APLICACIÓN DE MACHINE LEARNING PARA LA DETECCIÓN DE DECRECIMIENTOS FORBUSH EN SEÑALES OBTENIDAS DE MONITORES DE RADIACIÓN SOLAR

ÁLVARO ALEXIS VALBUENA DELGADO

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

FACULTAD DE INGENIERÍA FÍSICO-MECÁNICAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER

BUCARAMANGA

2015

APLICACIÓN DE MACHINE LEARNING PARA LA DETECCIÓN DE DECRECIMIENTOS FORBUSH EN SEÑALES OBTENIDAS DE MONITORES DE RADIACIÓN SOLAR

ÁLVARO ALEXIS VALBUENA DELGADO

Trabajo de grado para optar al título de

Ingeniero de sistemas

Director

Ph.D. Raúl Ramos Pollán

ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER

BUCARAMANGA

2015

DEDICATORIA

Dedico este trabajo, fruto de un gran esfuerzo, a mis padres Álvaro y Ana, por brindarme todos en estos años; por su paciencia y sacrificio.

A mi hermana Diana, por sus concejos como profesional y sobre todo por su cariño.

A Natalie, por brindarme todo su apoyo y cariño, sobre todo en los momentos más difíciles, su enorme sacrificio también hizo posible este trabajo.

Al profesor Raúl, por ofrecerme su conocimiento, experiencia y por ser el mejor guía.

Álvaro.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

Título: APLICACIÓN DE MACHINE LEARNING PARA LA DETECCIÓN DE DECRECIMIENTOS FORBUSH EN SEÑALES OBTENIDAS DE MONITORES DE RADIACIÓN SOLAR*.

Autor: Álvaro Alexis Valbuena Delgado**.

Palabras clave: Machine Learning, Decrecimientos Forbush, Máquinas de vectores de soporte, Python.

Descripción: En el estudio de los fenómenos astrofísicos, se ha visto la gran importancia que han llegado a tener las herramientas informáticas para lograr entenderlos, y al final, entender la naturaleza del mundo que nos rodea. El desarrollo de estas herramientas es imprescindible para lograr alcanzar el conocimiento que se puede obtener del mecanismo que hace funcionar estos fenómenos.

Con el aporte de las nuevas técnicas que trae el campo del machine learning para procesar la información que representa estos fenómenos, se presenta una gran oportunidad para aportar herramientas informáticas que ayuden a los científicos a lograr sus metas y aportar su conocimiento a la humanidad.

Es por esto que se hace necesario, gracias a las técnicas ofrecidas por el machine learning, desarrollar una serie de herramientas informáticas que se implementarán en diferentes detectores alrededor del mundo y ayudarán a los científicos en su estudio de la radiación solar, la gestión y economía de los detectores ya que al analizar las señales en tiempo real, se almacenarán solo los eventos detectados y no la totalidad de la señales, esto es debido a que los sistemas en los que posiblemente se implemente el modelo puedan ser sistemas Raspberry Pi (ordenadores de placa reducida).

Por esto también se hace necesario la aplicación del machine learning, porque una vez se ha desarrollado y entrenado el modelo, será entonces posible identificar, seleccionar y guardar solo los datos necesarios para su transmisión, evitando así la transferencia de grandes volúmenes de información innecesaria.

En este trabajo se propones desarrollar un modelo predictivo el cual permita al grupo Halley en colaboración con el proyecto LAGO, detectar y almacenar solo los eventos astrofísicos, automatizando un proceso de clasificación el cual optimizaría la capacidad de almacenamiento que tiene el proyecto LAGO.

* Trabajo de Grado

** Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas, Director Ph.D. Raúl Ramos Pollan.

ABSTRACT

Título: MACHINE LEARNING APPLICATION FOR DETECTION OF FORBUSH DECREASE ON SIGNALS OBTAINED FROM SOLAR RADIATION MONITORS*.

Author: Álvaro Alexis Valbuena Delgado**.

