

Treball de Fi de Grau en Física

Estudio de la extracción de la señal resonante
de sucesos $t\bar{t}$ mediante técnicas de Machine
Learning en el experimento ATLAS

JUNIO 2022

Alumno: Azael García Rodríguez

Tutor (1): Santiago González de la Hoz
Tutor (2): José Francisco Salt Cairols

Resumen

El objetivo de este trabajo es hacer uso de diversas técnicas de *machine learning* para poder diferenciar de forma más eficaz la resonancia de los quarks $t\bar{t}$ producidos en una colisión protón-protón. La colisión pp con energía de centro de masa 8 TeV se encuadra en el estudio de física de altas energías que lleva a cabo el detector ATLAS construido en el colisionador LHC. Las partículas depositan su energía en la materia del detector, producto de la colisión, y a partir de estos se intenta averiguar si ha podido existir una partícula X de masa indeterminada cuya desintegración es la que da lugar al par de quarks top-antitop, que clasificaremos como *señal*. Esto conformaría parte de la búsqueda de nueva física *Beyond the Standard Model* (BSM).

Para lograr esto, se hace uso de técnicas de aprendizaje supervisado usando el lenguaje R y trabajando en el entorno RStudio. Se han utilizado *Decision Trees* (DT) y métodos más avanzados basados en el uso de DTs tales como *Random Forest* (RF) y *Boosted Decision Trees* (BDT), concretamente utilizando el algoritmo *Adaptive Boosting*. Para entrenar nuestros modelos, se hace uso de datos generados por simulaciones de las colisiones por Montecarlo encontrados en el repositorio HEPMASS. Se pueden encontrar diversos *datasets* y se usará uno con los datos sin procesar para analizar las variables físicas del problema y los otros se encuentran con los datos normalizados para entrenar y evaluar correctamente nuestros modelos.

Abstract

The aim of this work is to make use of various machine learning techniques to differentiate more efficiently the resonance of the $t\bar{t}$ quarks produced in a proton-proton collision. The pp collision with a center-of-mass energy of 8 TeV is part of the study of high energy physics carried out by the ATLAS detector built at the LHC collider. The particles leave their energy in the detector, product of the collision, and from these we try to find out if there could have been an X particle of undetermined mass whose decay is the one that creates the top-antitop quark pair, which we will classify as signal. This will be part of the search for new physics Beyond the Standard Model (BSM).

To achieve this goal, we make use of supervised learning techniques using the R language and working in the RStudio environment. We have used Decision Trees (DT) and more advanced methods based on the use of DTs such as Random Forest (RF) and Boosted Decision Trees (BDT) specifically using the Adaptive Boosting algorithm. In order to train our models, we make use of data generated by Montecarlo collision simulations found in the HEPMASS repository. Various datasets can be found there and we will use one with raw data to analyze the physical variables of the problem and the others will be found with the normalized data to correctly train and evaluate our models.

Índice

1. Introducción al problema	3
1.1. Modelo Estándar y sus limitaciones	3
1.2. Colisión pp en el LHC	3
1.2.1. Colisiones pp en SM y BSM	4
1.3. Medición de los productos de la colisión: Los $jets$	5
1.4. Colisionador LHC y detector ATLAS	5
2. Análisis de los datos HEPMASS	7
2.1. Variables de los $datasets$	7
2.2. Distribución de las variables del $dataset$	8
2.3. Matrices de correlación de las variables	10
3. Modelos de clasificación binaria haciendo uso de distintos algoritmos de <i>Machine Learning</i>.	11
3.1. <i>Decision Trees</i>	11
3.2. <i>Boosted Decision Trees</i>	13
3.3. <i>Random Forest</i>	14
4. Métricas del rendimiento de los modelos	15
4.1. <i>Accuracy</i>	16
4.2. <i>Kappa de Cohen</i>	16
4.3. <i>Sensitivity</i>	17
4.4. <i>Specificity</i>	17
4.5. <i>F1-Score</i>	17
4.6. AUC (<i>Area Under the Curve</i>)	17
5. Resultados	17
5.1. <i>Decision Tree</i>	17
5.2. <i>Boosted Decision Tree</i> con AdaBoost	20
5.3. <i>Random Forest</i>	23
6. Conclusiones	24
6.1. <i>Conclusions</i>	25
Referencias.	25

1. Introducción al problema

Para entender uno de los objetivos de la nueva física existente más allá del SM, tales como la extracción de la señal resonante $t\bar{t}$ sobre un fondo de sucesos SM, hay que comprender que es el modelo estándar y explicar que ocurre en una colisión pp con energía en el centro de masa con valor $\sqrt{s} = 8\text{TeV}$ en el experimento ATLAS del LHC.

1.1. Modelo Estándar y sus limitaciones

El modelo estándar (SM) es la teoría de la física de partículas que clasifica todas las partículas elementales conocidas y las interacciones existentes entre ellas. Es una teoría cuántica de campos que recoge otras relacionadas como la teoría electrodébil, la cromodinámica cuántica y la electrodinámica cuántica. Puede verse su clasificación en la figura 1. El modelo estándar tiene limitaciones [7] para poder explicar correctamente ciertos sucesos, por ejemplo la fuerza gravitacional queda fuera del modelo, no explica la existencia de la materia oscura, pues ninguna partícula del SM tiene indicios de ser la componente de este tipo de materia, no incorpora las oscilaciones de neutrinos ni que sus masas sean distintas de cero, etc. Por eso se buscan modelos de nueva física de partículas que a partir del SM que incorporen partículas exóticas y sus interacciones.

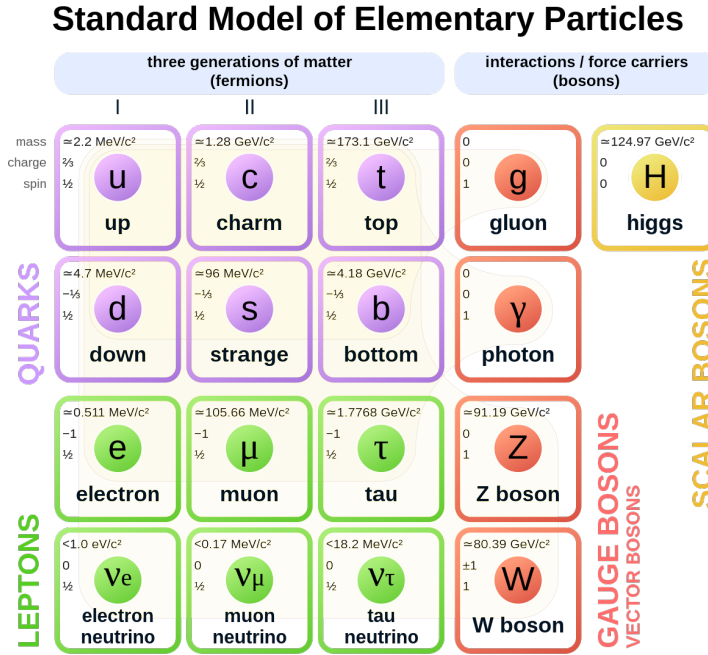


Figura 1: Partículas elementales del SM

1.2. Colisión pp en el LHC

Fijándonos ahora en el caso de una colisión pp en el LHC que da lugar a quarks $t\bar{t}$, el proceso que tiene lugar puede explicarse tanto dentro del marco SM como del BSM. Los protones son hadrones conformados por quarks *up* y uno *down* que se encuentran rodeados de otras parejas transitorias de quarks y antiquarks y gluones, siendo estos últimos los

encargados de la interacción fuerte entre quarks que convierte el protón en una partícula estable.

En el LHC se aceleran protones con una velocidad similar a la de la luz y chocan a altas energías. De dicha colisión surgen una serie de quarks y desintegraciones que dan lugar a nuevas partículas. Esto es debido a que los quarks y los gluones de ambos protones interaccionan entre ellos al encontrarse lo suficientemente cerca y crean quarks diferentes a los iniciales y otros gluones. Según el SM, el acoplamiento entre los dos gluones generan la desintegración $t\bar{t}$, cuyo quark t se desintegra a su vez en un quark *bottom* y un bosón W^+ , que a su vez da lugar a por ejemplo un leptón muónico o electrónico y un neutrino. Por su parte el quark b se desintegra rápidamente en una serie de partículas que genera un *jet*. Por contrapartida, el quark \bar{t} se desintegra en un quark \bar{b} que emite un *jet* y un bosón W^- que se desintegra hadrónicamente en dos quarks $q\bar{q}$ que emiten sus correspondientes *jets*.

