utilizaremos los siguientes atributos artificiales:

- **Suma de los valores de todas las clasificaciones:** Mediante este atributo conseguimos priorizar aquellas instancias que tengan un mayor número de clasificadores correctos (Las instancias con valores 8 y 7 representan instancias que han sido clasificadas como positivas por todos o casi todos los clasificadores, mientras que instancias con valor 2 o 3 habrán sido clasificadas por menos clasificadores, generando una mayor desconfianza)
- **KNN del dataset de clasificadores:** Visualizaremos que los datos del dataset generado por al primera capa tienen una disposición espacial clusterizada por clases y que mediante KNN podemos recalcar aún más estos clústeres.

Adicionalmente existen otras técnica de feature engineering que no exploramos en el trabajo presente, pero podrían servir como vías de trabajo futuro. Entre ellas:

- Distancia mínima entre instancias de la misma clase
- Distancia mínima de cada instancia a las dos instancias más cercanas de la misma clase.
- Distancia mínima de cada instancia a las cuatro instancias más cercanas de la misma clase.

Finalmente, veremos que descartamos la opción de utilizar KNN en este paso dado que no aporta una mejora significativa sobre la separación de los datos.

4.2.3 Predicción final

Una vez transformado el dataset generado en la primera capa con los atributos artificiales de la segunda capa, incorporamos un clasificador *metalearner* que genere una predicción de las instancias.

Inicialmente utilizaremos un único clasificador. En este caso, XGBoost. No obstante, se podrían plantear otras variedades en las que se utilicen 3 clasificadores y se haga una media ponderada de los resultados obtenidos.

4.3 Medidas para evitar overfitting

Una vez definida la estructura del clasificador en tres capas hemos de tomar la decisión de qué estrategia de training y test se debe utilizar de cara a optimizar los resultados asegurando que la clasificación sea representativa del problema y no de la muestra de datos (Overfitting), así como de optimizar los tiempos computacionales de cada muestra.

4.3.1 Boosting – N Fold Cross Validation

Decidimos utilizar un procedimiento de 10 Fold Cross Validation para entrenar y testear los modelos de clasificación iniciales debido a los siguientes motivos:

- **Aleatoriedad de las muestras:** Al utilizar un método de 10 Fold Cross Validation nos aseguramos de que tenemos un modelo robusto basado en la media aritmética de 10 submodelos entrenados con 10 training sets parcialmente distintos. Si empleáramos un modelo de training y test fijo corremos el riesgo que nuestros datos no sean lo suficientemente representativos pues podemos dejarnos un conjunto importante dentro del test set.
- **Aplicación sobre un modelo de varias capas:** Al emplear el 10 fold cross validation sobre nuestro dataset nos aseguramos el generar una predicción sobre la totalidad de nuestra muestra y así generar el dataset de clasificaciones de los distintos algoritmos. Si empleáramos un modelo de train y test, obtendríamos bien un conjunto de datos pequeño, representativo de las predicciones del test set o bien un conjunto grande compuesto de las predicciones de training y test, pero corriendo el riesgo que sobre el training set tenemos un modelo sobredimensionado para la muestra.

4.3.2 Boosting – Número de Iteraciones y Learning Rate

El learning rate de los algoritmos basados en boosted trees representa el grado de aprendizaje que se produce entre una iteración y otra: Cuánto más alto sea este valor, más tendrá en cuenta el *weak learner* los errores cometidos en clasificación por el anterior árbol y por lo tanto mejor será la predicción del set.
Uno de los riesgos de emplear valores altos de este learning rate es que se produzca un overfitting del modelo sobre el training set y, por lo tanto, poner en riesgo la validez de nuestro modelo. Es importante por lo tanto encontrar un equilibrio entre el aprendizaje del algoritmo y el número de iteraciones.

Utilizando XGBoost como clasificador de referencia, realizamos sampling del dataset en el que asignamos un 80% de los datos para entrenar el modelo y guardamos un 20% de los restantes para poner a prueba nuestro modelo. Mostramos en la figura adjunta un gráfica con la evolución del aprendizaje del modelo obtenido respecto al training set y con respecto al test set en función del número de iteraciones realizadas manteniendo un learning rate de 0.3.

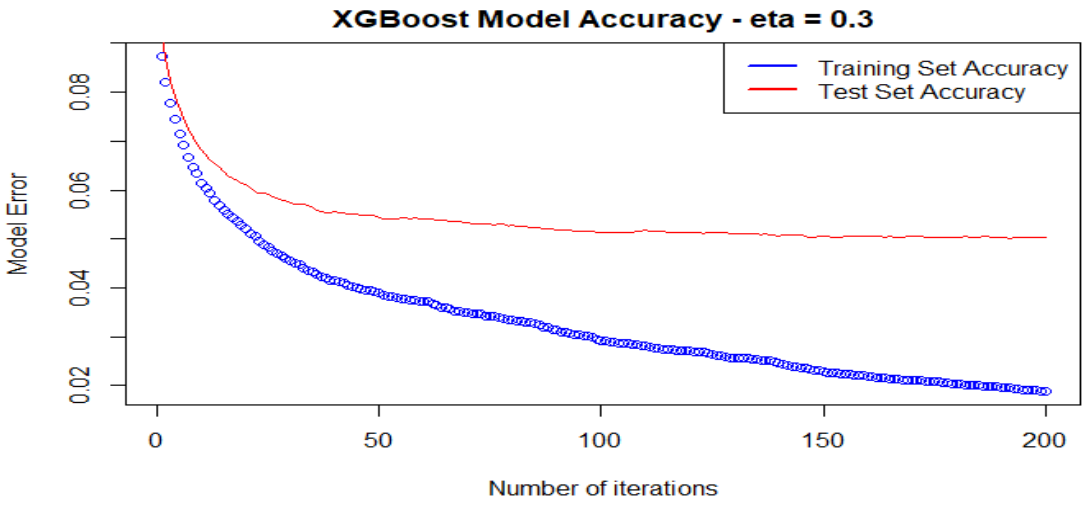

Figura 18: Aprendizaje del modelo empleando un training set y un test set con un learning rate de 0.30

Podemos ver cómo este modelo optimiza los resultados obtenidos sobre el training y el test set, pero llega un momento alrededor de la iteración 100 en el que el error del test set tiene una tendencia asintótica, sin llegar a generarse una mejora significativa en la predicción, mientras que el modelo sigue adaptándose al training set y generándose por lo tanto la situación de overfitting no deseada.

Probamos un caso opuesto en el que mantenemos el valor de eta a 0.05, un aprendizaje 6 veces inferior al utilizado en el caso anterior, pero en este caso realizamos 100 iteraciones sobre la muestra de nuestro conjunto de datos para entrenar a nuestro algoritmo

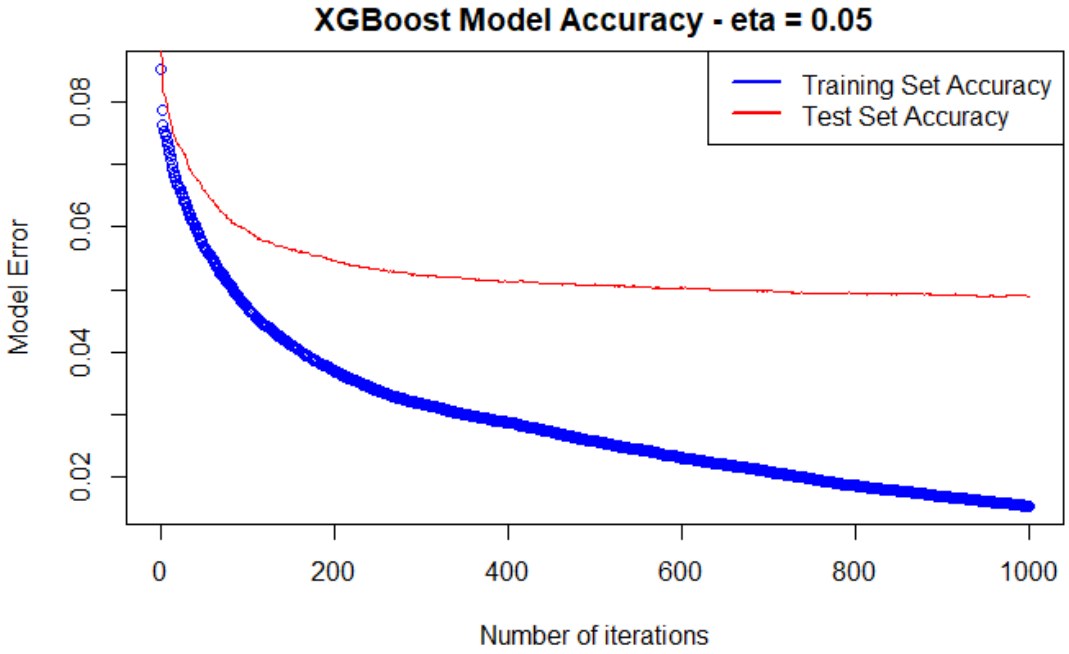

Figura 19: Aprendizaje del modelo empleando un training set y un test set con un learning rate de 0.05

En este caso apreciamos una curva similar a la obtenida anteriormente, pero lo hacemos con un cambio de escala. Donde antes utilizábamos 100 iteraciones en este momento utilizamos hasta 1000 iteraciones para el aprendizaje de la función, salvo que en este caso vemos que el error en el test set deja de tener una mejora significativa alrededor de la iteración número 600.

Las precisiones obtenidas por ambos modelos de aprendizaje (eta 0.3 contra eta 0.05) llegan a resultados similares tanto para el training como para el test set, pero utilizando un *slow learner* nos aseguramos de que minimizamos el overfitting comparando ambos modelos.

En base a los resultados obtenidos decidimos utilizar un eta de 0.05 y 600 iteraciones para el aprendizaje del algoritmo. Potencialmente podríamos establecer un aprendizaje de 0.01 y un número de iteraciones superior a 2000, pero los gastos computacionales en este caso serían mucho más elevados en comparación con el resultado final obtenido.

4.4 Clasificadores – CATboost

De cara a optimizar los resultados del algoritmo hacemos una primera prueba con un modelo de 2 Fold Cross Validation testeando el algoritmo frente a la profundidad máxima de los árboles *Depth,* optimizando el valor del área ROC generado por el modelo.

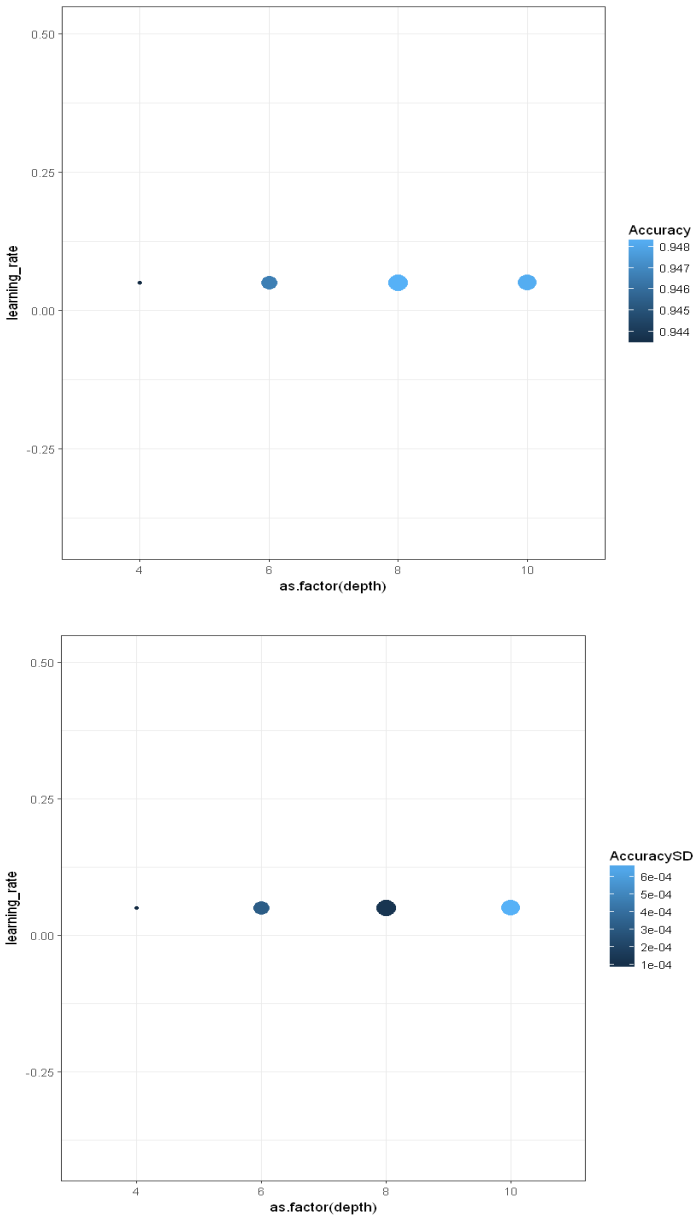

Figura 20: Representación de la precisión del modelo en función de la profundidad de los árboles

En este caso apreciamos que hay una ligera mejora creciente entre los distintos modelos, aumentando desde 94.4% hasta 94.8% empleando árboles de profundidades 4, 6, 8 y 10. SI atendemos a la desviación estándar de la precisión de los modelos, nos encontramos con que, cuánto más profundos son los árboles empleados, mayor es la variación estándar de la distribución de precisión.