Keywords: Machine Learning, Forbush decrease, Support vector machines, Python.

Descripción: In the study of astrophysical phenomena, it has been the great importance that have come to have the tools to achieve understand, and ultimately understand the nature of the world around us. The development of these tools is essential to reaching the knowledge that you can get the mechanism that drives these phenomena.

With the contribution of new techniques that brings the field of machine learning to process information that represents these phenomena, presents a great opportunity to provide tools to help scientists achieve their goals and contribute their knowledge to humanity.

That is why it is necessary, thanks to the techniques offered by the machine learning, develop a series of tools to be implemented in different detectors around the world and help scientists in their study of solar radiation, management and economy as detectors to analyze signals in real time, be stored only detected events and not the entire signal, that is because the systems in which the model is implemented possibly can be Raspberry Pi systems (computers reduced) plate. Thus the application of machine learning is also necessary, because once it is developed and trained the model will then be possible to identify, select and save only the data required for transmission, thus preventing the transfer of large volumes of unnecessary information.

In this paper we propose to develop a predictive model which allows the Halley group in collaboration with the LAGO project, detect and store only the astrophysical events, automating a sorting process which would optimize the storage capacity that has the LAGO project.

* Bachelor thesis

** Physic-mechanical Engineering Faculty. Systems Engineering School. Director Ph.D. Raúl Ramos Pollan.

INTRODUCCIÓN

En el estudio de los fenómenos astrofísicos, se ha visto la gran importancia que han llegado a tener las herramientas informáticas para lograr entenderlos, y al final, entender la naturaleza del mundo que nos rodea. El desarrollo de estas herramientas es imprescindible para lograr alcanzar el conocimiento que se puede obtener del mecanismo que hace funcionar estos fenómenos.

Con el aporte de las nuevas técnicas que trae el campo del machine learning para procesar la información que representa estos fenómenos, se presenta una gran oportunidad para aportar herramientas informáticas que ayuden a los científicos a lograr sus metas y aportar su conocimiento a la humanidad.

Es por esto que se hace necesario, gracias a las técnicas ofrecidas por el machine learning, desarrollar una serie de herramientas informáticas que se implementarán en diferentes detectores alrededor del mundo y ayudarán a los científicos en su estudio de la radiación solar, la gestión y economía de los detectores ya que al analizar las señales en tiempo real, se almacenarán solo los eventos detectados y no la totalidad de la señales, esto es debido a que los sistemas en los que posiblemente se implemente el modelo puedan ser sistemas Raspberry Pi (ordenadores de placa reducida).

Por esto también se hace necesario la aplicación del machine learning, porque una vez se ha desarrollado y entrenado el modelo, será entonces posible identificar, seleccionar y guardar solo los datos necesarios para su transmisión, evitando así la transferencia de grandes volúmenes de información innecesaria.

En este trabajo se propones desarrollar un modelo predictivo el cual permita al grupo Halley en colaboración con el proyecto LAGO, detectar y almacenar solo los eventos astrofísicos, automatizando un proceso de clasificación el cual optimizaría la capacidad de almacenamiento que tiene el proyecto LAGO.

Este documento se estructura en tres capítulos los cuales se relacionan a continuación:

El primer capítulo hace referencia al marco teórico de la temática, en él se indican los conceptos y métodos empleados para el desarrollo del modelo y los métodos usados para validarlo.

El segundo capítulo se estructura la metodología y los procedimientos que ayudan a determinar el modelo predictivo que mejor encaje en el problema.

Finalmente se presentan las conclusiones y recomendaciones producto del desarrollo de este proyecto de investigación.

1 MARCO TEÓRICO

1.1 DECRECIMIENTOS FORBUSH

Los decrecimientos forbush (FD por sus siglas en inglés) son un rápido decrecimiento de los rayos cósmicos detectados debido a un barrido que sufren partículas cargadas que llegan hasta la tierra.