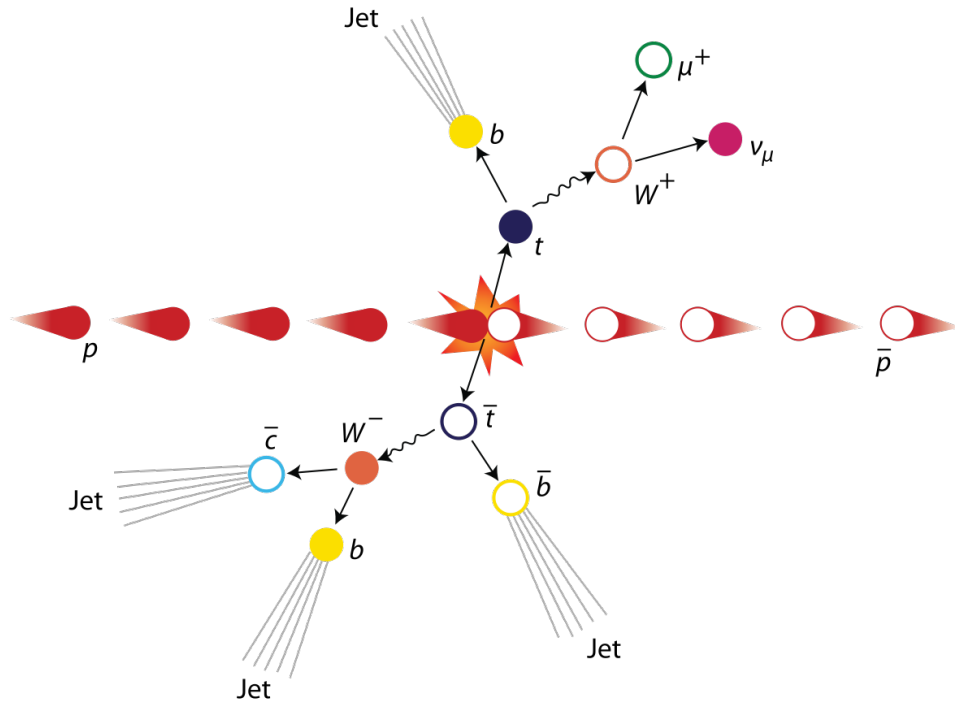


Figura 2: Colisión pp , formación del par de quarks $t\bar{t}$ y los productos de su desintegración

1.2.1. Colisiones pp en SM y BSM

Un suceso posible que se encuadra en el marco BSM es que justo después de la colisión pp se genere una resonancia que da lugar a una partícula exótica X de masa desconocida [12], que sale a partir de las teorías de Kaluza-Klein y posible opción a ser componente de la materia oscura, que es la que posteriormente se desintegra en el par $t\bar{t}$. Las colisiones con resultados dentro del SM como se consideran conocidos los calificamos de *background* y los sucesos BSM los trataremos como *signal*.

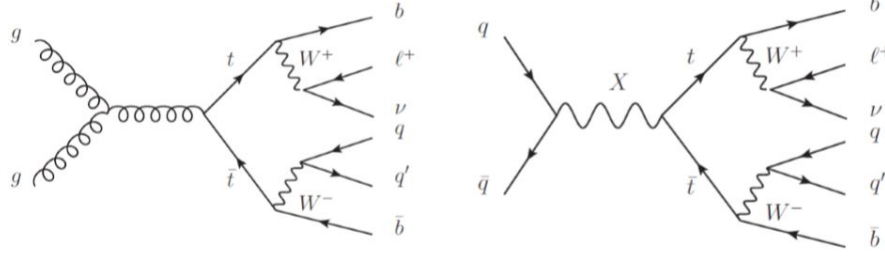


Figura 3: Diagrama de Feynman de la colisión pp descrito como suceso encuadrado en el SM (izquierda) y en el BSM (derecha)

1.3. Medición de los productos de la colisión: Los *jets*

La interacción fuerte entre quarks viene explicada por la teoría de la cromodinámica cuántica (QCD), que como se ha explicado previamente, en los hadrones se encuentran confinados gluones y quarks [16]. Los quarks no pueden existir libremente sino que generan otros que son pareja y forman bariones o mesones. Cuando se forman en la colisión un par $q\bar{q}$ que se alejan, para mantener su color emiten gluones y más parejas de quarks que se recombinan y emiten partículas neutras o cargadas, formando haces de hadrones que dejan *jets* que son las trazas detectables por ATLAS.

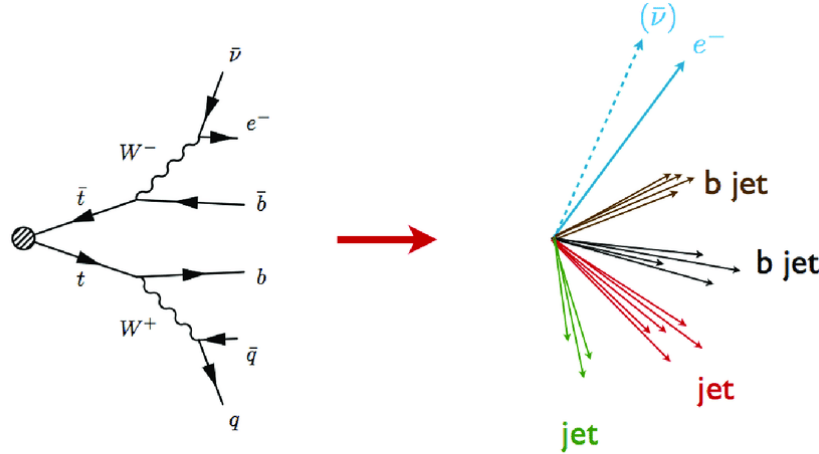


Figura 4: Los cuatro *jets* producidos por las trazas de las partículas generadas tras la desintegración $t\bar{t}$

1.4. Colisionador LHC y detector ATLAS

El *Large Hadron Collider* (LHC) es un acelerador de partículas del CERN [5]. Se trata de un anillo de 27 km de recorrido formado por electroimanes superconductores que guían las partículas y contiene diversos aceleradores. Una de las diversas funciones que tiene el LHC es encontrar nuevas partículas acelerando haces de partículas a velocidades similares a la de la luz y haciendo que colisionen a altas energías (del orden del TeV). Realmente,

antes de llegar al LHC, los haces ya han pasado por varios aceleradores, siendo el LHC el último de estos. Estas colisiones dan lugar a cientos de nuevas partículas que son detectadas por diversos detectores situados a lo largo del anillo, tales como CMS, ALICE, LHCb y ATLAS, siendo ATLAS uno de los más importantes.

A *large Toroidal LHC Apparatus* o ATLAS es un detector diseñado con el propósito de medir todo tipo de señales con el objetivo de encontrar nuevas partículas. El detector está formado por una serie de cilindros concéntricos, pues en cada capa cilíndrica se detectan ciertos tipos de partículas, excepto los neutrinos, que no interactúan con los detectores debido a que apenas interacciona con la materia. Aún así, puede deducirse a partir del momento restante aplicando conservación del momento en la colisión. Las distintas capas del detector interactúan con los productos de una colisión pp de forma que el detector interno interactúa con las partículas cargadas y cuyo punto inicial de trayectoria da información del tipo de partícula y puede realizar un etiquetado *btag*. Por su parte, los calorímetros situados fuera del imán solenoidal son los encargados de detectar los leptones (calorímetro electromagnético interno) y los quarks (calorímetro hadrónico externo, capaz de absorber la energía de las partículas que atraviesan el calorímetro EM sin interactuar, pero que interactúan a través de la fuerza fuerte).

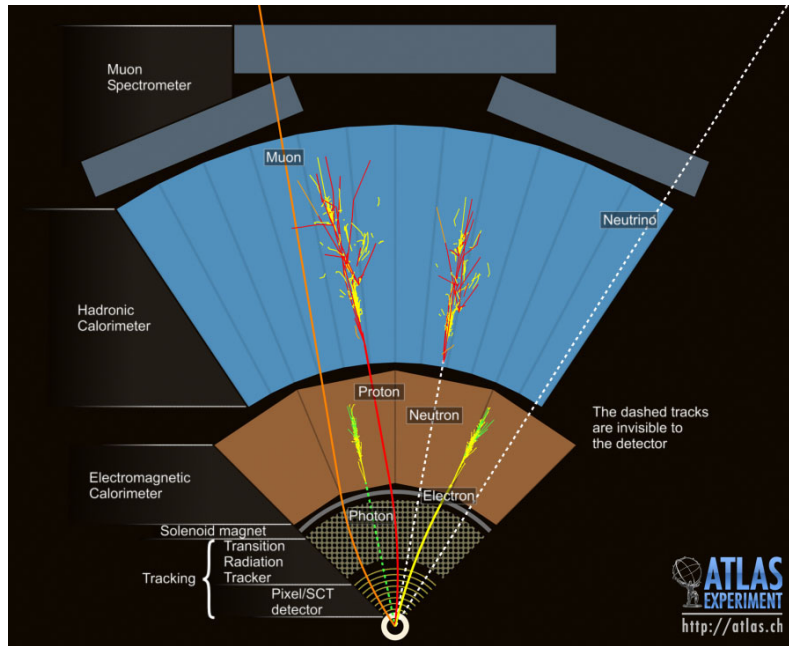


Figura 5: Detector ATLAS, las capas que lo componen y las partículas que detecta cada capa

Los haces de protones al colisionar en una dirección concreta del espacio, hacen que los productos de la desintegración $t\bar{t}$ se distribuyan en un plano transversal que corta las diversas capas cilíndricas del detector, por lo que trabajar en coordenadas cilíndricas es más práctico. Se miden el momento transversal p_T , el ángulo azimutal ϕ y la *pseudorapidity* η , una magnitud invariante Lorentz (cuyo valor es $\eta = -\ln(\tan(\frac{\theta}{2}))$ siendo θ el valor del ángulo polar), de los 4 *jets* resultantes, del leptón y del neutrino.

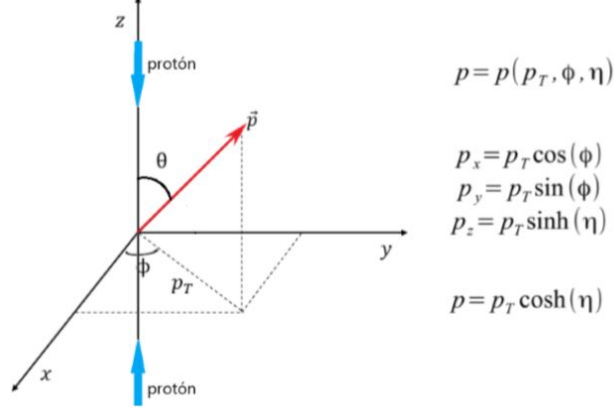


Figura 6: Esquema de coordenadas cilíndricas de las variables cinemáticas de los productos de la colisión pp

2. Análisis de los datos HEPMASS

Para poder entrenar los modelos de *machine learning* que se usarán posteriormente, se requiere de varios conjuntos de datos modificados específicamente para este propósito. Se hará uso de los datos de colisiones pp simuladas HEPMASS recogidos en el repositorio *UC Irvine Machine Learning Repository* [3].