Decidimos utilizar una profundidad de árbol de 8 dado que se encuentra entre los modelos con mejor precisión pero que al mismo tiempo tiene una desviación estándar menor a la obtenida por un modelo más profundo. Mostramos a continuación los resultados obtenidos por el modelo final de Catboost:

CatBoost - Confusion Matrix and Statistics

Prediction	Actual 0	1
0	71842	3516
1	6128	157861

ROC Area : 0.9898
Accuracy : 0.9497215
Kappa : 0.8845

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'depth' was held constant at a value of 8
Tuning parameter 'rsm' was held constant at a value of 0.95
Tuning parameter 'border_count' was held constant at a value of 64
```

Tabla 8: Resultados del clasificador CATBoost

Podemos ver una mejora significativa en los resultados obtenidos en la métrica de área bajo la curva ROC comparando el modelo original de Adaboost utilizado en la tesis doctoral en comparación con el modelo CATBoost optimizando los parámetros, pasamos de un 95.80% a un valor de 98.98%.

Atendiendo a las variables más importantes en el resultado de predicción del modelo:

CatBoost - Variable Importance

	Overall
B_PT	100.000
B_P	72.564
B_DeltaChi2OneTrack	45.674
B_IP	44.679
Kst_IP	33.351
Kst_PT	32.350
Kst_P	28.183
Kst_IPChi2	25.920
Kst_Helicity	24.788
B_FD	19.073
B_FDChi2	19.006
Kaon_PT	18.618
B_IPChi2	16.175
Pion_TrackChi2Ndof	14.066
Kaon_P	13.834
Kaon_TrackChi2Ndof	11.604
B_VrtxChi2	11.212
Kaon_IPChi2	9.993
Pion_PT	8.688
Pion_IPChi2	7.350

Tabla 9: Importancia de las variables en el modelo CATBoost

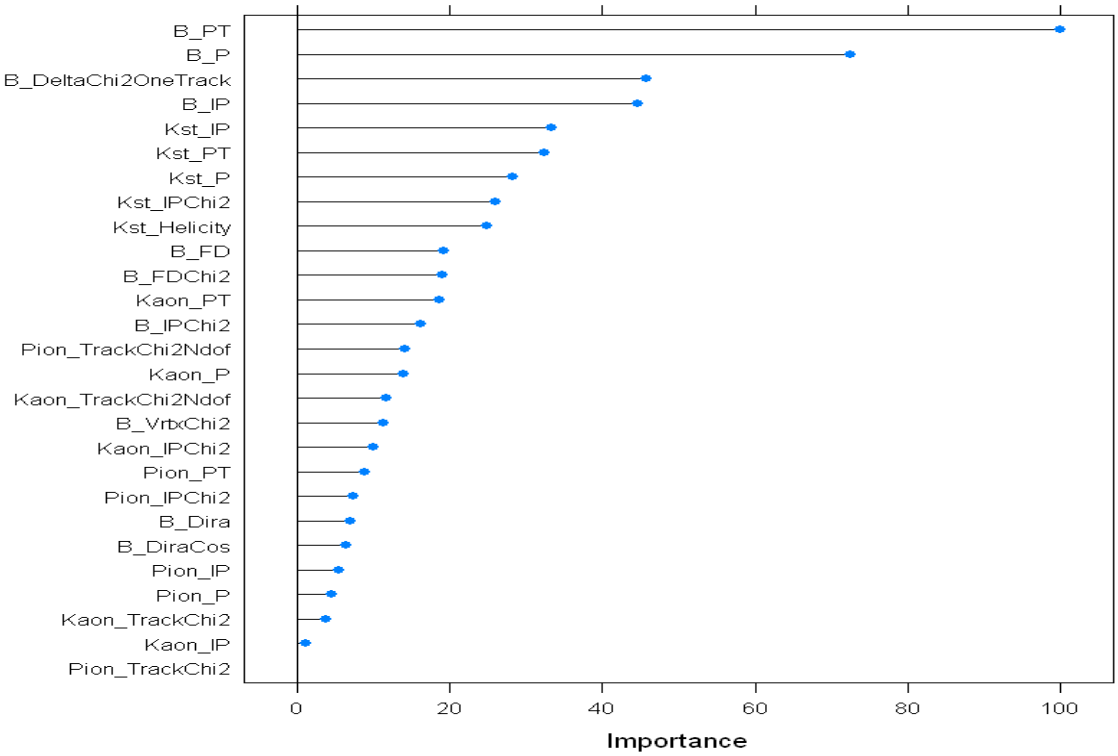

Figura 20: Representación gráfica de la importancia de variables en CATBoost

Este modelo depende en gran medida de los valores referentes a la B, dando mayor importancia a B_PT y a B_P. Estos resultados son importantes pues en el primer modelo de decision tree definido con carácter exploratorio de datos apreciamos que era otras las variables con mayor impacto sobre el modelo. Esto tendrá un gran impacto en los resultados finales pues estaremos agregando distintos modelos clasificadores que consiguen buenos resultados empleando variables importantes distintas y, por lo tanto, serán mejores clasificando ciertas instancias en comparación de otros modelos y gracias a esto conseguiremos mejorar los resultados con el *metalearner* final.

4.5 Clasificadores – XGBoost

Partiendo de la base estudiada en el punto 4.3.2 sobre el learning rate y el número de iteraciones, buscamos el encontrar los valores óptimos para el resto de los parámetros de XGBoost. En este caso evaluamos qué valor de profundidad máxima de los árboles maximiza el AUC. Hacemos una primera iteración con un modelo de 2 Fold Cross Validation:

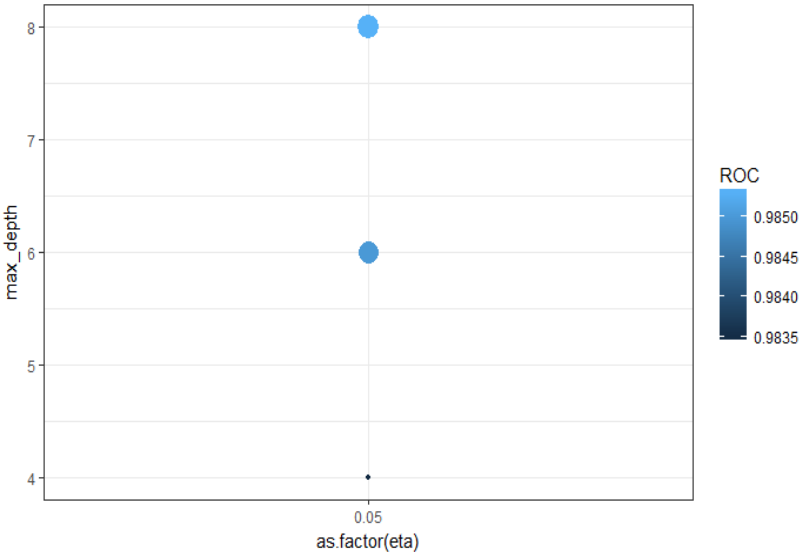

Figura 21: AUC del modelo XGBoost en función de la profundidad máxima de árbol.

Podemos ver que para este modelo no existe ninguna mejora significativa independientemente de la profundidad máxima del árbol dado que únicamente representa un orden de magnitud de milésimas. SI tenemos en cuenta que esto es una primera aproximación, seleccionamos una profundidad de árbol de 4 niveles para estar más seguros

eXtreme Gradient Boosting - Confusion Matrix and Statistics

Prediction	Actual 0	1
0	70688	4841
1	7282	156536

ROC Area : 0.9837105
Accuracy : 0.9493497

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'depth' was held constant at a value of 4
Tuning parameter 'min_child_weight' was held constant at a value of 10
Tuning parameter 'subsample' was held constant at a value of 0.50
```

Tabla 10: Resultados del Clasificador XGBoost

Utilizando XGBoost conseguimos un área ROC superior al obtenido como referencia, pasamos de un 95.80% a un valor de 98.37%. Este resultado es ligeramente inferior al obtenido por CATBoost, pero podría haber sido ligeramente similar de haber escogido una profundidad de árbol similar (4 en XGBoost contra 8 en CATBoost). En este caso no nos importa dado que buscamos clasificadores con características distintas que nos ayuden a mejorar la clasificación final.

XGBoost - Variable Importance

	Overall
B_DeltaChi2OneTrack	100.000
Kst_IPChi2	36.115
B_PT	24.586
B_IP	24.261
Kst_PT	18.456
B_P	11.484
B_Dira	8.853
Kst_IP	8.104
B_IPChi2	8.083
Kaon_IPChi2	7.920
Pion_IPChi2	7.518
Kst_P	6.200
Kaon_PT	5.139
B_FDChi2	4.286
Pion_TrackChi2Ndof	3.931
B_DiraCos	3.825
B_VrtxChi2	3.413
Kst_Helicity	3.225
Kaon_TrackChi2Ndof	2.639
Kaon_P	2.043

Tabla 11: Importancia de las variables en el modelo XGBoost

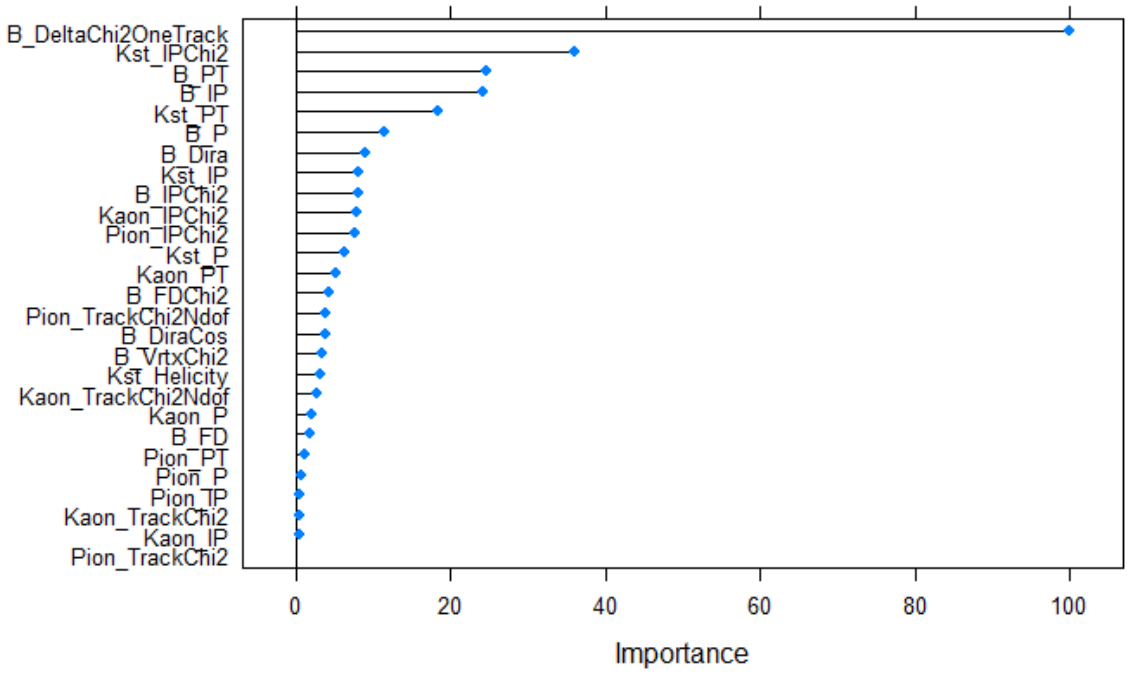

Figura 22: Representación gráfica de la importancia de variables en XGBoost

Atendiendo a las importancias de las variables dentro del modelo nos encontramos que, en conjunto, el top de variables es relativamente similares a los existentes en CATBoost. Sin embargo, la importancia de las variables más significativas es distinta a la obtenida previamente.