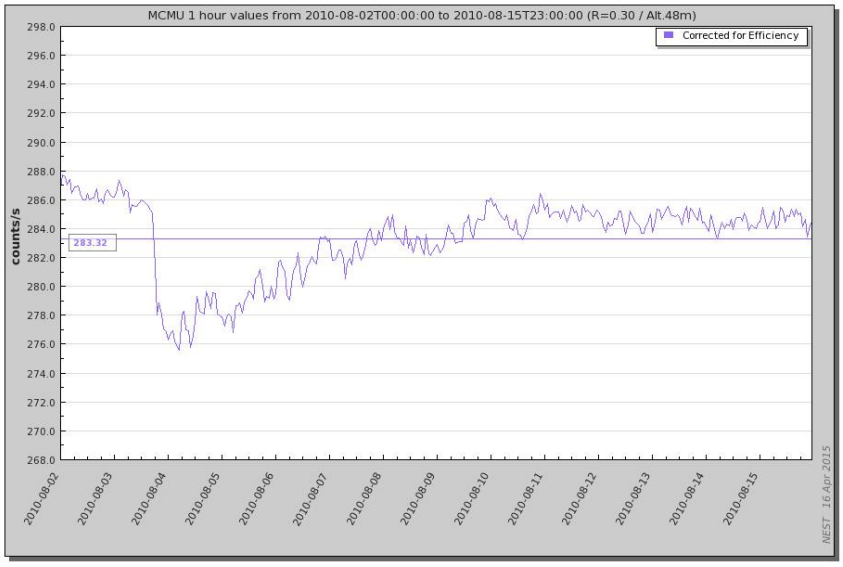
Estos eventos se producen cuando las manchas solares explotan, ya que con frecuencia arrojan lejos del sol grandes cantidades de gas caliente. Estas nubes de gas son llamadas eyecciones de masa coronal (CME por sus siglas en inglés) que también contienen campos de fuerza magnética que desvían las partículas cargadas de modo que cuando una CME pasa más allá de la tierra, barre con la mayor parte de los rayos cósmicos cargados eléctricamente produciendo así un decrecimiento de estos rayos o FD. [1]

Para detectar este tipo de rayos se usan monitores de neutrones (NM) que toman medidas, de acuerdo a la capacidad del monitor, hasta varias veces por segundo, y transfieren estos datos a las terminales para ser tratados y almacenados. Los FD son vistos solo después que los datos son procesados y corregidos (por ejemplo, corrección por presión atmosférica).

Es gracias a colaboraciones de varios grupos de investigadores, que señales de varios detectores a nivel mundial son dispuestas al público en bases de datos en las cuales se pueden descargar de acuerdo a las necesidades propias de cada usuarios. Un ejemplo de esto es NEST [2], una base de datos de señales de

monitores de neutrones, de la que se pueden descargar las señales con opciones de correcciones que el usuario desee aplicar.

Figura 1. Decrecimiento forbush captado por NM McMurdo



Fuente: NEST

1.2 MACHINE LEARNING (ML)

Machine learning o ML es un subcampo de la inteligencia artificial dedicada a los algoritmos que permiten a los computadores aprender. Lo que esto significa, en la mayoría de los casos, es que a un algoritmo se le es dado un conjunto de datos y éste infiere información acerca de las propiedades de los datos la cual, le permite hacer predicciones sobre otros datos que podría ver en el futuro. Esto es posible debido a que casi todos los datos no aleatorios contienen patrones, y estos patrones

le permiten a la máquina hacer generalizaciones. Con el fin de generalizar, entrena un modelo con lo que determina son los aspectos importante de los datos.

Para entender cómo los modelos vienen a ser, considere un ejemplo simple de filtrado de correo electrónico. Suponga que se recibe una gran cantidad de spam que contiene las palabras “farmacia en línea”. Todo ser humano está equipado para reconocer patrones y determinar rápidamente que cualquier mensaje con las palabras “farmacia en línea” es spam y debe ser movido directamente a la basura. Esto es una generalización, de hecho se ha creado un modelo mental de lo que es spam. Después de reportar muchos de estos mensajes como spam, un algoritmo de ML diseñado para filtrar spam debe ser capaz de hacer la misma generalización.