Los diversos *datasets* empleados son generados usando simulaciones de Montecarlo. Dado que nuestro objetivo es poder discernir de sucesos (también llamados *observables*) SM (ruido) o BSM (señal), cada tanda de datos tiene asociado un valor booleano de '0' si el suceso generado se clasifica como ruido o de '1' si el suceso se clasifica como señal. Se disponen de varios *datasets* para cada masa de la resonancia $t\bar{t}$, $m = 500$ GeV, 750 GeV, 1000 GeV, 1250 GeV y 1500 GeV. También se disponen de otros *datasets*, como uno que solo contiene datos de ruido o *background*. Hay otros que contienen los sucesos de distintas masas combinados en un mismo *dataset* que puede resultar útil para entrenar modelos de clasificación, más relacionados con redes neuronales.

A su vez, estos *datasets* se dividen en unos que contienen las variables *raw* y otros normalizados que se usarán para entrenar nuestros modelos. Los datos *raw* contienen los valores físicos del proceso y son muy útiles para realizar un análisis exploratorio de los datos. A su vez, a los datos normalizados se les ha transformado ciertas variables tales como los momentos transversales y las masas invariantes con $\log(x + 10^{-5})$.

2.1. Variables de los *datasets*

En todos los *datasets* hay 28 variables (27 si el *dataset* contiene solo observables para una masa específica). Nos referimos como variables del problema a las distintas magnitudes físicas recogidas en los *datasets*, propias de una colisión pp , tales como los momentos transversales de las distintas partículas, ángulos azimutales de estas etc. Estas variables se pueden clasificar como un *label* que identifica el suceso como ruido o señal, 22 variables *low level* y 5 *high level* y en algunos casos, otra variable que contiene la masa de la resonancia. Las variables *high level* realmente se tratan de reconstrucciones hechas a partir de las *low level* que son las que recogen la información de las partículas resultado de la colisión pp y que se han desintegrado.

Dentro de las variables *Low level* encontramos:

- Momentos transversales de los 4 jets más energéticos.
- Momento transversal del neutrino, el cual se deduce por conservación.
- Momento transversal del leptón.
- Azimutal ϕ de los 4 jets, del leptón y del neutrino.

Y dentro de las *High level* se encuentran las masas invariantes: M_{jj} , M_{jjj} , M_{lv} , M_{jlv} y M_{wwbb} .

2.2. Distribución de las variables del *dataset*

Es de gran utilidad representar histogramas de la frecuencia de aparición que tienen las variables en función de la energía y como evoluciona según aumenta la masa de la partícula X.

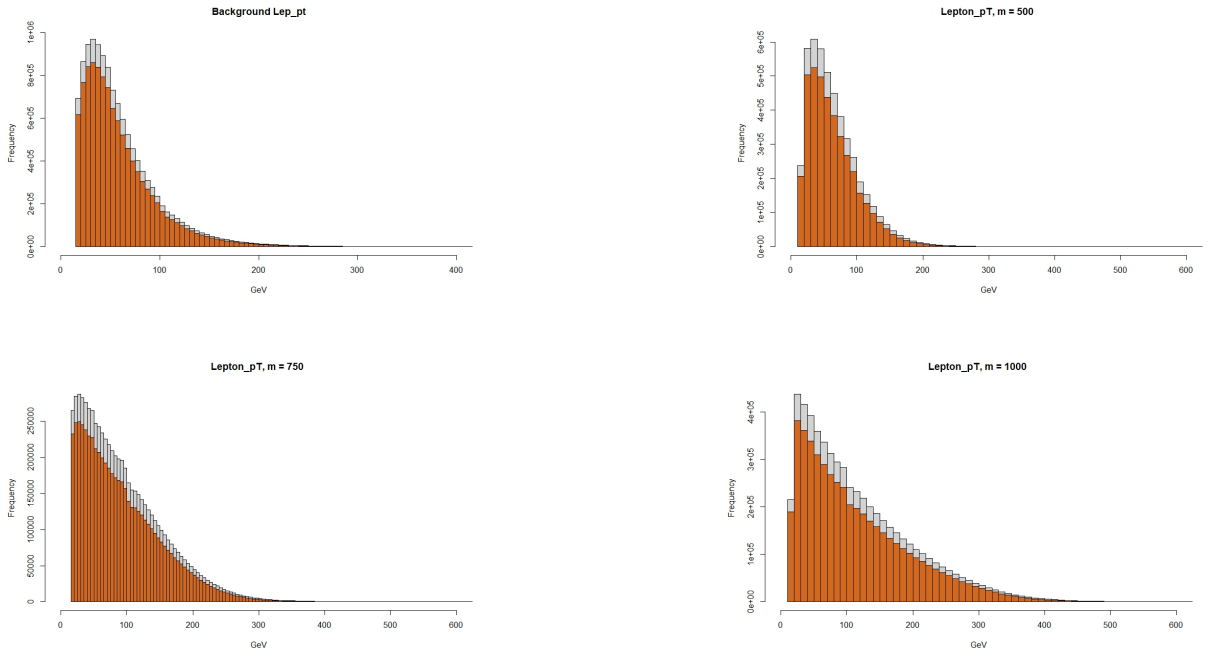




Figura 7: Distribución del momento transversal del leptón y en los *datasets* con todas las masas. El blanco corresponde a todos los sucesos y el marrón corresponde a los sucesos en los cuales se etiquetó a 2 *jets* como *btag*

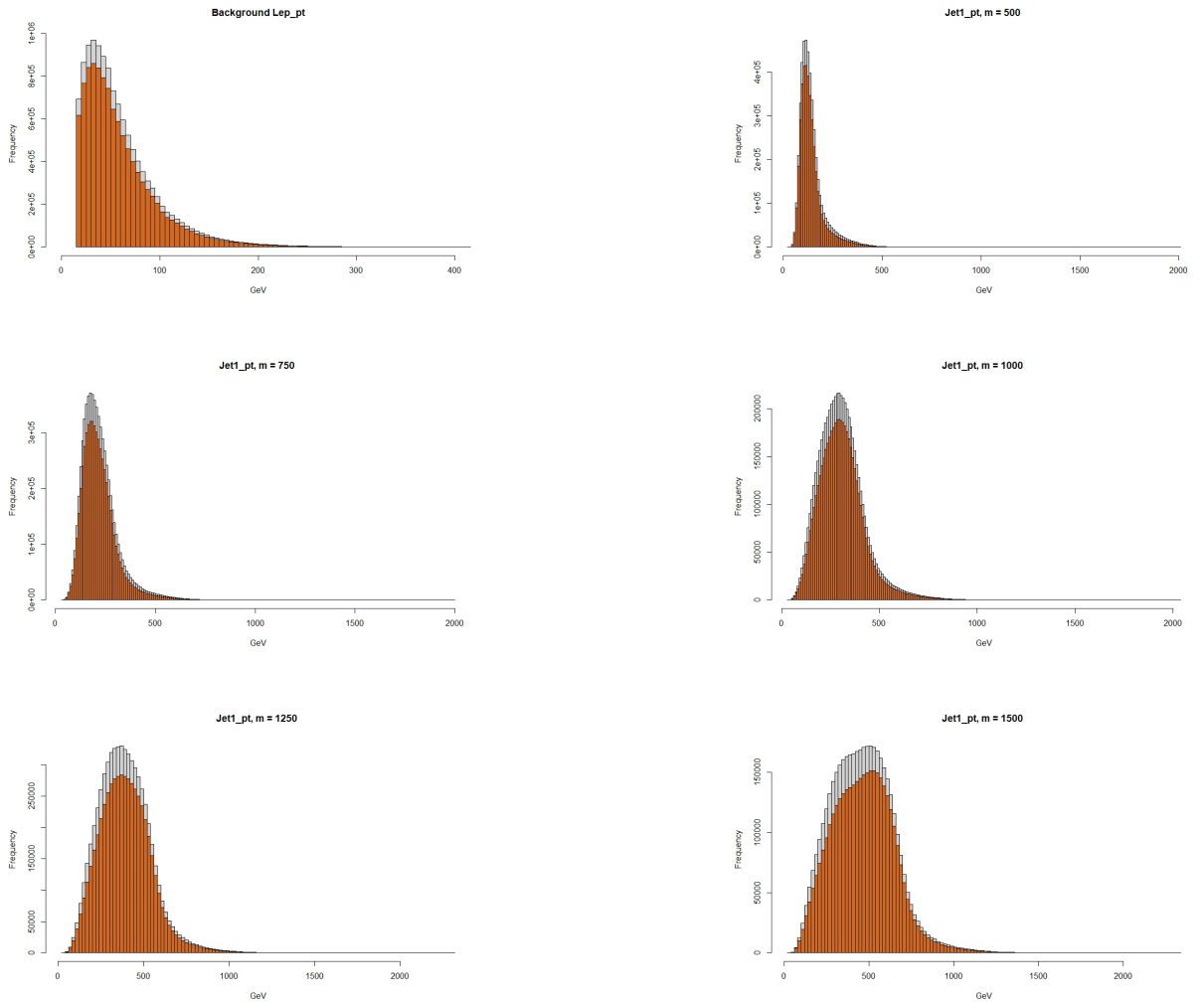


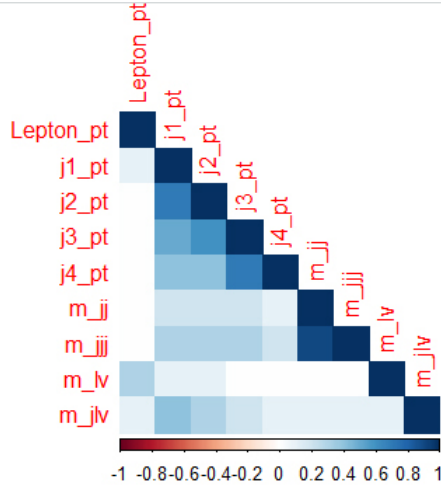
Figura 8: Distribución del momento transversal del *jet* más energético en los *datasets* con todas las masas. El blanco corresponde a todos los sucesos y el marrón corresponde a los sucesos en los cuales se etiquetó a 2 *jets* como *btag*

Tanto en la figura 7 como en la 8 se observa claramente que a mayor masa, más se diferencian los sucesos del *background*. Por otra parte, hay otras variables que serán irre-

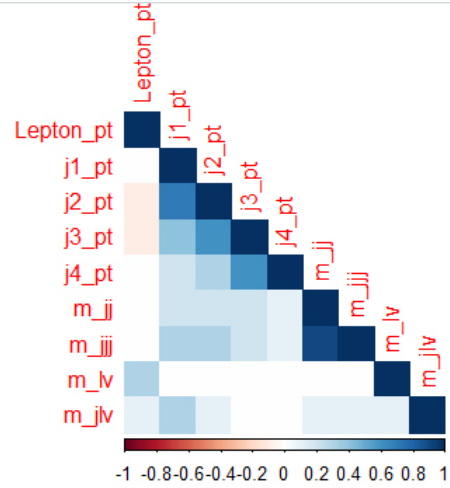
levantes para nuestro problema pues siguen distribuciones homogéneas como los ángulos de cualquier partícula.