4.6 Clasificadores – Adaboost

Con respecto a Adaboost debemos optimizar parámetros como el coeficiente de aprendizaje (coeflearn), el número de árboles a utilizar en cada iteración (mlearn) o la profundidad de estos (maxdepth).

En primer lugar, evaluamos qué coeficiente optimiza mejor los resultados obtenidos

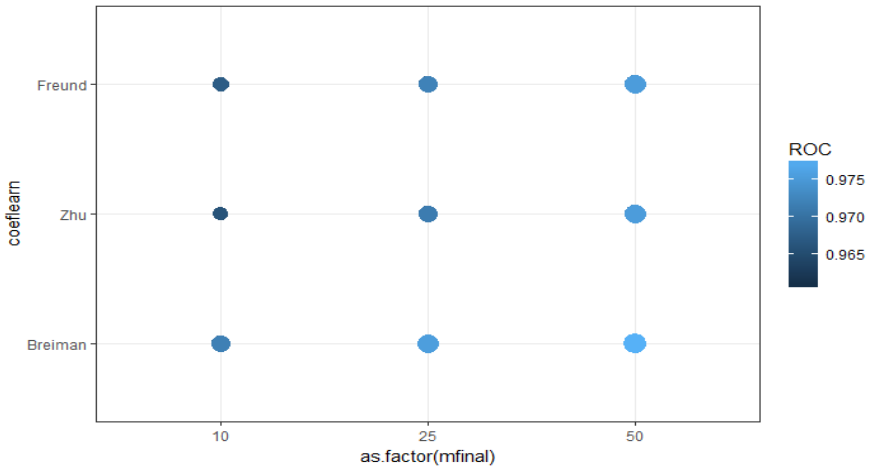

Figura 23: Área ROC obtenido en función del coeficiente de aprendizaje en Adaboost

El coeficiente de Breinman es el que alcanza unos mejores resultados sobre nuestro modelo maximizando en base a un parámetro alfa, resultante del cálculo logarítmico de los errores: alpha=1/2ln((1-err)/err)

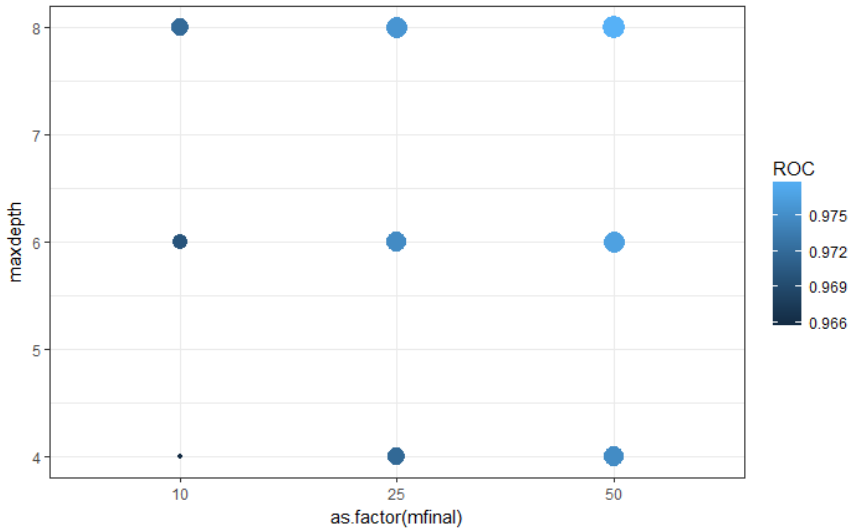

Figura 24: Área ROC obtenido en función del número de árboles y de la profundidad de estos

Existe una diferencia de más del 1% entre todas las combinaciones. Decidimos utilizar una combinación de parámetros más conservadora con el objetivo de minimizar las posibilidades de overfitting en el modelo, seleccionando por lo tanto un valor de mfinal de 25 y un maxdepth de 6 con el coeficiente Breinman.

Adaboost - Confusion Matrix and Statistics

Prediction \ Actual	0	1
0	65953	7166
1	12017	154211

ROC Area : 0.975262
Accuracy : 0.919852

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'mfinal' was held constant at a value of 25
Tuning parameter 'coeflearn' was held constant at a value of Breiman
Tuning parameter 'subsample' was held constant at a value of 0.50
```

Tabla 12: Resultados del Clasificador Adaboost

Utilizando el clasificador Adaboost con parameter tuning (600 iteraciones frente a las 200 en el benchmark o profundidad de los árboles de 6 frente a 2) mejoramos los resultados de ROC Área a 97.52%. Podemos ver como en este caso los resultados obtenidos son peores que los alcanzados con XGBoost o CATboost y aún peores en la precisión del dataset.

Adaboost - Variable Importance

	Overall
B_DeltaChi2OneTrack	42.3008
B_IP	9.6938
Kst_IPChi2	9.6667
B_PT	7.3276
Kst_PT	5.1815
B_Dira	4.3635
B_P	3.7085
Kst_IP	2.9287
Kst_P	1.7308
B_IPChi2	1.5483
B_FDChi2	1.4823
Kaon_IPChi2	1.4504
Pion_TrackChi2Ndof	1.2939
B_VrtxChi2	1.1980
Kaon_PT	0.9563
Pion_IPChi2	0.8561
Kaon_TrackChi2Ndof	0.8012
B_FD	0.7981
Kst_Helicity	0.7215
Kaon_P	0.6868

Tabla 13: Importancia de las variables del modelo Adaboost

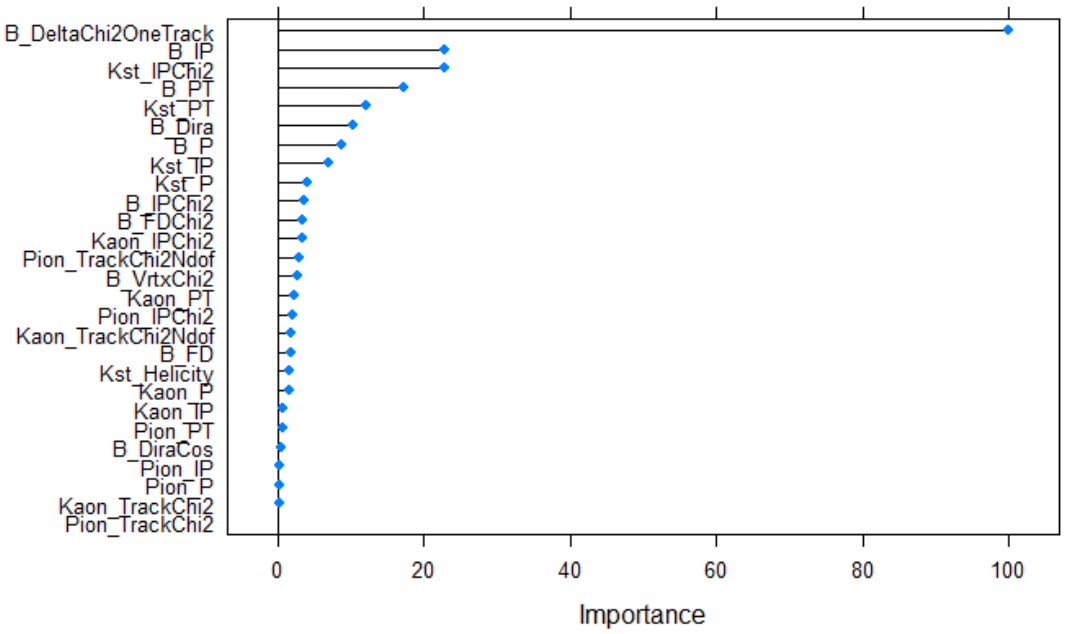

Figura 25: Representación gráfica de la importancia de variables en Adaboost

Al igual que sucede en XGBoost la variable más importante es B_DeltaCHi2OneTrack.

4.7 Clasificadores - Bagged Adaboost - Adabag

Los parámetros de optimización de Adabag son los mismos, pero este algoritmo utiliza por defecto el coeficiente Breinman como coeficiente de aprendizaje por lo que únicamente debemos optimizar el número de árboles por iteración y la profundidad de estos.

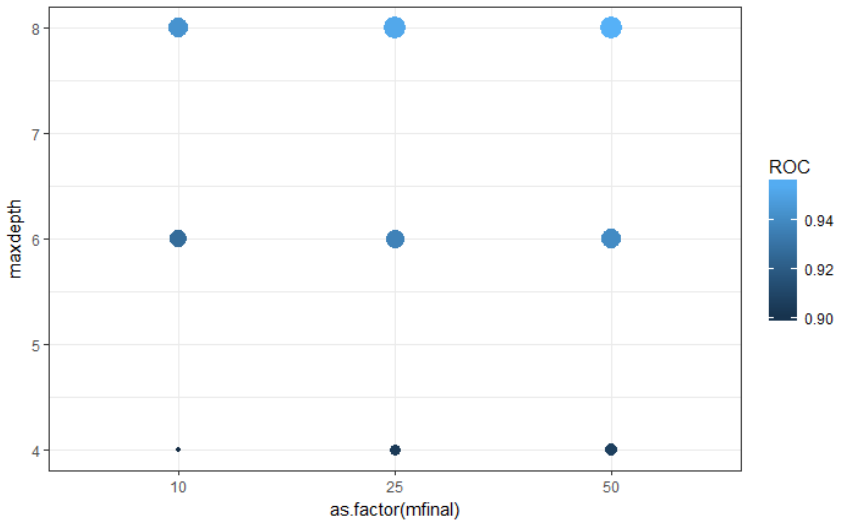

Figura 26: Área ROC obtenido en función del número de árboles y de la profundidad de estos

En esta ocasión, dado que se produce un bagging de todos los árboles calculados, optamos por unos parámetros más agresivos dentro de los umbrales de seguridad: Profundidad de 8 y número de árboles por iteración igual a 50.

Adabag - Confusion Matrix and Statistics

Prediction \ Actual	0	1
0	66295	7435
1	11675	153942

ROC Area : 0.9533945
Accuracy : 0.9198528

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'mfinal' was held constant at a value of 50
Tuning parameter 'maxdepth' was held constant at a value of 8
Tuning parameter 'subsample' was held constant at a value of 0.50
```

Tabla 14 - Resultados del Clasificador Adabag

Comparativamente con respecto al modelo de Adaboost se obtienen resultados inferiores, pero similares a los obtenidos por el Adaboost de referencia pese a tener unos parámetros que facilitan un mejor resultado del área ROC. El motivo de estos resultados obtenidos es debido al bagging producido que busca generar una aleatoriedad mayor dentro de la agregación, minimizando el posible overfitting sacrificando a cambio ligeramente los resultados obtenidos.