Hay muchos algoritmos de ML diferentes, todos con diferentes fortalezas y equipados para diferentes tipos de problemas. Algunos, como los arboles de decisión, son transparentes por lo que el observador puede entender totalmente el proceso de razonamiento llevado a cabo por la máquina. Otros como las redes neuronales, son “black box”, lo que significa que producen una respuesta, pero es a menudo muy difícil de reproducir el razonamiento detrás de ella. [3]

1.3 ALGUNOS TIPOS DE MACHINE LEARNING

Para abordar problemas con las técnicas de ML, hay un par de cosas que se deben tener en cuenta.

La primera son los datos. Escoger las variables que se quieren usar (las cuales son llamadas features) es una parte importante para encontrar buenas soluciones a los problemas. La segunda es escoger como procesar esa información.

Esta es una aproximación a la definición de aprendizaje en el sentido de mejorar en una tarea a través de la práctica. Esto lleva a una pregunta vital: cómo sabe la maquina si está mejorando o no?

Hay muchas respuestas posibles diferentes a esta pregunta, y estas producen diferentes tipos de ML. Se le puede decir al algoritmo la respuesta correcta para un problema para que lo haga bien una próxima vez. Se espera que solo se le deban decir unas pocas respuestas correctas y entonces puede resolver como obtener las respuestas correctas para otros problemas (generalización). Alternativamente, se le puede decir si la respuesta es o no correcta, pero no la forma de encontrarla, de modo que tiene que **buscar** la respuesta correcta. Una variante de esto es que se le puede dar una puntuación por la respuesta, de acuerdo a lo correcta que es en lugar de solo una respuesta "correcta o incorrecta". Finalmente, se podría no tener ninguna respuesta correcta, solo se quiere que el algoritmo encuentre salidas que tienen algo en común.

Estas diferentes respuestas a esta pregunta proveen una manera útil de clasificar los diferentes algoritmos de los que se estaban hablando:

1.3.1 Aprendizaje supervisado

Un conjunto de datos de entrenamiento con la respuesta correcta (targets), son proporcionados, y con base en este conjunto de entrenamiento, el algoritmo generaliza para responder correctamente a todas las posibles entradas. Esto también es llamado “aprendizaje por ejemplos”.

1.3.2 Aprendizaje no supervisado

Las respuestas correctas no son proporcionadas, en cambio, el algoritmo trata de identificar similitudes entre las entradas de modo que las entradas que tienen algo en común se clasifican juntas. La aproximación estadística al aprendizaje no supervisado es conocida como estimación de densidad. [4]

1.3.3 Aprendizaje por refuerzo

Este tipo de ML permite a las máquinas y agentes de software determinar automáticamente el comportamiento ideal en un contexto específico, con el fin de maximizar su rendimiento. Simples retroalimentaciones de recompensas son necesarias para los agentes para aprender su comportamiento; esto es conocido como la señal de refuerzo.

Hay muchos algoritmos diferentes que abordan este problema. De hecho, el aprendizaje por refuerzo es definido para un tipo específico de problema, y todas sus soluciones se clasifican como algoritmos de aprendizaje por refuerzo. En estos problemas, un agente se supone que decide la mejor acción a seleccionar basado

en su estado actual. Cuando este paso es repetido, el problema es conocido como un proceso de decisión de Markov. [5]