2.3. Matrices de correlación de las variables

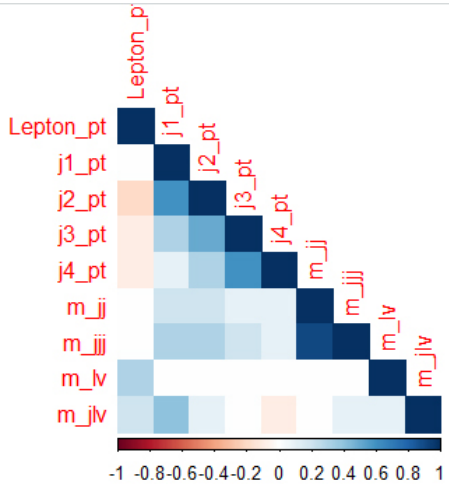
Usando los *outputs* sin normalizar de los datos generados, podemos establecer cuan correlacionados están las variables de nuestro problema y establecer que variables son las más importantes para la creación de nuestros modelos. Ni los ángulos azimutales ni la *pseudorapidity* son relevantes para ninguna partícula, así que han sido previamente descartados.



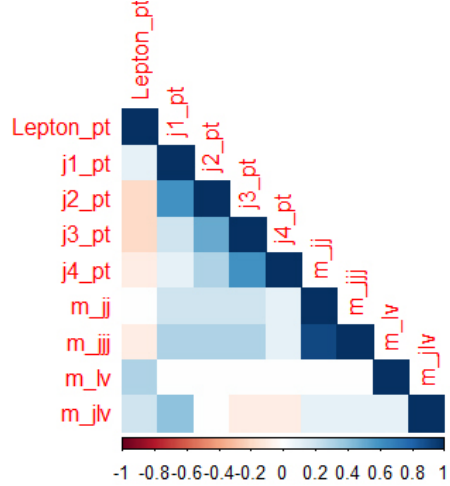
(a) Matriz de correlación para $m = 500$ GeV



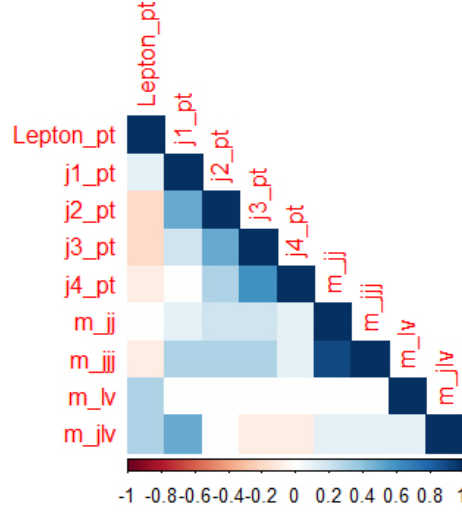
(b) Matriz de correlación para $m = 750$ GeV



(c) Matriz de correlación para $m = 1000$ GeV



(d) Matriz de correlación para $m = 1250$ GeV



(e) Matriz de correlación para $m = 1500$ GeV

Figura 9: Matrices de correlación entre las variables de nuestros *datasets* sin normalizar

Lo más destacable de la figura 9 es observar la fuerte correlación entre las masas invariantes M_{jj} y M_{jjj} , lo cual tiene sentido pues M_{jj} es usado para calcular M_{jjj} .

3. Modelos de clasificación binaria haciendo uso de distintos algoritmos de *Machine Learning*.

Ahora se explicará como se han integrado distintos métodos de *machine learning* en nuestros modelos. Como se ha mencionado previamente, haremos uso de *Decision Trees* y de métodos más avanzados que parten de este tales como *Boosted Decision Trees* y *Random Forest*. Para facilitar el entrenamiento de estos modelos, se han usado los *datasets* con los datos normalizados. Estos *datasets* contiene un 50 % de observables correspondientes a un suceso señal y otro 50 % correspondiente a sucesos *background*. Luego se ha realizado una subdivisión de los *datasets* en otros de tipo *train* y *test*. Para generar el *dataset train* usaremos el 70 % de los datos originales para proporcionar al modelo una variedad suficientes de observables. El otro 30 % de los datos los agruparemos para crear el *test* que será de utilidad para medir el rendimiento de nuestros modelos sobre observables nuevos.

3.1. *Decision Trees*

Un árbol de decision o *Decision Tree* [13] es un algoritmo de aprendizaje supervisado que usaremos para un problema de clasificación binario. Un *decision tree* consta de distintos tipos de nodos, el nodo *root*, que no tiene ninguna entrada por parte de otros nodos, *internal* o *test nodes*, que son aquellos que tienen una entrada y varias salidas. Por último están las hojas también llamados *terminal* o *decision nodes*, que son aquellos nodos que no cuentan con salidas.

En un árbol de decisión cada nodo interno toma una decisión basada en si una variable cumple o no una condición y se divide en diversos subespacios. Tras pasar por un

número determinado de nodos internos, las hojas recogen una clase y la probabilidad de que esta sea cierta. Pueden manejar tanto datos cualitativos como cuantitativos. La complejidad de un árbol de decisión viene dada por diversos parámetros como el número de nodos, el número total de hojas, la profundidad del árbol y por último, un hiperparámetro que controla todos los anteriores llamado *Complexity Parameter* (CP). Esta herramienta nos es muy útil para nuestro problema, pues los DT pueden manejar *datasets* muy grandes y tienen un gran valor predictivo.

Para este trabajo se ha hecho uso del programa RStudio [14] y de la librería *rpart* para generar los DTs, *rpart.plot* [15] para visualizar los árboles y *caret* [6] para generar las matrices de confusión y medir el rendimiento del modelo.

Se puede ejemplificar el proceso por el cuál se entrena un DT con los datos del *train* como una superficie bidimensional sobre la que se sitúan ambas clases. Se busca un valor en cada eje de tal forma que con una única división del espacio se consiga separar eficazmente ambas clases. Este proceso saca un valor numérico umbral para un nodo y se repite para cada nodo con el objetivo de que cada subespacio contenga una única clase. Para mejorar el poder del clasificador, se modifican los parámetros del DT, si se disponen de pocas ramas y hojas, se obtienen árboles que tienden a infra-ajustar los datos y disminuye su eficacia. Por el contrario, si se dispone de demasiadas ramas y hojas, los árboles tienden a sobre-ajustar (*overfitting*) y le dan más importancia de la debida a algunas variables de los observables.

Para que cada nodo decida que valor numérico es el más óptimo para dividir el subespacio, se emplean métricas como el índice de Gini, la entropía y el error de clasificación [9]. Siendo p_i la probabilidad de una clase i

Índice de Gini

$$Gini = 1 - \sum p_i^2$$

Error de clasificación

$$Error_{clas} = 1 - \max(p_i)$$

Entropía

$$Entropia = - \sum p_i \log_2 \frac{1}{p_i}$$

Con esto, se puede obtener un índice de Gini en función de los pesos de cada suceso para maximizar $Gini_{Padre} - (Gini_{Hijo1} + Gini_{Hijo2})$.

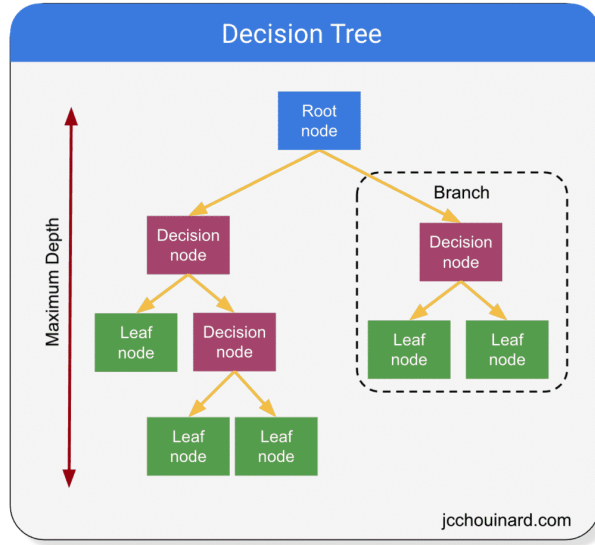


Figura 10: Estructura simplificada de un *Decision Tree*

3.2. *Boosted Decision Trees*

Se hace uso de técnicas alternativas que combinan de diversas formas los árboles de decision. Los *Boosted Decision Trees* permiten utilizar árboles con poco poder clasificatorio, como árboles poco complejos o compuestos por un único nodo (*stumps*) como buenos clasificadores. Para ello, al set de datos de entrenamiento se le asigna un peso de forma uniforme a cada suceso. Según si el árbol simple ha sido capaz de clasificar correctamente un suceso o no, se le reasigna un nuevo peso. Por ejemplo, a un suceso incorrectamente predicho se le aumenta su peso para que si en la siguiente iteración se vuelve a clasificar incorrectamente, se penalice ese tipo de modelo. A mayor número de iteraciones, menor error de entrenamiento y mayor poder clasificador. Son menos propensos a tener problemas de *overfitting* incluso habiendo empleado árboles de hasta profundidad 3.