Adabag - Variable Importance

	Overall
B_DeltaChi2OneTrack	49.0171
Kst_IPChi2	11.4758
B_Dira	9.8359
B_PT	7.6136
B_IP	6.7229
Kst_PT	6.1637
B_P	2.0220
Kst_IP	1.5394
B_IPChi2	1.2813
B_FDChi2	0.9792
Kaon_IPChi2	0.6185
B_DiraCos	0.6177
Kaon_PT	0.3414
Kst_P	0.3325
Pion_TrackChi2Ndof	0.3262
B_VrtxChi2	0.1938
Kaon_IP	0.1831
B_FD	0.1563
Pion_IPChi2	0.1533
Kaon_TrackChi2Ndof	0.1172

Tabla 15 – Importancia de las variables en el modelo Adabag

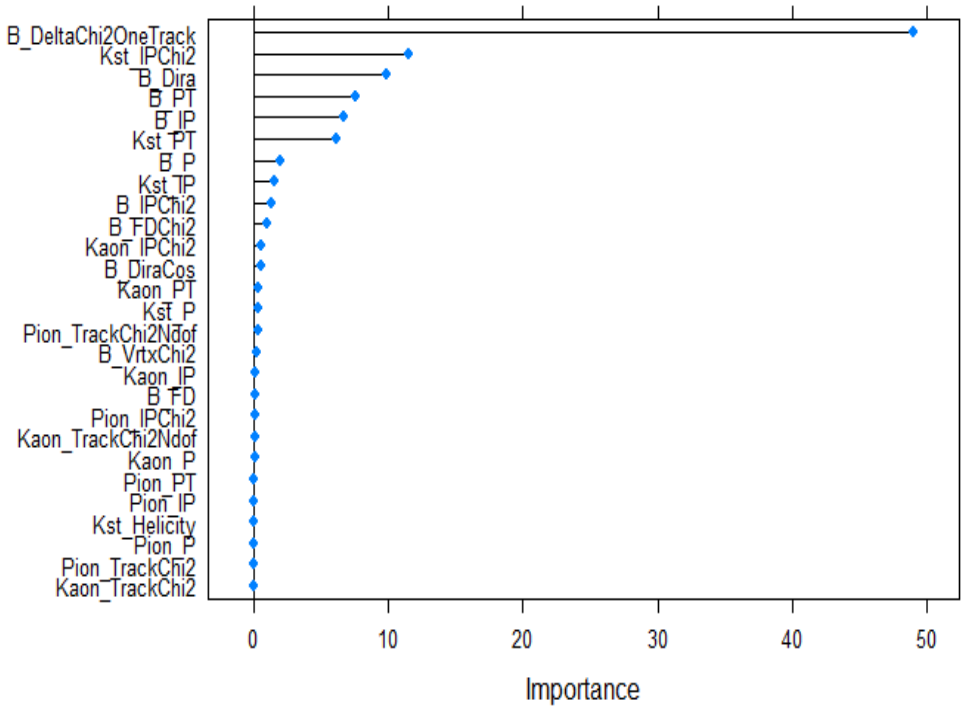

Figura 27: Representación gráfica de la importancia de variables en Adaboost

4.8 Clasificadores - GBM via h2o.io

El caso de GBM es similar a los casos vistos hasta ahora en términos de optimización de parámetros, dado que se basa en Boosting Trees. Mostramos a continuación el Área ROC en función del número de árboles y de la profundidad máxima de los árboles.

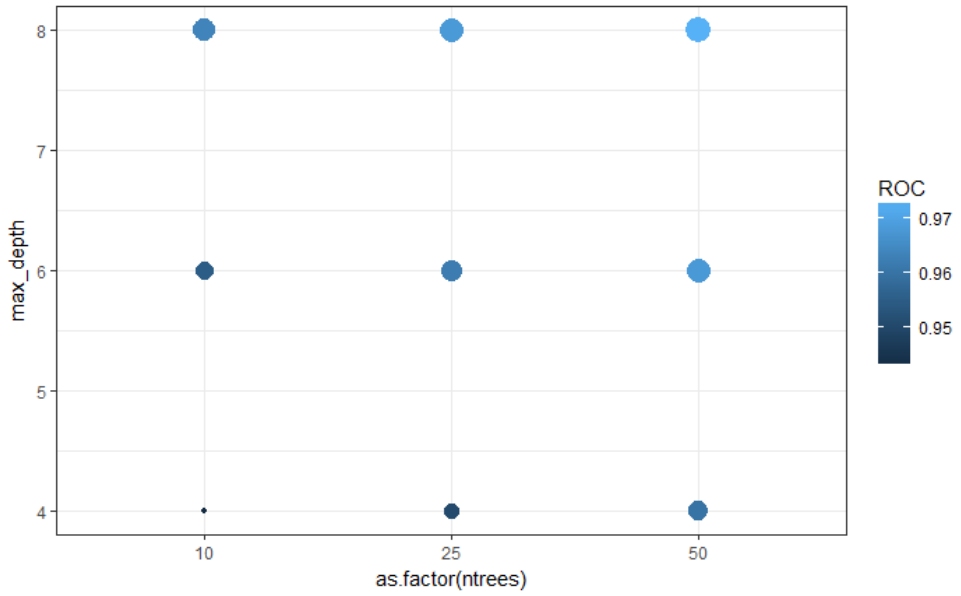

Figura 28: Representación gráfica de la importancia de variables en GBM

En este caso utilizamos una configuración media con una profundidad de árbol de 6 y 25 árboles por cada iteración.

GBM via H2O - Confusion Matrix and Statistics

Prediction	Actual 0	1
0	62492	5829
1	15478	155548

ROC Area : 0.9616831
Accuracy : 0.9109786

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'ntrees' was held constant at a value of 25
Tuning parameter 'minrows' was held constant at a value of 4
Tuning parameter 'maxdepth' was held constant at a value of 6
Tuning parameter 'col_sample_rate' was held constant at a value of 0.7
```

Tabla 16 - Resultados del Clasificador GBM

Obtenemos unos resultados medios con respecto a los clasificadores utilizados hasta ahora (Entre 95% y 99%)

GBM - Variable Importance

	Overall
Kst_IPChi2	83661.5
B_Dira	77290.6
B_PT	36873.6
Kaon_IPChi2	33566.5
Kst_PT	29255.4
B_IP	23159.3
B_P	16166.9
Kaon_PT	9084.3
Kst_IP	8785.0
B_IPChi2	6987.7
B_FD	1921.9
Kst_P	1756.7
Pion_TrackChi2Ndof	1497.3
Pion_IP	1363.8
Kaon_IP	1161.1
B_VrtxChi2	1098.3
Kaon_P	830.5
Kst_Helicity	564.1
Pion_IPChi2	527.7
Kaon_TrackChi2Ndof	464.9

Tabla 17 – Importancia de las variables en el modelo GBM

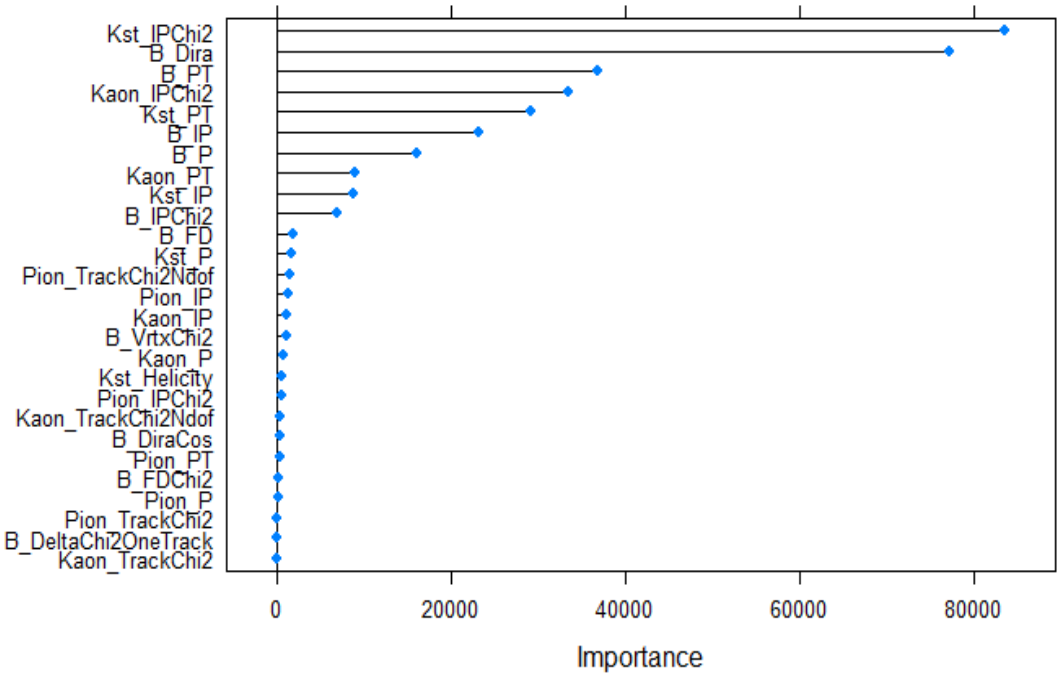

Figura 29: Representación gráfica de la importancia de variables en GBM

4.9 Clasificadores – Clasificadores no Boosted Trees

La ventaja de utilizar modelos de Boosted Trees para clasificación es que pueden adaptarse al conjunto de datos fácilmente y atendiendo a los parámetros de ajuste de los modelos, podemos asegurar que no existe overfitting y generar así un modelo representativo.

Haremos pruebas con tres algoritmos adicionales más “simples” que los testeados hasta ahora y con una necesidad menor de tuning, por lo que arriesgaremos el tener unos resultados menos precisos o tener un modelo sobre-adaptado.

Como punto adicional, estos modelos no tienen la posibilidad de representación de la importancia de las variables.

- **Knn**

El caso de Knn es un caso muy especial, dado que este es un algoritmo que por definición no tiene un uso principal para ejercer como clasificador. No obstante, puede llegar a cumplir esa función de una forma aproximada.

En este caso, en la función train de caret, solo existe un único parámetro para optimizar que es k, siendo k el número de centroides a identificar. En ámbitos de clasificación K debe ser igual al número de clases del dataset, en este caso 2.

KNN - Confusion Matrix and Statistics

	Actual	
Prediction	0	1
0	56383	0
0.33	9	4
0.50	21577	21907
0.66	1	8
1	0	139458

ROC Area : 0.8509626
Accuracy : 0.7231499

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'k' was held constant at a value of 2
```

Tabla 18 – Resultados del Clasificador KNN

De Knn podemos sacar la siguiente conclusión: Como clasificador cumple de una forma muy pobre dado que hay en torno a un 28% de las instancias que no sabe clasificar y por lo tanto le genera una probabilidad de 50%, lo que genera que la precisión del algoritmo sea la menor obtenida hasta el momento. No obstante, vemos que de todas las instancias que clasifica correctamente, no comete errores en ninguna. Esto se debe a que las instancias del dataset están muy separadas y hay una pequeña zona de solape en las que KNN no es capaz de diferenciarlas.

Utilizaremos este modelo en el dataset de clasificaciones pues, aunque no tiene mucha precisión global, agrupa las instancias dentro de una misma clase y por lo tanto puede ayudar al metalearner final.

- **Random Forest**

En este caso tenemos que optimizar los parámetros de min_node_size y mtry.

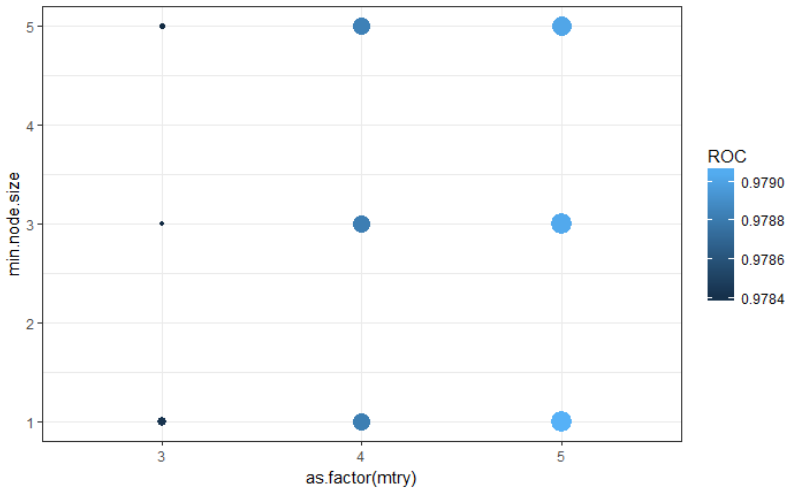

Figura 30: Representación gráfica del Área ROC en función de los parámetros de Random Forest

Random Forest - Confusion Matrix and Statistics

Prediction	Actual 0	1
0	77800	134
1	40	161373

ROC Area : 0.9790379
Accuracy : 0.9945341

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'innodesize' was held constant at a value of 5
Tuning parameter 'mtry' was held constant at a value of 4
```

Tabla 19 – Resultados del Clasificador Random Forest

Se aprecia que se consiguen unos resultados demasiado buenos y sobre-adaptados al modelo existente, dado que únicamente con dos parámetros establecidos obtenemos unos

resultados extraordinarios. Decidimos descartar este método por posible overfitting, sobre todo teniendo en cuenta que existe correlación entre las variables del dataset.

- **Treebag**

Treebag es un algoritmo que no requiere de ningún parámetro de optimización en la función train de caret. Tras realizar un training con 10 Fold Cross Validation obtenemos unos resultados que a priori resultan demasiado buenos:

Treebag - Confusion Matrix and Statistics

Prediction	Actual 0	1
0	66029	7277
1	11941	154100

ROC Area : 0.9756247
Accuracy : 0.9945341

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:
```

Tabla 20 – Resultados del Clasificador Treebag

La precisión obtenida de 99.45% nos hace sospechar que se trata de un modelo caído en el sobre entrenamiento de los datos, debido a las variables correlacionadas del dataset.