1.4 APRENDIZAJE SUPERVISADO

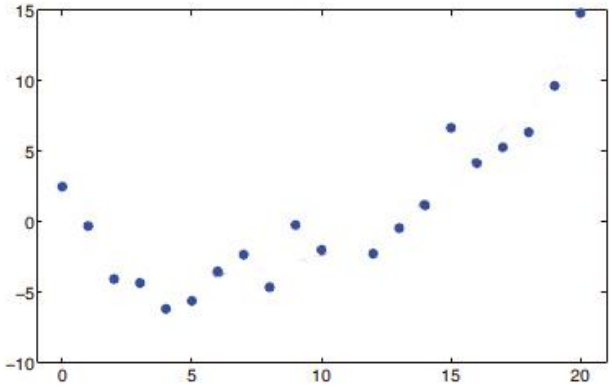
Como se sabe, en el aprendizaje supervisado se tiene un conjunto de datos para los cuales se conocen sus resultados y se desea obtener, al inferir de estos datos, los resultados para nuevos datos (predicciones).

Pero los problemas que se resuelven con los algoritmos del aprendizaje supervisado, manejan diferentes tipos de datos, los cuales, en este tipo de aprendizaje se ubican en una de dos categorías:

1.4.1 Regresión

En los problemas de regresión, los valores que se manejan son de tipo continuo o cuantitativos. En un ejemplo simple, suponga que se tiene un conjunto de datos en los cuales para cada valor real único de entrada $x_i \in \mathbb{R}$ se tiene un valor único de respuesta $y_i \in \mathbb{R}$ en miles ($1x10^3$) (ver figura 2). [6]

Figura 2. Ejemplo de un conjunto de datos

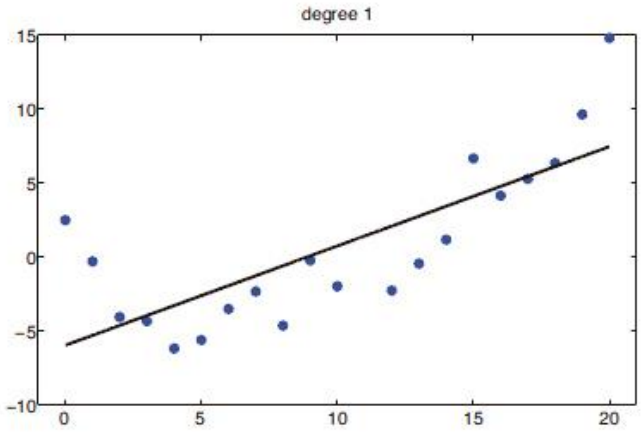


Fuente: Kevin P. Murphy, “Machine Learning: A probabilistic Perspective”, 2012

Dado este conjunto de datos, digamos que se quiere averiguar qué valor de salida puede tener el valor de entrada “11”. Cómo puede ayudar un algoritmo de aprendizaje en este problema?

Una opción que un algoritmo de aprendizaje puede hacer es ajustar una línea recta a través de los datos y basado en esto, predecir el valor que puede tener “11” (ver figura 3), el cual parece ser “1000”.

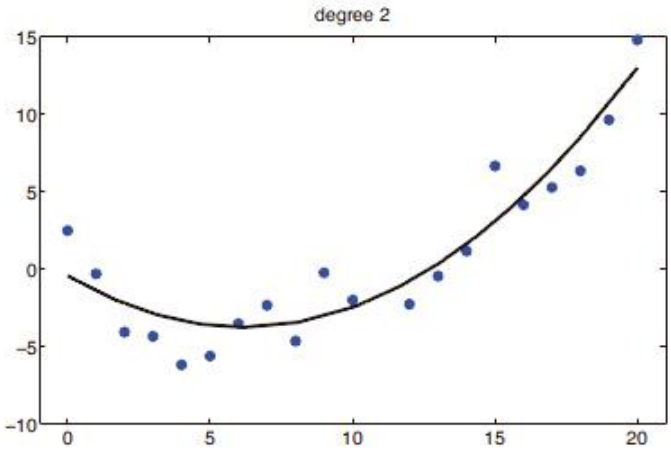
Figura 3. Ajuste con una línea recta



Fuente: Kevin P. Murphy, “Machine Learning: A probabilistic Perspective”, 2012

Pero tal vez, éste no es el único algoritmo de aprendizaje que se pueda usar, puede haber uno mejor. Por ejemplo, en vez de ajustar una línea recta a los datos, se puede decir que sería mejor ajustar con una función cuadrática (ver figura 4). Al hacer esto y hacer la predicción, parece que el resultado de la predicción para el valor “11” puede ser “-2000”. Esto es lo que se denomina un problema de regresión, porque se trata de predecir valores continuos.