Los BDTs usados se han creado con el algoritmo *Adaptive Boosting* o *AdaBoost*, haciendo uso de las librerías de R *ada* [8] y *adabag*, cuyo funcionamiento se describe a continuación:

- Primero se le asigna un peso igual a todos los sucesos tal que $w(x_i) = \frac{1}{N}$ siendo N el número total de observables.
- Luego se genera un árbol de cada variable del problema de profundidad 1 llamado *stump* y el árbol que tenga menor índice de Gini será el que se escoja primero.
- Posteriormente se calcula un parámetro llamado *Amount of Say, Importance* o α que dará el rendimiento de nuestro *stump*:

$$\alpha = \frac{1}{2} \log_e \frac{1 - ErrorTotal}{ErrorTotal}$$

Siendo el *Error Total* los sucesos mal clasificados multiplicados por el peso de ese suceso.

- Una vez calculado el *Amount of Say*, se recalculan los pesos de los sucesos de forma que:

$$\omega' = \omega e^{\pm \alpha}$$

Siendo ω' el peso nuevo y ω el peso anterior. El α será positivo si el suceso fue incorrectamente clasificado por el árbol anterior, dándole así más peso a ese suceso, o será negativa si el suceso fue correctamente clasificado, disminuyendo así su peso. Hay que tener en cuenta que la suma de los nuevos pesos no será igual a 1, por lo que habrá que normalizarlos de nuevo.

- Luego se organizan los sucesos en subgrupos cuyos valores vienen dados por los nuevos pesos, de tal forma que se crea un espacio donde los sucesos incorrectamente clasificados ocupan más que los correctamente clasificados. Esto puede observarse en la figura 11 de un *dataset* correspondiente a un ejercicio de ejemplo.

Row No.	Gender	Age	Income	Illness	New Sample Weights	Buckets
1	Male	41	40000	Yes	0.1004/0.8004=0.1254	0 to 0.1254
2	Male	54	30000	No	0.1004/0.8004=0.1254	0.1254 to 0.2508
3	Female	42	25000	No	0.1004/0.8004=0.1254	0.2508 to 0.3762
4	Female	40	60000	Yes	0.3988/0.8004=0.4982	0.3762 to 0.8744
5	Male	46	50000	Yes	0.1004/0.8004=0.1254	0.8744 to 0.9998

Figura 11: División del espacio en función de los grupos creados usando los nuevos pesos

- Se seleccionan N números al azar entre 0 y 1 y se usan para sacar sucesos del *dataset* original y crear uno nuevo que naturalmente contendrá en mayor proporción sucesos incorrectamente clasificados.
- Finalmente, con el *dataset* nuevo, se repite el proceso, es decir, se le asigna un peso uniforme a cada suceso, se crea un árbol con la variable que dé menor índice de Gini posible, se calcula el α , los nuevos pesos y se genera otro *dataset*. Se realizan las iteraciones necesarias hasta que el *Training Error* sea mínimo.

3.3. *Random Forest*

Los *Random Forests* fueron introducidos por Leo Breiman, inspirado a su vez por el trabajo de Amit y Geman y fueron desarrollados como competencia al *boosting* [2]. Los RF pueden ser utilizados para problemas de clasificación, como es nuestro caso, o de regresión. Tienen diversas ventajas como:

- Manejar problemas de regresión y de clasificación multiclase.
- Son relativamente rápidos a la hora de entrenar los modelos y de elaborar las predicciones.
- Son fáciles de ajustar.
- Estima con facilidad un error general del modelo.
- Puede implementarse en paralelo, es decir, dividir el número de DTs a calcular en más de un único *core* de la CPU del ordenador, reduciendo así su tiempo de cálculo.

Partiendo de múltiples subdivisiones aleatorias del *dataset train*, se generan DTs cuyos nodos recogen una combinación de variables distintas entre ellos debido a la aleatoriedad de los datos escogidos para crearlos. El resultado de estos árboles se combinan homogéneamente, sin tener en cuenta ningún tipo de peso entre los árboles. Dado que para generar estos árboles se escogen grupos de sucesos aleatorios del *dataset train* original, hay un conjunto de sucesos que no se habrán empleado en la creación de ninguno de los árboles, a esto se le llama *Out-of-Bag Data*, el cual resulta extremadamente útil para sacar una estimación del error general del modelo. Haciendo uso del paquete *randomForest* de RStudio [11] se han creado nuestros modelos, pudiendo modificar el número de árboles y el número de nodos que contienen cada uno de ellos.

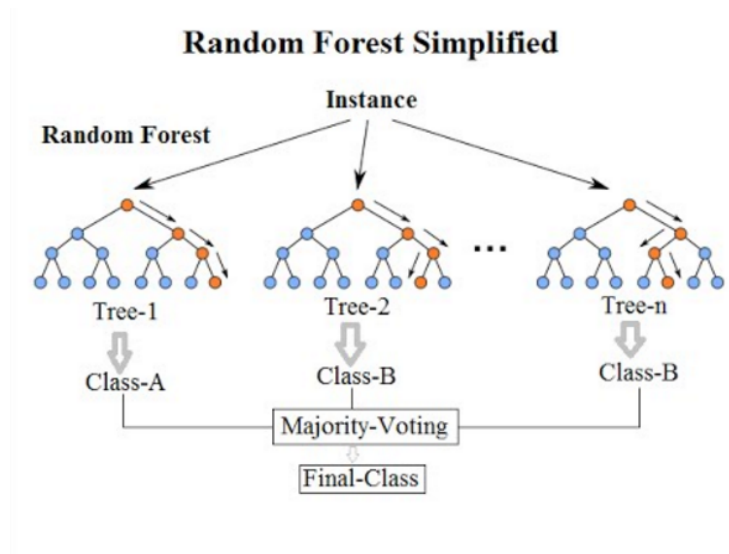


Figura 12: Esquema del funcionamiento de un *Random Forest*

4. Métricas del rendimiento de los modelos

Una vez creado nuestros modelos, cuyo objetivo es separar de la forma más eficaz nuestra señal del fondo, es necesario poder medir adecuadamente el rendimiento de estos. Para ello, se observa cada modelo empleado como clasifica el *dataset test* en señal o fondo si la *puridad* (P) supera un umbral determinado del 0.5. Una vez realizada la predicción del *test* se comprueba las clases predichas con la clase real y se saca la *matriz de confusión*.

En una matriz de confusión se puede visualizar fácilmente la comparación entre la predicción y las clases reales como puede observarse en la figura 13

VALORES PREDICCIÓN	Verdaderos positivos	Falsos Positivos
	Falsos Negativos	Verdaderos Negativos
	VALORES REALES	

Figura 13: Matriz de confusión

En una matriz de confusión hay:

- *True Positive* (TP): Un suceso que es señal se clasifica correctamente como señal.
- *False Positive* (FP): Un suceso que es fondo es incorrectamente clasificado como señal.
- *False Negative* (FN): Un suceso que es señal es incorrectamente clasificado como fondo.
- *True Negative* (TN): Un suceso que es fondo se clasifica correctamente como fondo.

Gracias a estos parámetros, es posible construir clasificadores más refinados como el *accuracy*, la *Kappa de Cohen*, *sensitivity*, *specificity*, *F1-Score*, *AUC*...

4.1. *Accuracy*

Es el porcentaje que tiene un modelo de clasificar correctamente un suceso entre su totalidad. Su resultado oscila entre el 0 y el 1, siendo el 1 una clasificación perfecta y se puede calcular como:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

4.2. *Kappa de Cohen*

Es una medida estadística que compara la concordancia observada del conjunto de datos usados respecto a la que podría ocurrir de forma aleatoria. Su valor se encuentra en el rango entre -1 y 1 siendo 0 el puro azar. Es sensible a *datasets* desbalanceados. Puede calcularse como:

$$Kappa = \frac{(TP + FP) \cdot (TP + FN) + (TN + FP) \cdot (TN + FN)}{TP + TN + FP + FN}$$

4.3. *Sensitivity*

Es el porcentaje de señal que el modelo es capaz de clasificar correctamente de entre los clasificados como positivos siendo:

$$Sensitivity = \frac{TP}{TP + FP}$$

4.4. *Specificity*

Porcentaje de señal que el modelo es capaz de clasificar correctamente de entre todos los sucesos siendo:

$$Specificity = \frac{TN}{TN + FP}$$

4.5. *F1-Score*

Se trata de la medida armónica entre *sensitivity* y *specificity* siendo:

$$F1 - Score = 2 \frac{Sensitivity \cdot Specificity}{Sensitivity + Specificity}$$

4.6. *AUC (Area Under the Curve)*

Es el área subtendida por la curva ROC (*Receiver Operator Characteristic*) [4], que es una representación de la *sensitivity* frente a $1 - specificity$. Su valor oscila entre el 0 y el 1, siendo 1 una predicción perfecta.

5. Resultados

Una vez explicado el fundamento de los distintos modelos empleados de *Machine Learning*, veremos como hemos entrenado los modelos con distintos *datasets train* de las todas las posibles masas de la resonancia $t\bar{t}$. Primero se ha buscado el ajuste óptimo de los parámetros de cada método utilizado (*tune*).

5.1. *Decision Tree*

En los DTs podemos modificar tanto la profundidad de los árboles como el número máximo o mínimo de hojas, etc. Pero solo modificaremos el *Complexity Parameter* (CP), pues se trata de un hiperparámetro que controla los demás. En las siguientes figuras puede observarse su rendimiento para todas las masas siendo entrenado con un *train* y cuya predicción se ha hecho y comprobado con un *test* cuya clasificación entre señal y fondo ya fue generada junto a los datos simulados.