Optamos por lo tanto por descartar los resultados obtenidos por Treebag dado que, de incluirlos, tendrían un peso demasiado grande en el *metalearner* final.

4.8 Resultados Individuales

Hemos empleado 8 clasificadores distintos:

- CatBoost
- XGBoost
- GBM
- Adaboost
- Adabag
- Knn
- Random Forest
- Treebag

De los cuales descartamos el modelo de Random Forest y de Treebag debido a que parecen modelos sobre entrenados.

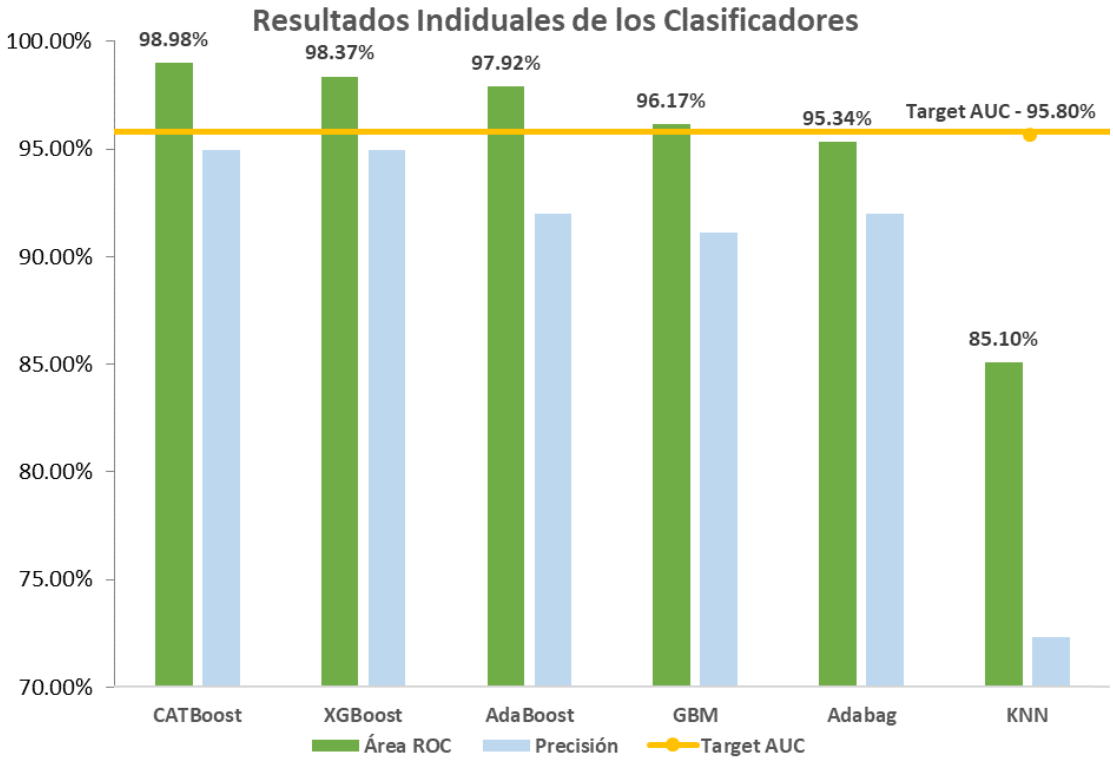

Figura 31: Comparativa de los resultados obtenidos por cada clasificador

Si hacemos una valoración del trabajo realizado hasta ahora podemos estar contentos pues, hemos logrado mejorar los resultados obtenidos inicialmente con 4 clasificadores distintos, y con tres de ellos la mejora es significativa. Este podría ser el punto final del proceso de clasificación, pero hemos visto que podríamos llegar a mejorar los resultados dado que los principales clasificadores utilizados tienen una importancia de variables distinta y, por lo tanto, pueden servir en un *metalearner* agrupado.

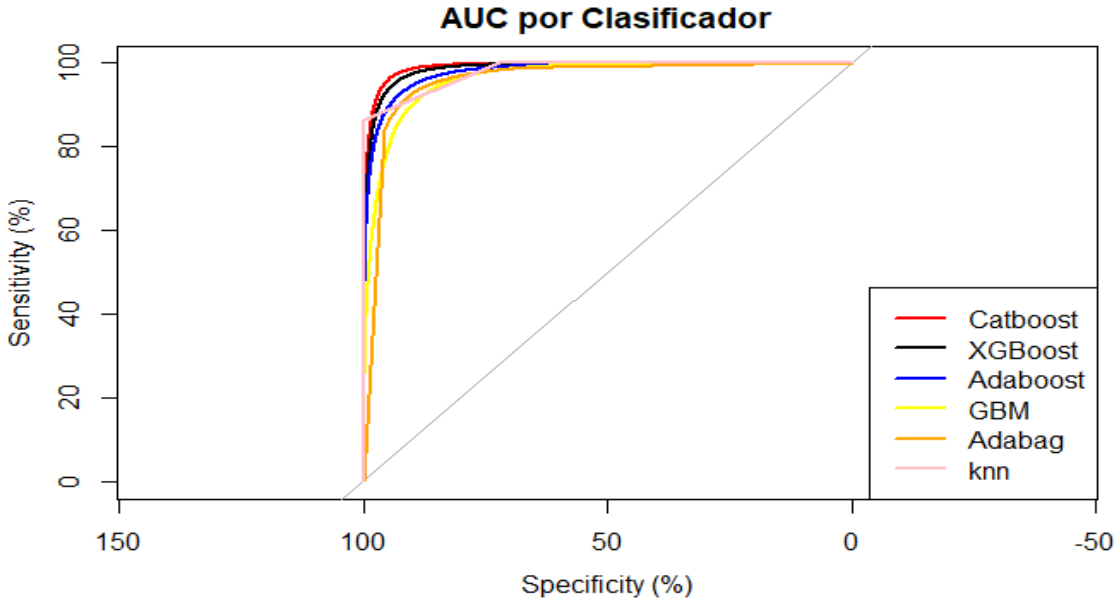

Figura 32: Área ROC del conjunto de clasificadores

Generamos un nuevo dataset resultante de todas las predicciones realizadas por cada uno de los clasificadores.

4.9 Feature Engineering

Como punto de partida generamos dos datasets distintos con los datos generados:

- Dataset con las probabilidades registradas para cada instancia por cada algoritmo, así como la clasificación de cada instancia.
- Dataset con las predicción de clase registradas para cada instancia por cada algoritmo a excepción de knn, en donde mantenemos la probabilidad obtenida debido al elevado número de instancias con predicción de 0.50, así como la clasificación de cada instancia.

El objetivo de este punto es generar parámetros adicionales resultantes de reglas heurísticas de los datos presentes, con el objetivo de separar las clases aún más y lograr así una mejora en los resultados.

En primer lugar definiremos los dos parámetros que emplearemos sobre los dos datasets y posteriormente comentaremos el impacto generado por la inclusión de estos nuevos atributos.

4.9.1 Suma de valores de la Predicción / Probabilidad de Predicción

Esta heurística tiene como objetivo el dar más importancia a las instancias que mayor número de predicciones correctas tengan (Valores de 6, siendo 6 el máximo valor posible, indican que todos los clasificadores han hecho una predicción positiva sobre la instancia. Valores de 0 o 1 indican que todos o casi todos los clasificadores han hecho una predicción negativa sobre la instancia). Esto genera que se prioricen las instancias con mayor acumulación de resultados similares por los clasificadores.

4.9.2 KNN del nuevo dataset generado

Entre los clasificadores utilizados previamente mencionamos KNN, el cuál nos sirvió para hacer una clasificación de las instancias en base a sus características numéricas. En este caso no buscamos hacer una clasificación de los resultados, sino más bien agrupar conjuntos de instancias con características numéricas similares dentro del dataset y acuciar aún más la distancia entre las dos clases existentes

4.9.3 Impacto de Feature Engineering sobre los datasets

Para evaluar el impacto del feature engineering realizado lo haremos de forma gráfica, construyendo el plotting de las instancias sobre el plano ortogonal de PCA tal y como hiciéramos en el punto de visualización del dataset. Recordamos cómo encontrábamos en el punto inicial nuestro conjunto de datos original

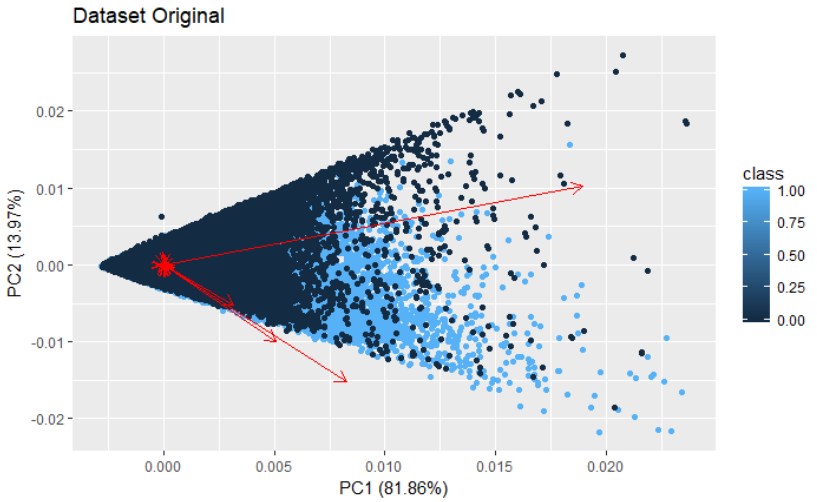

Figura 33: Visualización de las clases sobre el plano de componentes principales

Dataset de probabilidades de clases

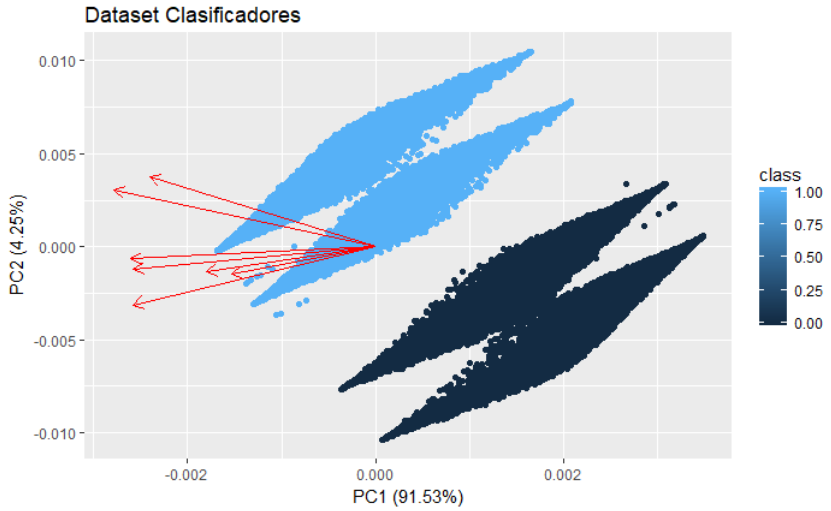

Figura 33: Visualización del dataset de probabilidades sobre el plano de componentes principales

En primer lugar, se aprecia una clara diferencia entre el dataset original y el resultante de las predicciones. Con el nuevo conjunto de datos hemos conseguido agrupar nuestros datos en 2 clústeres claramente diferenciados. Además, la componente principal calculada en el PCA ha pasado de suponer un 81.86% a un 91.53%.

Como comentario sobre las figuras adjuntas con la representación de las clases, figuran unos ejes marcados en rojo representativos de las proyecciones en el plano de la dirección de las principales componentes del dataset.