Figura 4. Ajuste con una función cuadrática



Fuente: Kevin P. Murphy, “Machine Learning: A probabilistic Perspective”, 2012

1.4.2 Clasificación

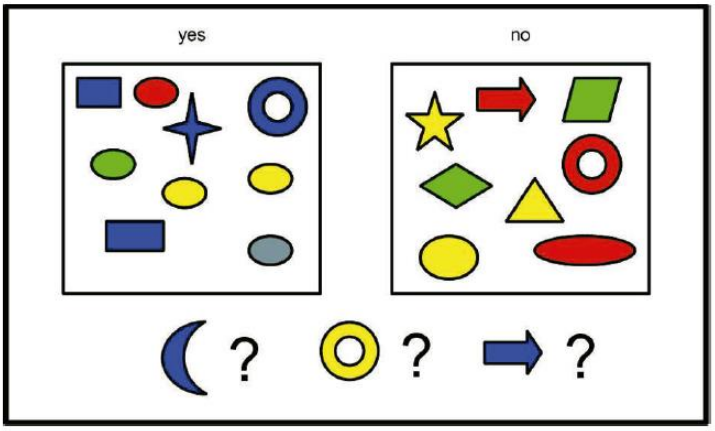
En los problemas de clasificación, los valores que se predicen son de tipo cualitativo [7]. En estos problemas, el objetivo es aprender un mapeo desde las entradas x a las salidas y, donde $y \in \{1, \dots, C\}$, y en donde C es el número de clases. Si $C = 2$, el esto es llamado clasificación binaria (en cuyo caso a menudo se asume $y \in \{0,1\}$). Si $C > 2$, esto es llamado clasificación multiclase. Si las etiquetas de clases no son excluyentes entre sí (por ejemplo, alguien puede ser clasificado como alto y fuerte), se llama clasificación multi-etiqueta, pero esto se ve mejor como la predicción múltiple de etiquetas de clase binaria relacionadas (también llamado modelo de múltiples salidas).

Una manera de formular el problema es como una función de aproximación. Se asume $y = f(x)$ para alguna función desconocida f, y el objetivo del aprendizaje es

estimar la función f dado un conjunto de entrenamiento etiquetado, luego se hacen predicciones usando $y = f(x)$. El principal objetivo es hacer predicciones sobre las entradas nuevas, es decir, las que no se han visto antes (esto se llama generalización), ya que la predicción de la respuesta en el conjunto de entrenamiento es fácil (basta solo con mirar la respuesta). [6]

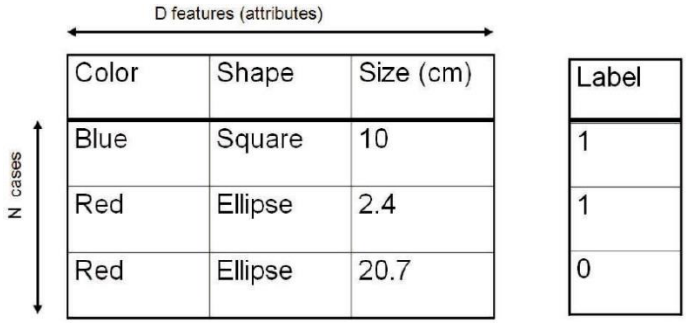
Como un ejemplo simple de clasificación, considere que se tiene dos clases de objetos (ver figura 4), los cuales corresponden a dos etiquetas 0 y 1. Las entradas son formas de colores. Estas son descritas por un conjunto de D características (features) o atributos, los cuales son guardados en una matriz X de N x D (ver figura 5). Las características de entrada x pueden ser discretas, continuas o una combinación de las dos. Además de las entradas, se tiene un vector de entrenamiento con etiquetas y.