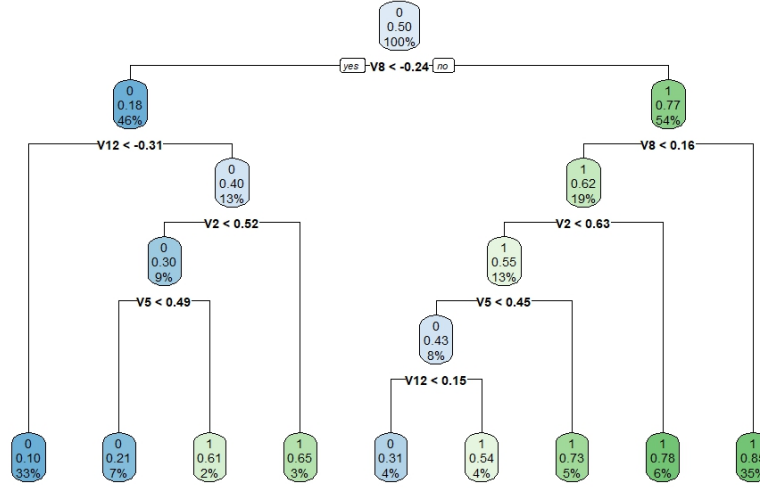
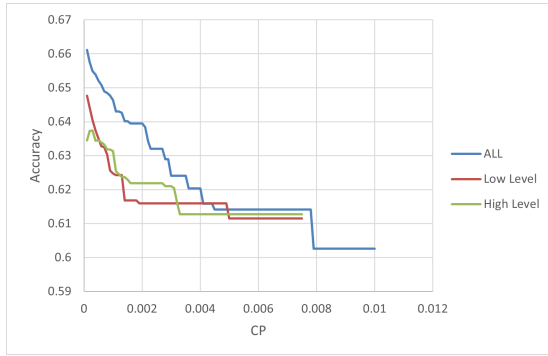
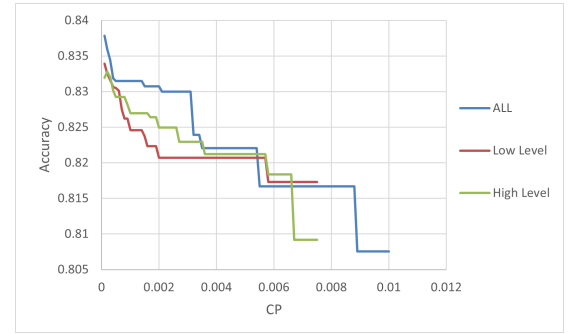


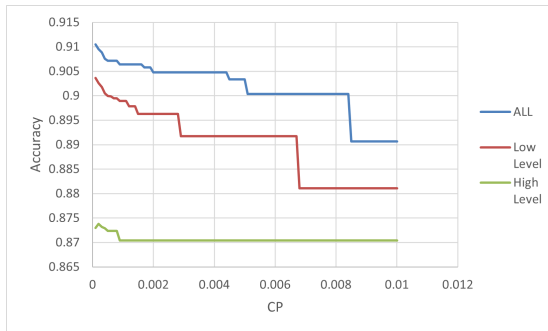
Figura 14: Ejemplo de *Decision Tree* de $m = 750$ GeV y $CP = 0.005$ haciendo uso de 4 variables distintas: $V2$ = Momento transversal del leptón, $V5$ = Momento transversal del neutrino, $V8$ = Momento transversal del *jet* 1 y $V12$ = Momento transversal del *jet* 2



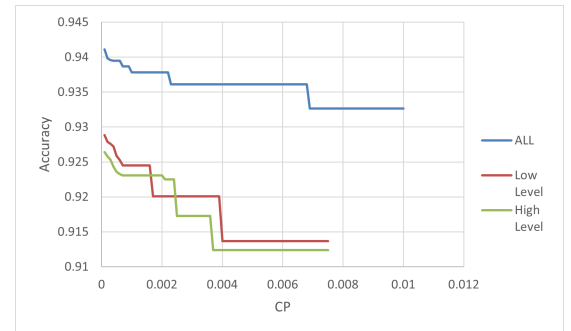
(a) Accuracy en función del CP para $m = 500$ GeV usando variables *Low Level* y *High Level*



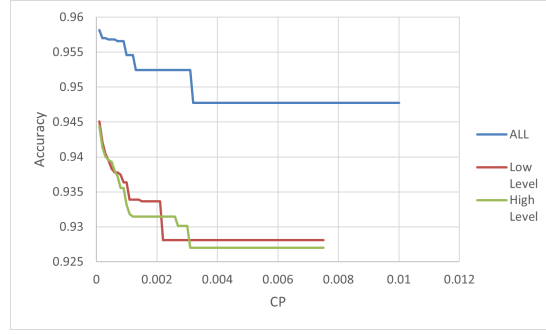
(b) Accuracy en función del CP para $m = 750$ GeV usando variables *Low Level* y *High Level*



(c) Accuracy en función del CP para $m = 1000$ GeV usando variables *Low Level* y *High Level*



(d) Accuracy en función del CP para $m = 1250$ GeV usando variables *Low Level* y *High Level*



(e) Accuracy en función del CP para $m = 1500$ GeV usando variables *Low Level* y *High Level*

Figura 15: Accuracy en función del CP

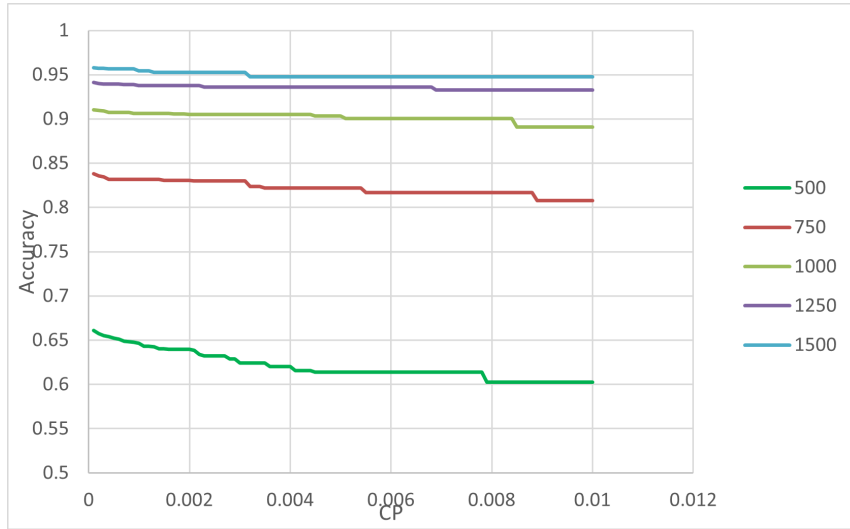


Figura 16: Accuracy en función del CP para todas las masas de la partícula X

Como se puede observar, se obtiene un mejor *accuracy* según menor es el CP. Esto es lógico, pues el CP puede entenderse como el parámetro que detiene el ‘crecimiento’ del árbol y a menor CP, mayor el árbol y mayor el número de nodos que aumentan las divisiones en el espacio de clases. Aún a pesar de que los DTs con muchas ramas y hojas suelen padecer de *overfitting* a los datos empleados en el *dataset train*, estos son muy similares a las predicciones realizadas con el *dataset test* (pues son datos simulados a partir de procesos idénticos) y no resultan en un caso de *overfitting*.

Y ahora evaluaremos los DTs haciendo las predicciones con un *dataset test* que se ha modificado cambiando la clasificación entre señal y fondo originales (los generados en la simulación), por otro más ‘realista’. Se ha realizado separando la señal y el fondo en función de la masa invariante M_{wwbb} , pues es lógicamente la variable del problema que mejor divide los sucesos.

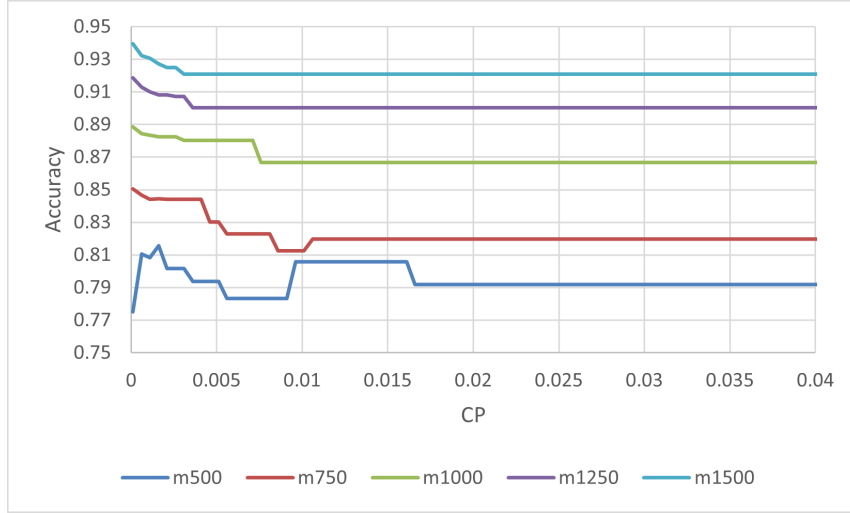


Figura 17: *Accuracy* en función del CP para todas las masas de la partícula X, usando un *test* que simula una clasificación real.

En la figura 17 se ven ciertas anomalías en con los datos correspondientes a $m = 500$ GeV, esto es debido a que sus sucesos BSM son difícilmente distinguibles del *background* y los DTs generados sufren de *overfitting*.