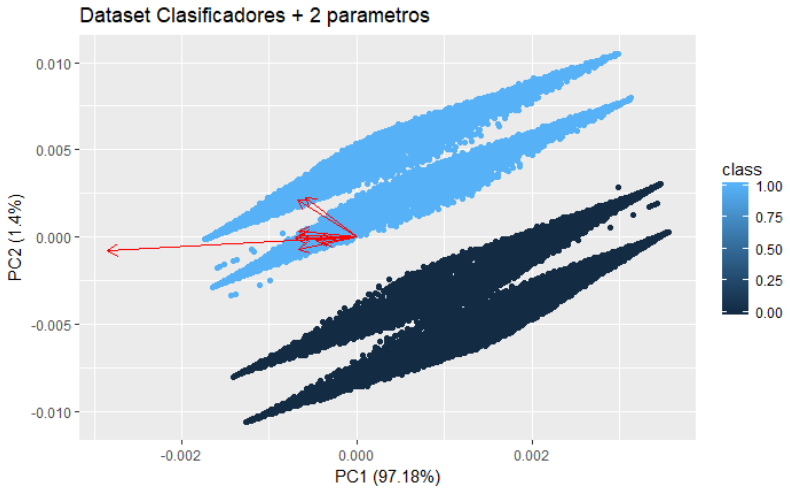

Figura 34: Visualización del dataset de probabilidades sobre el plano de componentes principales incluyendo los atributos basados en heurísticas

A priori no se aprecia una gran diferencia entre ambos conjuntos, pero en este caso la componente principal ha pasado a suponer un 97.18% frente al 91.53% anterior. Debido a esto, ambos conjuntos de datos han sufrido una homotecia por la cuál tienen una presencia mayor a lo largo de la primera componente principal y una presencia menor en la componente secundaria, logrando por lo tanto un agrupamiento de los datos.

Dataset de predicciones de clases

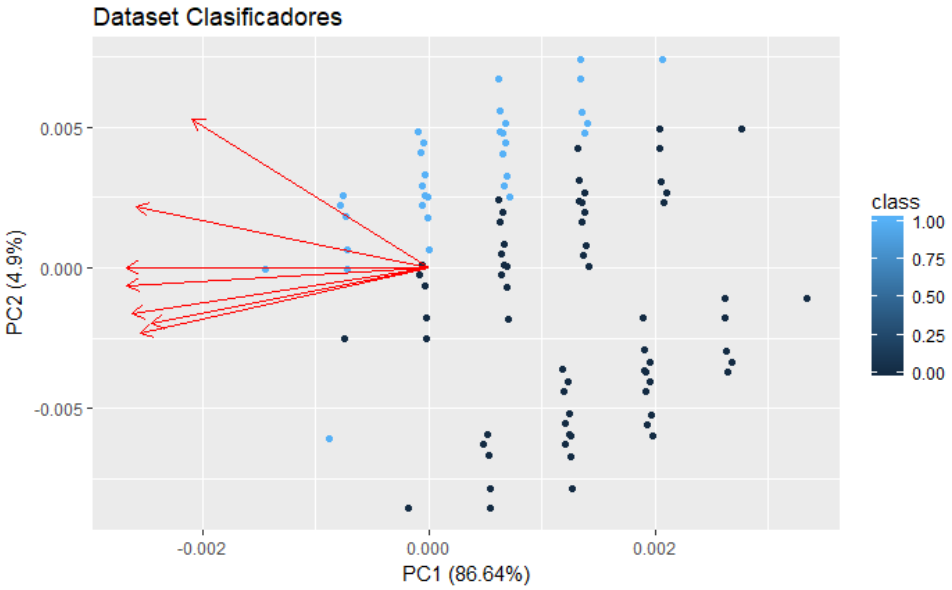

Figura 35: Visualización del dataset de predicciones sobre el plano de componentes principales

En este caso, dado que nuestro nuevo dataset se compone principalmente de instancias binarias (Salvo el caso de knn, donde mantenemos la predicción realizada), vemos que los datos se agrupan en múltiples clústeres de zona y se ve un ligero solape entre las clases, basado en la variación de la componente secundaria del conjunto de datos. En este caso

pasamos de una agrupación de la varianza del dataset sobre la primera componente principal de un 81.86% a un 86.64%.

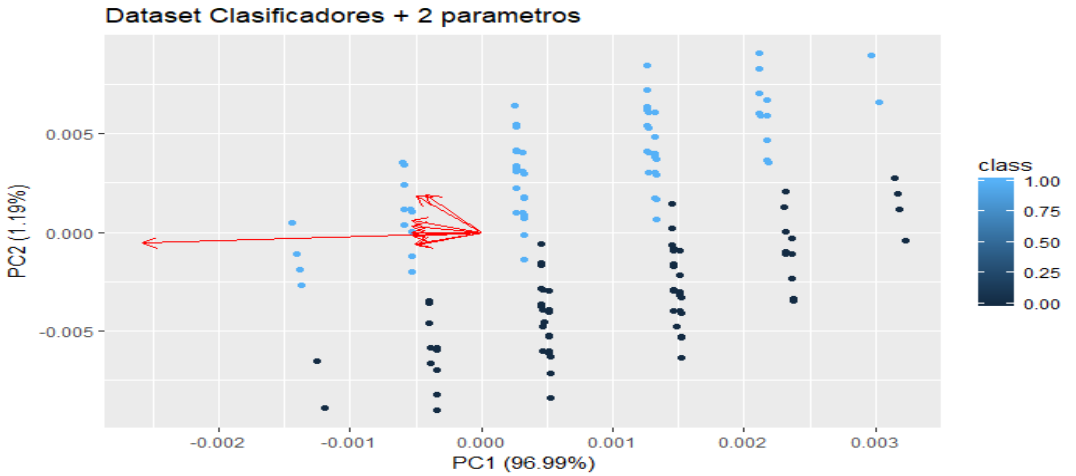

Figura 36: Visualización del dataset de predicciones sobre el plano de componentes principales incluyendo los atributos basados en heurísticas

Vemos una mejora significativa en la agrupación de los datos dado que antes de aplicar estas heurísticas había dos subconjuntos de cada clase, ahora hay únicamente 2 conjuntos principales en los que se aprecia una separación evidente entre las instancias. Finalmente, la componente principal supone en este caso un 96.99% de la varianza total.

4.10 Clasificador Final

Con el objetivo de hacer la clasificación final de nuestro conjunto de datos utilizaremos XGBoost. En primer lugar, utilizaremos un Split 80 – 20 del dataset para afinar los parámetros del modelo y aseguras así que no hay overfitting

4.10.1 Learning rate y número de iteraciones:

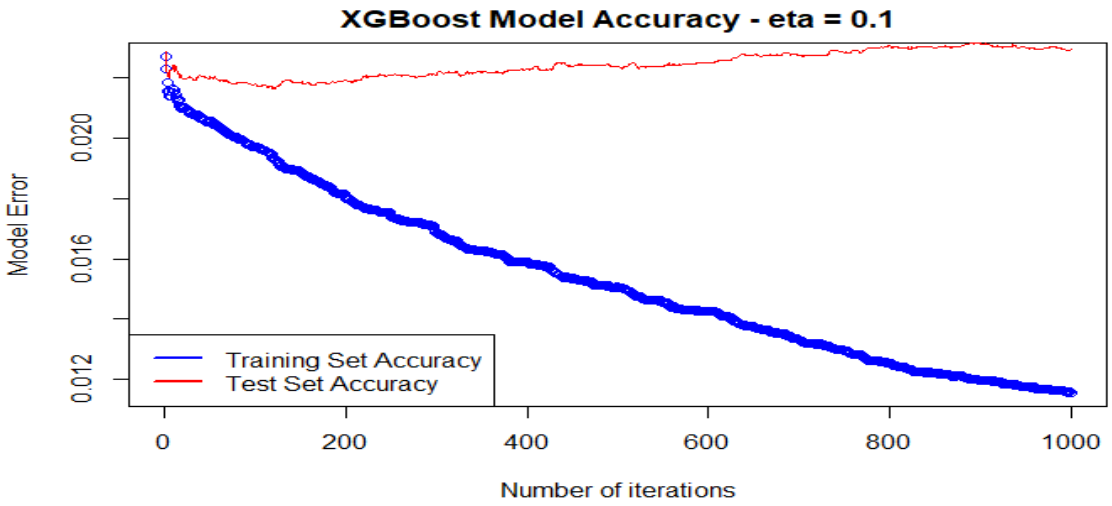

Figura 37: Error del modelo en training y test set con learning rate de 0.1

Vemos como se produce sobre-entrenamiento del modelo a partir de la iteración 150, en donde el modelo sigue aprendiendo de los datos del training set pero con cada iteración se produce un error mayor en el test set.

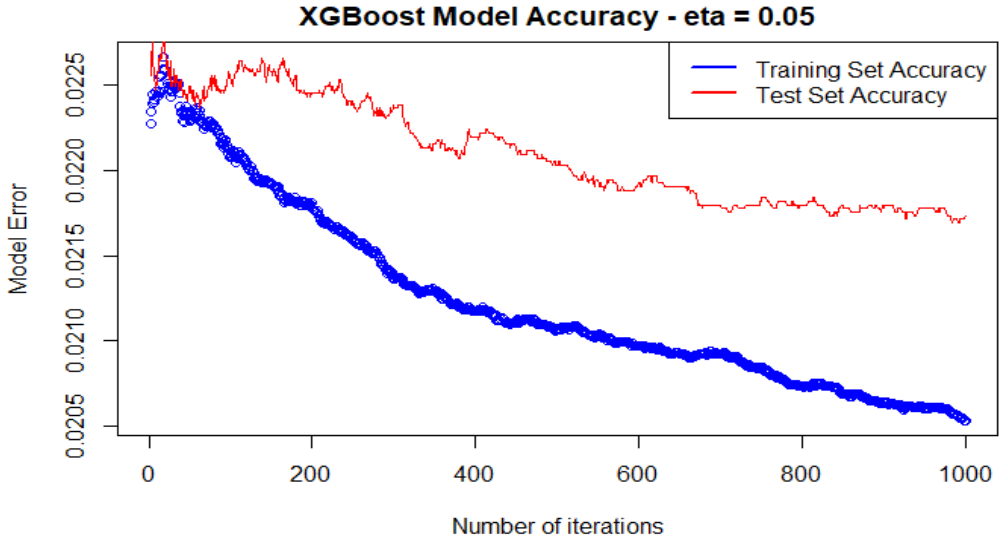

Figura 38: Error del modelo en training y test set con learning rate de 0.05

Con un ratio de aprendizaje de 0.05 vemos que no llega a producirse overfitting dado que el error producido en el test set es menor con cada iteración.

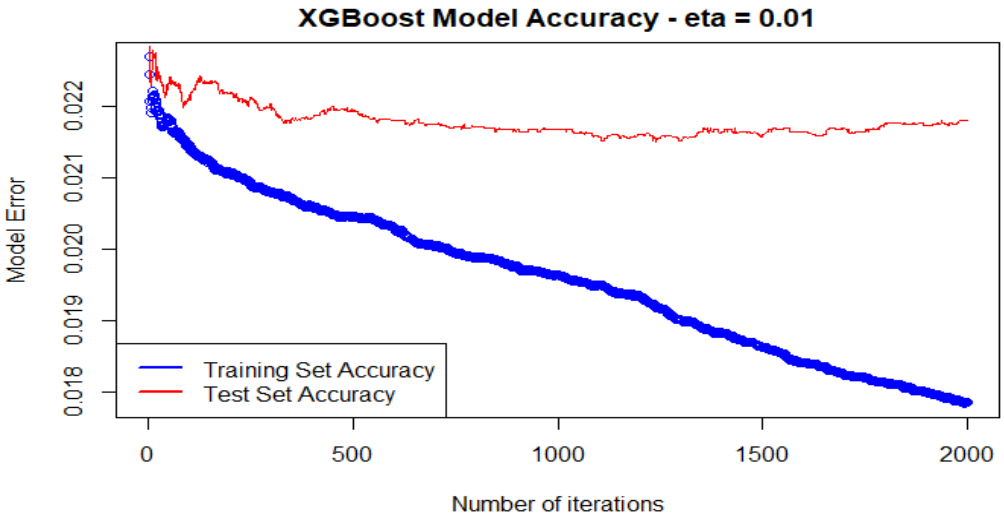

Figura 39: Error del modelo en training y test set con learning rate de 0.01

Con un ratio de aprendizaje de 0.01 se produce un aprendizaje mayor en escala pero no se aprecia ninguna mejora significativa del aprendizaje del test set por encima de la iteración número 1200.