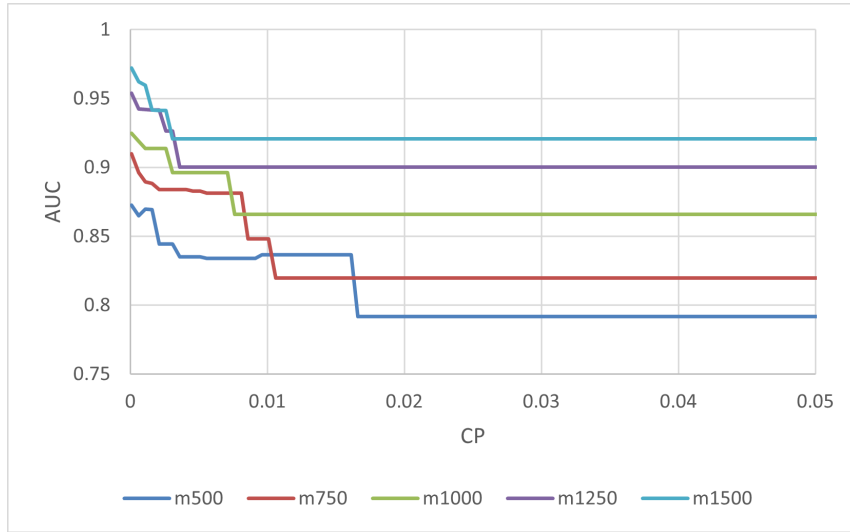
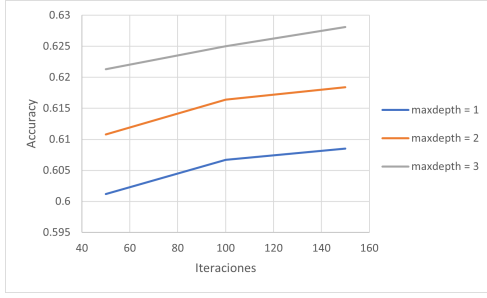


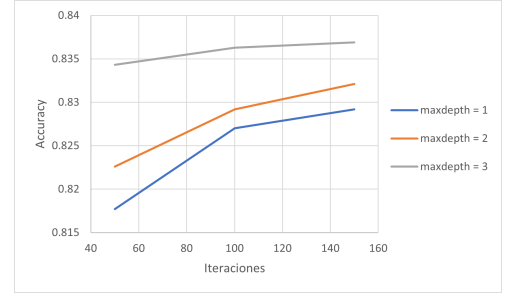
Figura 18: *Area Under the Curve* en función del CP para todas las masas de la partícula X, usando un *test* que simula una clasificación real.

5.2. *Boosted Decision Tree* con AdaBoost

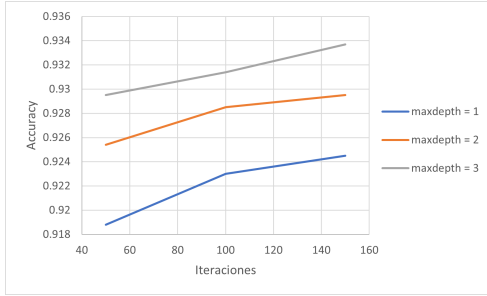
Una vez establecido como varía los resultados de los DT en función de la masa de la partícula, usaremos el método de BDT AdaBoost. Se comparará el rendimiento de los BDT al variar el número de iteraciones y la profundidad de los árboles empleados con los datos *test* originales.



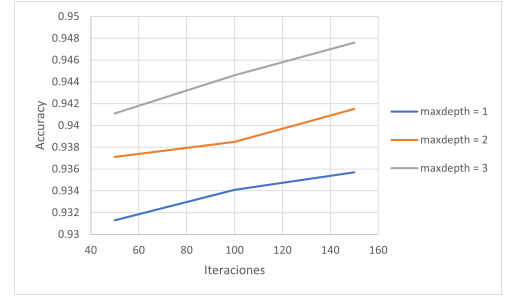
(a) *Accuracy* en función de la profundidad máxima del árbol y de sus iteraciones para sucesos de $m = 500$ GeV.



(b) *Accuracy* en función de la profundidad máxima del árbol y de sus iteraciones para sucesos de $m = 750$ GeV.



(c) *Accuracy* en función de la profundidad máxima del árbol y de sus iteraciones para sucesos de $m = 1250$ GeV.



(d) *Accuracy* en función de la profundidad máxima del árbol y de sus iteraciones para sucesos de $m = 1500$ GeV.

Figura 19: *Accuracy* en función de la profundidad de los árboles

Como vemos en la figura 19, el *accuracy* aumenta según lo hace el número de iteraciones y la profundidad de los árboles.

Ahora procederemos con los datos *test* ‘reales’, y se evaluará su rendimiento con las distintas métricas. Escogeremos una profundidad máxima del árbol de 3 niveles y 300 iteraciones, pues como se ve en las siguientes figuras, el *training error* (ver figura 20) llegará a un mínimo y es muy costoso computacionalmente.

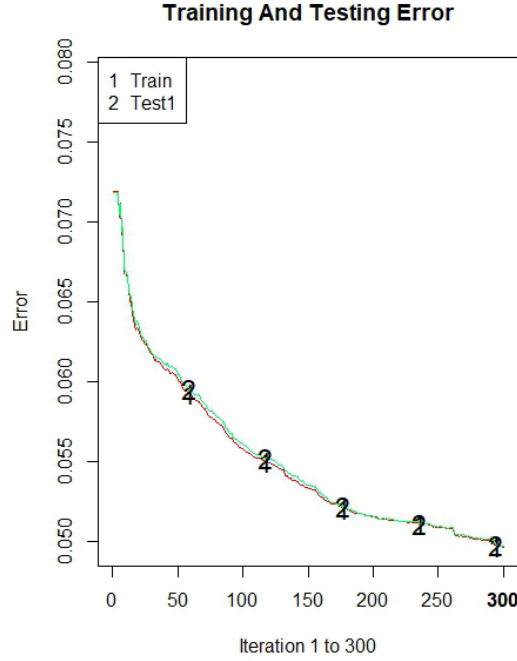
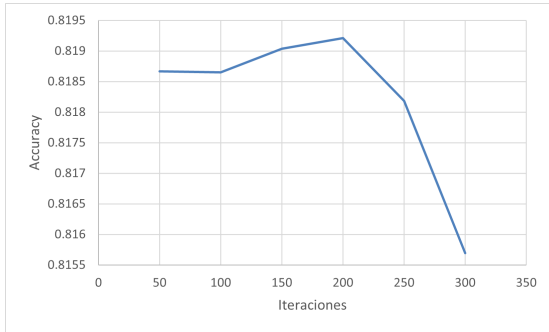
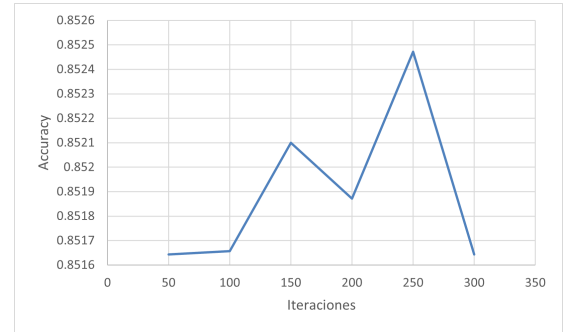


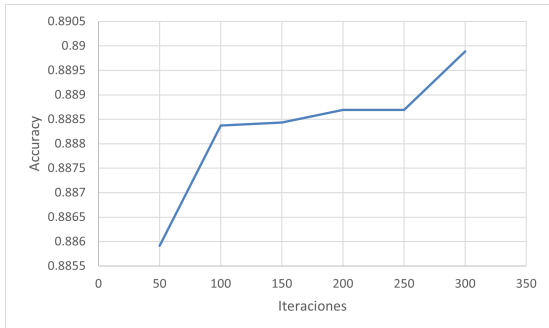
Figura 20: Ejemplo del *training error* del AdaBoost con cada iteración. Se puede observar en rojo el error de los datos *train* y en turquesa el error de los datos *test*, el cuál es lógicamente mayor.



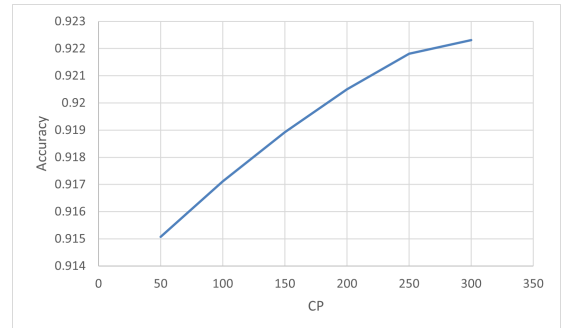
(a) *Accuracy* en función de las iteraciones para datos ‘reales’ de $m = 500$ GeV



(b) *Accuracy* en función de las iteraciones para datos ‘reales’ de $m = 750$ GeV

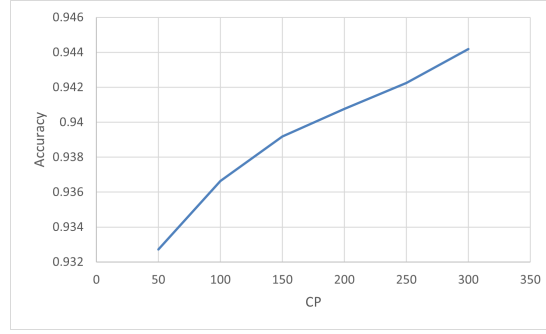


(c) *Accuracy* en función de las iteraciones para datos ‘reales’ de $m = 1000$ GeV



(d) *Accuracy* en función de las iteraciones para datos ‘reales’ de $m = 1250$ GeV

En la figuras 21a y 21b se ve que según aumentan las iteraciones, disminuye su *accuracy* porque los sucesos de las masas más bajas son más difíciles de distinguir del *background*.