Utilizamos para el modelo final una validación de 10 Fold Cross Validation con eta de 0.01 y 1200 iteraciones.

XGBoost Metalearner - Confusion Matrix and Statistics

```
             Actual
Prediction      0      1
         0  74741   1955
         1   3229 159422
```

ROC Area : 0.9974952
Accuracy : 0.9783411

```
239347 samples
    27 predictor
     2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Resampling results:

Tuning parameter 'max_depth' was held constant at a value of 8
Tuning parameter 'nrounds' was held constant at a value of 1200
Tuning parameter 'min_child_weight' was held constant at a value of 10
Tuning parameter 'subsample' was held constant at a value of 0.5
```

Tabla 21 – Resultados obtenidos por el modelo metalearner

Vemos que el nuevo modelo supera con creces los resultados de los modelos individuales tanto en Área bajo la curva ROC como en precisión de clasificación.

Atendiendo a la importancia de los clasificadores:

XGBoost Metalearner - Importance

	Overall
cat_p1	100.00000
xgb_p1	28.38094
knn_p1	12.58676
fe_suma	4.01589
gbm_h20_p1	0.57156
ada_p1	0.17685
adabag_p1	0.07046
fe_km	0.00000

Tabla 21 – Resultados obtenidos por el modelo metalearner

CATboost supone una gran parte de la predicción sobre la que se basa este modelo de metalearning, seguido de XGBoost. Por el contrario, el feature engineering realizado tiene un impacto muy pequeño, siendo nulo en el caso de KNN.

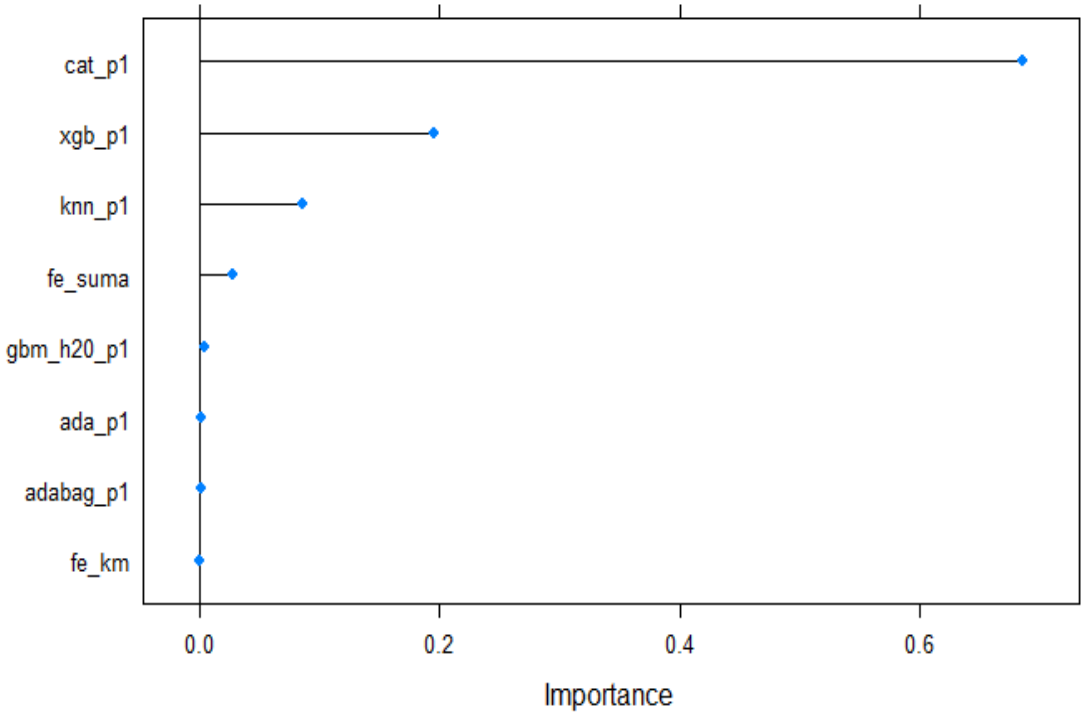

Figura 40: Representación gráfica de la importancia de los modelos en el modelo final

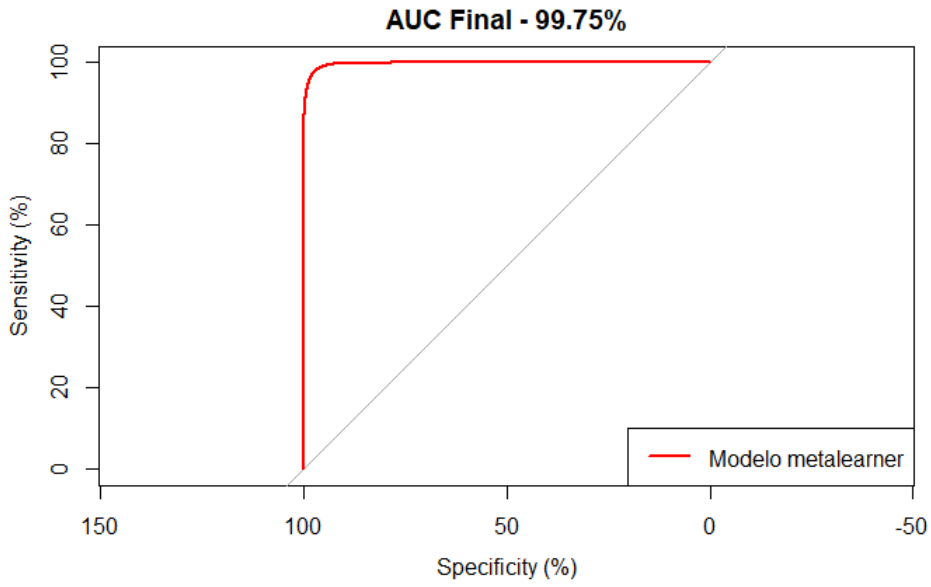

Figura 41: Curva ROC del modelo generado

Comparando los resultados obtenidos por el modelo final.

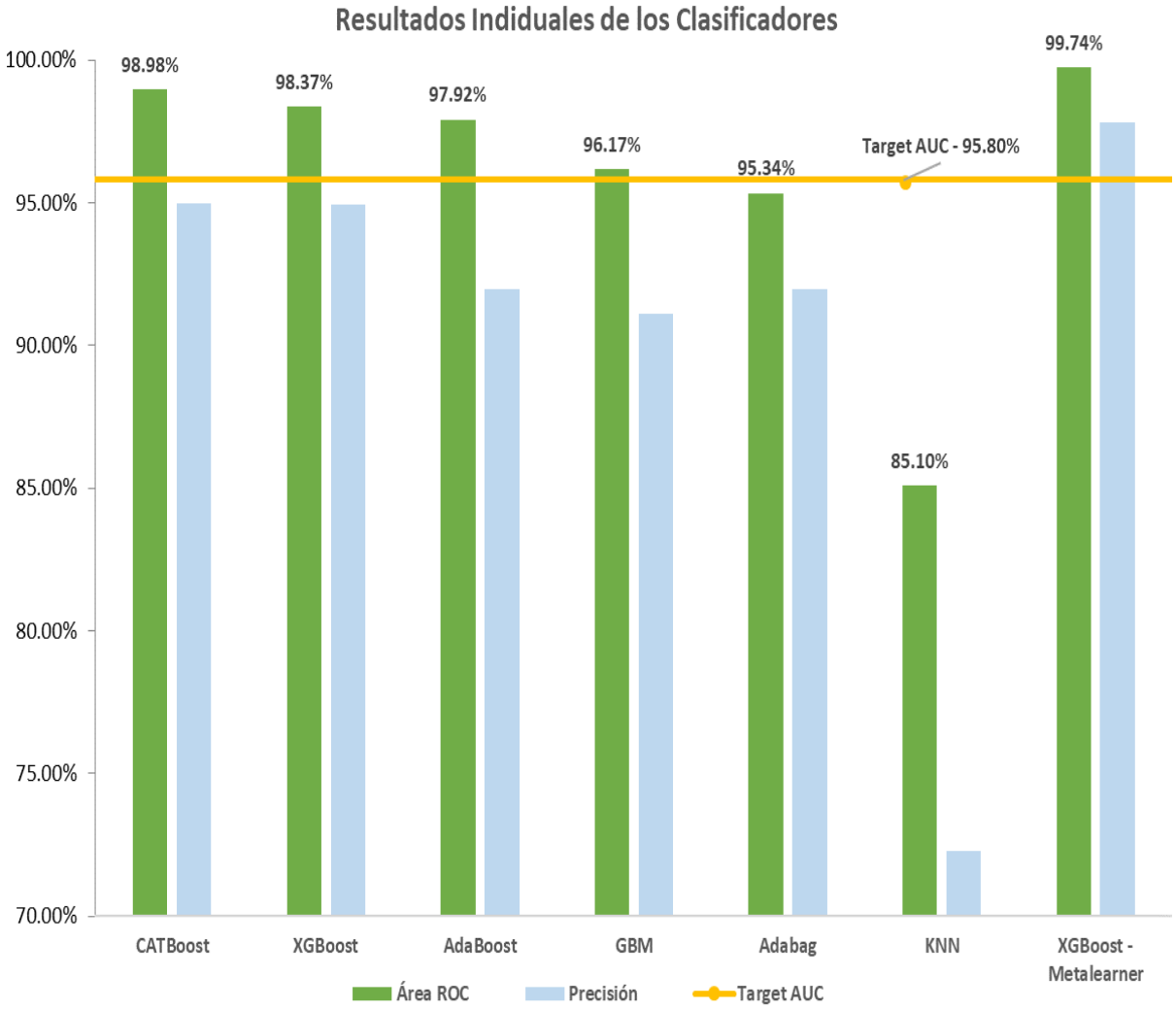

Figura 42: Área ROC y Precisión obtenida por los modelos empleados

5. RELACIÓN CON RESPECTO A LAS MATERIAS DEL MÁSTER

Si bien este trabajo tiene una aplicación directa sobre el análisis estadístico y de machine learning, hay una serie de conceptos e ideas de base aprendidos en el resto de las asignaturas que complementan el trabajo realizado.

Procedemos a explicar la relación del trabajo realizado con cada una de las materias cursadas en el máster.

Infraestructuras de computación

En la clase de infraestructuras de computación se hizo un estudio de las distintas infraestructuras existentes que sirven de base para la gestión de equipos de Tecnologías de Información y Comunicación. En este caso concreto nos ha servido para poder medir valorar los métodos computacionales empleados en base a la potencia de nuestra máquina. Adicionalmente, estos estudios se basan en las mediciones hechas por el detector de partículas LHCb, que finalmente es una máquina que recoge señales basados en procesos físicos y que transforma esas señales en datos.

Estructuras de datos y su almacenamiento

En la clase de estructuras de datos y su almacenamiento se estudia desde un aspecto de infraestructura y de procesos las diferentes metodologías de almacenamiento y manipulación de los datos, así como características inherentes de los datos como la Veracidad, Velocidad, Volumen o Variedad.

En este ejercicio hemos operado con unos datos estructurados generados por el LHCb, los cuáles no he tenido que extraer de ninguna base de datos pues se me han sido entregados en un archivo .dat. No obstante estos datos figuran en la base de datos del CERN, la cual, muy probablemente, tendrá una base de datos estructurados para guardar cada una de las mediciones generadas y asegurar así las cuatro V descritas anteriormente.

Herramientas de Análisis y Visualización de datos

En la asignatura de Herramientas de Análisis y Visualización de datos se estudia cómo hacer un análisis exploratorio de un conjunto de datos, así como los conocimientos estadísticos y probabilísticos necesarios para no correr riesgos en presunciones generalistas, además de conocimientos de visualización de los datos.

En este trabajo utilizamos conceptos como histogramas de frecuencias, manipulación de datos con el lenguaje R, análisis de componentes principales y representaciones gráficas en figuras para explicar los resultados obtenidos por nuestro análisis.

Sistemas basados en el conocimiento

En la asignatura de sistemas basados en el conocimiento se estudian conceptos de introducción a la inteligencia artificial además de estructuras globales de conocimiento, cómo acceder a ellas y como operar con ellas.

En este estudio aplicamos conceptos básicos como el uso de reglas heurísticas en nuestros algoritmos para mejorar los resultados minimizando tiempos computacionales

Pensamiento y creatividad. Aspectos éticos y legales de los datos

En la asignatura de Pensamiento y creatividad. Aspectos éticos y legales de los datos se estudia la implicación social del uso de herramientas basadas en datos desde el propio diseño, así como la normativa legal de uso, recolección y mantenimiento del dato.

En este ejercicio no hacemos un énfasis específico sobre la ética o legalidad de nuestros datos pues estos representan un estudio científico no representativo de ningún ser vivo.

Datos y negocio

En la asignatura de datos y negocio se estudia el poder del dato con un ámbito comercial-empresarial, además de cómo se puede plantear una estrategia de negocio basada en los datos.

Sobre la redacción de este Trabajo de Final de Máster no se aplican los conocimientos adquiridos en esta asignatura, pero sí habrá una aplicación directa del conocimiento en la defensa del trabajo en base a las masterclass sobre presentaciones eficaces.

Inteligencia Artificial para la ciencia de los datos

La asignatura de Inteligencia Artificial para la ciencia de los datos es la asignatura sobre la cuál se cimenta el trabajo realizado. En la asignatura se enseñan conceptos básicos de inteligencia artificial y los algoritmos más básicos empleados como KNN o Decision Trees y que sirven de base para los conceptos más avanzados como Bagging, Boosting o algoritmos complejos como Redes Neuronales.

En este ejercicio hemos utilizado conceptos como train set y test set, N Fold Cross Validation, Decision Trees, Boosted Trees Methods incluyendo los más eficaces actualmente como CATBoost o XGBoost, Feature Engineering, Bagging, Área Bajo la Curva ROC y Matriz de Confusión.

Herramientas avanzadas de análisis de datos

En la asignatura de Herramientas Avanzadas de Análisis de Datos se prolonga el conocimiento adquirido en Herramientas de Análisis y Visualización de datos y se entremezcla con Inteligencia Artificial para la ciencia de los datos, haciendo énfasis en conceptos como Naive Bayes o Métodos de Monte Carlo.

Hemos aplicado los conceptos numéricos y estadísticos enseñados en esta asignatura pero además nuestro modelo clasificador se basa en un conjunto de datos generados artificialmente por métodos de Monte Carlo, habiendo una aplicación directa sobre el trabajo.

Business Intelligence

La asignatura de Business Intelligence se complementa con la formación recibida en datos y negocio, pero se centra más en los números propios de la empresa con un carácter operacional en la que se estudian conceptos como ROI o Total Cost of Ownership.

Esta asignatura no tiene aplicación directa alguna sobre el trabajo realizado, pero sirve como base para entender los modelos de negocio públicos como el CERN.

Smart Cities

La asignatura de Smart Cities aporta una visión transversal del concepto de Smart Cities y sobre las aplicaciones de la ciudad digital, cubriendo aspectos teóricos, de aplicación ciudadana, de calidad ambiental o de gestión de infraestructura.

Esta asignatura no tiene aplicación directa alguna sobre el trabajo realizado, más que haber empleado técnicas de análisis de datos similares a la empleadas en la asignatura.

6. CONCLUSIONES, LÍNEAS ABIERTAS Y VALORACIÓN ECONÓMICA

6.1 Conclusiones

Del trabajo realizado sacamos conclusiones muy positivas pues hemos conseguido mejorar los resultados obtenidos originalmente en la tesis doctoral con 4 clasificadores individuales, únicamente utilizando algoritmos de clasificación similares pero más avanzados a los utilizados y afinando los parámetros para maximizar los resultados rápidamente y asegurando que los distintos modelos generados respetan estadísticamente la fiabilidad de la predicción sin caer en un posible sobre-entrenamiento. Como premio individual identificamos el algoritmo CATBoost que mejora los resultados un 3.18% sobre el clasificador de referencia, obteniendo un área final de 98.98%.

Adicionalmente se realiza un estudio de feature engineering y metalearning que permite agregar los resultados individuales obtenidos por los clasificadores individuales, ya que cada uno de ellos tiene características de aprendizaje distintas y se basan en distintos atributos para lograr la clasificación final, con el cual se consiguen maximizar los resultados obtenidos y dejando muy poco margen de mejora.

Como conclusiones obtenidas de las características del trabajo realizado:

- Los modelos de Gradient Boosting Trees funcionan muy bien dentro del dominio de desintegración de partículas y con el conjunto de datos utilizado.
- Los métodos de árboles aleatorios son susceptibles al ruido de la muestra y generan resultados demasiado adaptados a la muestra, logrando un resultado pobre en la aplicación del modelo sobre otras muestras de datos.
- KNN no cumple como clasificador individual pero agregado con otros clasificadores ayuda a mantener las características numéricas del dataset y sobre el clasificador final.
- KNN no aporta nada como atributo agregado para el metalearner.
- H20 es una tecnologia de Ineligencia Artificial muy potente dado que optimiza al máximo los recursos disponibles de la máquina por encima de los obtenidos por R

6.2 Líneas Abiertas

Si bien los resultados obtenidos en este ejercicio son muy positivos hay unas líneas de trabajo concretas que pueden ayudar a mejorar los resultados obtenidos, sobre todo desde un aspecto computacional.

Selección de atributos para el entrenamiento del modelo:

Actualmente utilizamos 28 atributos y una clase y si bien descartamos 5 atributos inicialmente, se aprecia por la importancia de los atributos en cada uno de los modelos obtenidos que hay una serie de variables que no son utilizadas. Profundizar en el estudio de estas variables que no aportan información al modelo puede reducir los costes computacionales del modelo sin arriesgar significativamente los resultados obtenidos, así como la reducción de variables correlacionadas puede mejorar los resultados obtenidos por los modelos que no se basan en Gradient Boosting.

Modificación numérica de las variables empleadas

Del análisis exploratorio de datos apreciamos que hay determinadas variables que necesitamos aplicar una función logarítmica para visualizar correctamente la separación entre clases. Aplicar estas manipulaciones puede ayudar a que el modelo distinga mejor las clases pues transformamos una variable de carácter exponencial a una con carácter lineal, mejorando también los tiempos de cálculo.

Potencialmente se podría hacer un análisis de discriminante lineal (LDA) para reducir la magnitud del problema de 28 atributos a un número mucho más reducido y obteniendo unos resultados similares. Al hacer esto no obstante se pierde conocimiento del dominio pues pasamos de atributos con un significado físico a dimensiones numéricas agregadas.

Mejora de los atributos artificiales generados en el proceso de feature engineering

En este ejercicio únicamente empleamos dos parámetros y uno de ellos no tuvo ningún aporte sobre el modelo. Otro tipo de atributos artificiales empleados en dominios similares al estudiado pueden ser:

- Distancia mínima entre instancias de la misma clase
- Distancia mínima de cada instancia a las dos instancias más cercanas de la misma clase.
- Distancia mínima de cada instancia a las cuatro instancias más cercanas de la misma clase.

Testing del modelo con datos reales

En este ejercicio únicamente hemos entrenado y testeado el modelo con datos reales de los cuáles únicamente hemos seleccionado los que conocemos como background combinatorio de la muestra y datos artificiales generados con métodos de Monte Carlo.

Para asegurar la robustez del modelo se puede testear con datos representativos de los cuáles conocemos la clase, dado que en el dataset de muestras reales que tenemos no tenemos la certeza de clasificación entre una señal positiva y una señal de ruido combinatorio.

Bagging del modelo generado con otros modelos construidos para otras desintegraciones

El modelo que hemos generado ha sido entrenado únicamente para clasificar la desintegración del Mesón B^0 en un mesón K^{*0} y en un fotón γ que a su vez, el mesón K^{*0} se reconstruye a partir de un pion y un kaon, $K^+\pi^-$ o $K^-\pi^+$, pero esta desintegración es una desintegración entre muchas otras descritas. Potencialmente podría realizarse un modelo como el hecho hasta ahora con cada proceso de desintegración conocido sobre el que se hagan test para agregarlos todos en un modelo único que sea capaz de identificar con precisión qué tipo de desintegración se utiliza.

6.3 Valoración Económica del Trabajo

A continuación incluimos un desglose económico de la implementación del proyecto de Data Science descrito incluyendo partidas horarias individuales:

Valoración económica				
Concepto	**Descripción**	**Precio (€/h)**	**Cantidad (h)**	**Subtotal**
1. Analista de Datos				
1.1 Recopilación de información	Incluye tiempo dedicado a investigación del dominio de análisis, reuniones con responsables de departamento y parte proporcional de costes de transporte	€ 35.00	30.00	€ 1,050.00
1.2 Análisis exploratorio de datos	Incluye tiempo dedicado al análisis de los datos recopilados así como parte proporcional de mantenimiento de licencias de análisis de datos	€ 35.00	40.00	€ 1,400.00
1.3 Elaboración de informe y presentación de resultados	Incluye tiempo dedicado a redacción de informa y parte proporcional de costes materiales y de transporte	€ 35.00	20.00	€ 700.00
			90.00	**€ 3,150.00**
2. Científico de Datos				
2.1 Análisis de modelos	Incluye tiempo dedicado al análisis de los posibles métodos de machine learning empleados	€ 50.00	30.00	€ 1,500.00
2.2 Implementación de modelos	Incluye tiempo dedicado a codificar el modelo apropiado, afinación de los mismos, optimización de resultados y confirmación de resultados con cliente	€ 50.00	80.00	€ 4,000.00
2.3 Implementación de modelo final	Incluye tiempo dedicado a automatizar y documentar el modelo final implementado	€ 50.00	50.00	€ 2,500.00
			160.00	**€ 8,000.00**
Total				**€ 11,150.00**

Tabla 22 – Valoración Económica del Proyecto

7. BIBLIOGRAFÍA

Intro:

[1] Modelo estándar: https://es.wikipedia.org/wiki/Modelo_est%C3%A1ndar_de_la_f%C3%ADsica_de_part%C3%ADculas
[2] Meson B: https://es.wikipedia.org/wiki/Mes%C3%B3n_B
[3] LHCb: https://es.wikipedia.org/wiki/LHCb
[4] Hadron: https://en.wikipedia.org/wiki/Hadron
[5] Tesis: Study of b-hadron decays into two hadrons and a photon at LHCb and first observation of b-baryon radiative decays

Señal:

[6] IFCA: https://ifca.unican.es/en-us/education-and-outreach/masterclasses-particle-physics
[7] Specificy and Sensitivity: https://en.wikipedia.org/wiki/Sensitivity_and_specificity
[8] ROC: https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Machine Learning Models:

[9] R Project: https://www.r-project.org/
[10] R Studio: https://www.rstudio.com/
[11] Jupyter: https://jupyter.org/
[12] Caret: http://topepo.github.io/caret/index.html
[13] H2O.ai: https://www.h2o.ai/gartner-magic-quadrant/
[14] Cuadrante mágico de Gartner en herramientas de Data Science y Machine Learning (2018): https://www.gartner.com/doc/reprints?id=1-4RMUF0K&ct=180222&st=sb
[15] XGBoost: http://xgboost.readthedocs.io/en/latest/
[16] XGBoost aplicado en el Higgs Challenge: https://higgsml.lal.in2p3.fr/prizes-and-award/award/
[17] XGBoost explicado en el CERN: https://cds.cern.ch/record/2017394?ln=de
[18] CATBoost https://tech.yandex.com/catboost/
[19] CATBoost: https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db
[20] Adaboost: https://en.wikipedia.org/wiki/AdaBoost
[21] GBM: https://en.wikipedia.org/wiki/Gradient_boosting
[22] Adabag: https://cran.r-project.org/web/packages/adabag/adabag.pdf
[23] KNN: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
[24] Treebag: http://datatechnotes.blogspot.com/2018/04/classification-with-bagging-treebag.html
[25] Random Forest: https://en.wikipedia.org/wiki/Random_forest