(e) *Accuracy* en función de las iteraciones para datos ‘reales’ de $m = 1500$ GeV

Figura 21: *Accuracy* en función de las iteraciones del AdaBoost con árboles de profundidad 3 para datos ‘reales’.

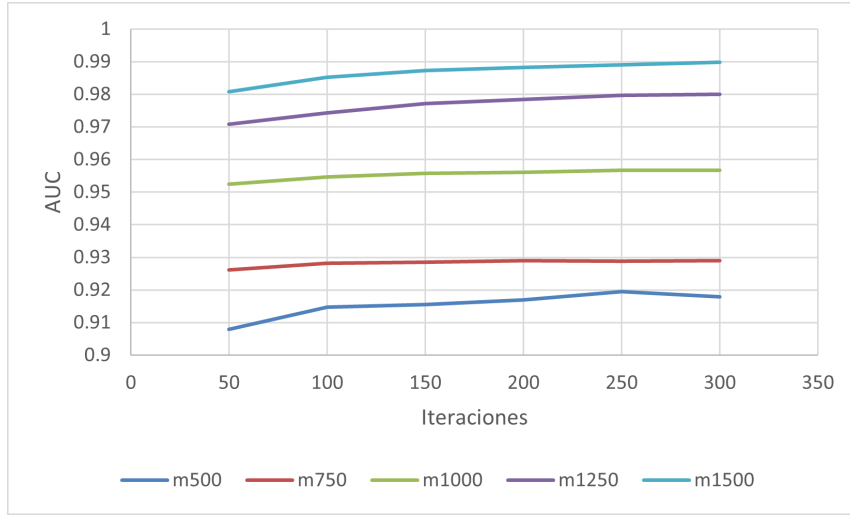


Figura 22: *Area Under the Curve* en función de las iteraciones del AdaBoost con árboles de profundidad 3 para todas las masas de la partícula X, usando un *test* que simula una clasificación real.

5.3. *Random Forest*

Para entrenar los modelos que usan el algoritmo *Random Forest* debido a un equilibrio entre eficacia y costo computacional, se han escogido muestras de 500 DTs de 5 variables cada uno de ellos. Empleando los datos *test* ‘reales’ obtenemos:

Masa (GeV)	Accuracy	Kappa	Sensitivity	Specificity	F1-Score
500	0.787	0.574	0.632	0.941	0.756
750	0.851	0.700	0.764	0.936	0.841
1000	0.895	0.790	0.842	0.946	0.891
1250	0.925	0.849	0.894	0.955	0.923
1500	0.946	0.892	0.928	0.964	0.946

Tabla 1: Rendimiento del método *Random Forest* con 500 ejemplares de 5 variables

6. Conclusiones

Se han entrenado todos los modelos de *machine learning* con los datos simulados de una colisión pp del experimento ATLAS a $\sqrt{s} = 8TeV$ para que clasifiquen unos sucesos como señal (BSM) o fondo (SM). Para lograr esto, se han utilizado los datos generados por Montecarlo que se encuentran en el repositorio HEPMASS, se ha analizado las variables del problema y determinado su relevancia y se ha empleado el programa RStudio con paquetes personalizados para cada método. Se ha discutido las diversas métricas que se pueden emplear para medir el rendimiento de los modelos y al construir los modelos, se ha buscado que parámetros de *tune* son los mejores en relación eficacia-costo computacional. Una característica común de todos los modelos es que, a mayor masa, mayor rendimiento, lo cual es lógico, pues sus sucesos son más distinguibles del *background* SM.

- En los DTs, a menor CP, mayor rendimiento.
- En los BDTs, aún a pesar de estar diseñados para trabajar con árboles con un único nivel de profundidad (*stumps*), para nuestro problema es más eficaz aumentarlo a 3 niveles de profundidad y limitar las iteraciones sobre 300, pues ahí es cuando el *training error* se estabiliza.

Luego, se le ha pasado a nuestros modelos un *dataset test* cuya clasificación de señal/fondo se ha cambiado por una que emula más un caso real en el cual desconocemos de antemano si se trata de un suceso BSM o SM. Para ello, se ha dividido usando el criterio de la masa invariante de la resonancia $M_{w\bar{w}bb}$. Se ha observado como por ejemplo en los casos para las masas más bajas $m = 500$ GeV y $m = 750$ GeV empeora el rendimiento o causa *overfitting* en el caso de los DTs o como baja la *accuracy* en cada iteración del AdaBoost debido a que sus sucesos BSM son muy semejantes al fondo SM.

Comparando todos los modelos usando los datos *test* ‘reales’ obtenemos:

	Mejor DT			Mejor BDT		
Masa (GeV)	Accuracy	Kappa	F1-Score	Accuracy	Kappa	F1-Score
500	0.815	0.631	0.795	0.819	0.638	0.785
750	0.850	0.700	0.836	0.852	0.704	0.837
1000	0.888	0.777	0.881	0.889	0.779	0.882
1250	0.918	0.837	0.916	0.922	0.844	0.919
1500	0.939	0.878	0.937	0.944	0.888	0.942

	Mejor RF		
Masa (GeV)	Accuracy	Kappa	F1-Score
500	0.787	0.574	0.756
750	0.851	0.700	0.841
1000	0.895	0.790	0.891
1250	0.925	0.849	0.923
1500	0.946	0.892	0.946

Tabla 2: Comparación de los métodos de ML usados

Para masas bajas, los BDTs parecen ser clasificadores más eficaces mientras que para mayores masas, tanto los BDTs como el RF mejoran ligeramente como clasificadores. Los resultados de los clasificadores son similares entre ellos, pues los parámetros de ajuste están muy optimizados, pero en el caso de los DTs sería muy complejo y costoso mejorar sus resultados, mientras que en el caso de los BDTs y RF sus resultados aún son ligeramente mejorables sin que sea tan costoso como los DTs.

Como mejora, se podría realizar estos cálculos en un equipo informático preparado para comprobar las diferencias de tiempo entre unos y otros métodos bajo las mismas condiciones.

6.1. Conclusions

All the machine learning models have been trained with the simulated data of a pp from the ATLAS experiment $\sqrt{s} = 8\text{TeV}$ to classify some events as signal (BSM) or background (SM). In order to achieve this, Montecarlo generated data found in the HEPMASS repository has been used, the problem variables have been analyzed, their relevance determined and RStudio software has been employed with custom packages for each method. The various metrics that can be used to measure the performance of the models, it has been sought which parameters of ‘tune’ are the best in computational cost-effectiveness. A common feature of all models is that the higher the mass, the higher the performance, which is logical, since their events are more distinguishable from the background SM.

- *In the DTs, the lower the CP, the higher the performance.*
- *In the BDTs, even though they are designed to work with trees with a single depth level (stumps), for our problem it is more efficient to increase it to 3 depth levels and cap the iterations around 300, because that is when the training error stabilizes.*

Then, a dataset test to our models whose signal/background classification has been changed to one that emulates more a real case in which we do not know in advance whether it is a BSM or SM event. For this purpose, it has been divided using the criterion of the invariant mass resonance M_{wwbb} . It has been observed how for example, in the cases for the lowest masses $m = 500\text{ GeV}$ y $m = 750\text{ GeV}$ it worsens the performance or causes overfitting in the case of DTs or how it lowers the accuracy in each iteration of the AdaBoost because it’s BSM events are very similar to the SM background.

Seeing table 2, we could know that for low masses, BDTs appear to be more effective classifiers while for higher masses, both BDTs and RF improve slightly as classifiers. The results of the classifiers are similar among them, since the fitting parameters are highly optimized, but in the case of the DTs it would be very complex and costly to improve their results, while in the case of the BDTs and RF their results are still slightly improvable without being as costly as the DTs.

As an improvement, these calculations could be performed on a computer prepared to test the time differences between one method and the other under the same conditions.

Referencias

- [1] G. Aad, T. Abajyan, and B. Abbott. Search for new phenomena in photon+jet events collected in proton–proton collisions at $s=8$ tev with the atlas detector. *Physics Letters B*, 728:562–578, 2014.
- [2] Adele Cutler, D. Richard Cutler, and John R. Stevens. *Random Forests*, pages 157–175. Springer US, Boston, MA, 2012.
- [3] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.
- [4] Tom Fawcett. *An Introduction to ROC Analysis*, pages 861–874. 2006.
- [5] N. V. Krasnikov. Physics at the large hadron collider. *Physics of Particles and Nuclei*, 28(5):441, sep 1997.
- [6] Max Kuhn. Classification and regression training, 2022.
- [7] Paul Langacker. *The Standard Model and Beyond*, pages 425–432. 2021.
- [8] Kjell Johnson Mark Culp and George Michailidis. The r package ada for stochastic boosting, 2016.
- [9] José Francisco Salt Miruna Alexandra, Santiago González. Estudi de l’extracció de senyal ressonant de successos $t\bar{t}$ mitjançant tècniques de machine learning a l’experiment atlas. 2021.
- [10] L.V.R Sai Y.P. Mohebbanaaz, Kumari. *Classification of ECG beats using optimized decision tree and adaptive boosted optimized decision tree*, pages 695–703. 2022.
- [11] Fortran original by Leo Breiman, R port by Andy Liaw Adele Cutler, and Matthew Wiener. Breiman and cutler’s random forests for classification and regression, 2022.
- [12] P. Sadowski D. Whiteson P. Baldi. *Searching for exotic particles in high-energy physics with deep learning*. 2014.
- [13] Lior Rokach and Oded Maimon. *Decision Trees*, volume 6, pages 165–192. 01 2005.
- [14] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, PBC., Boston, MA, 2020.
- [15] Brian Ripley Terry Therneau, Beth Atkinson. Recursive partitioning and regression trees, 2022.
- [16] Christopher Vermilion. Jet substructure at the large hadron collider. pages 65–80, 01 2011.
- [17] José Salt Ángela García, Jordi Muñoz. Machine learning en el estudio $t\bar{t}$ del experimento atlas. 2020.