Figura 5. Conjuntos de elementos etiquetados y 3 elementos no etiquetados



Fuente: Kevin P. Murphy, “Machine Learning: A probabilistic Perspective”, 2012

Figura 6. Matriz de N x D que representa el conjunto de entrenamiento



Fuente: Kevin P. Murphy, “Machine Learning: A probabilistic Perspective”, 2012

En la figura 4, los casos de prueba son una media luna azul, una rueda amarilla y una flecha azul. Ninguna de ellas se ha visto antes, por lo tanto hay que generalizar más allá del conjunto de entrenamiento.

Una conjetura razonable es que la media luna azul debe ser $y = 1$, ya que todas las formas azules tienen etiqueta 1 en el conjunto de entrenamiento. La rueda amarilla es difícil de clasificar, ya que algunas formas amarillas tienen etiqueta $y = 1$ y algunos se etiquetan $y = 0$, y algunos círculos se etiquetan $y = 1$ y otros $y = 0$. En consecuencia, no está claro cuál debe ser la etiqueta correcta en el caso del círculo amarillo. Del mismo modo, la etiqueta correcta para la flecha azul no es clara. Para resolver estos casos ambiguos, es deseable retornar una probabilidad con la cual, se calcula la estimación de la que sería la etiqueta correcta.

1.5 ALGORITMOS DE CLASIFICACION

1.5.1 Support vector machines (SVM)

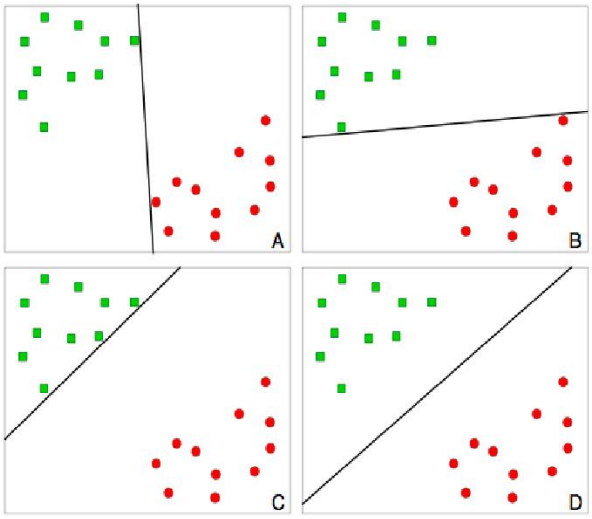
En los support vector machines (SVM), la idea básica es encontrar un hiperplano el cual separa los datos d-dimensionales perfectamente dentro de sus clases. Sin embargo, como ejemplo, los datos a menudo no son linealmente separables, las SVM's introducen la noción de un "kernel de espacio de características inducidas" el cual arroja los datos a un espacio de dimensiones superiores, donde son separables [8]. Considere un típico problema de clasificación. Algunos vectores de entrada (vectores de características) y algunas etiquetas son provistas. El objetivo del problema de clasificación es predecir las etiquetas de nuevos vectores de entrada de modo que la tasa de error de la clasificación sea mínima.

Hay muchos algoritmos para resolver tales problemas. Algunos requieren que los datos de entrada sean linealmente separables. Pero para muchas aplicaciones esta suposición no es apropiada. E incluso si se mantiene esta suposición, la mayoría de las veces hay muchas posibles soluciones para el hiperplano [9] (ver figura 6).

Un óptimo hiperplano es definido como la función de decisión lineal con máximo margen entre los vectores de las dos clases (ver figura 7), se puede observar que para construir tales hiperplamos óptimos, solo se tiene que tener en cuenta una pequeña cantidad de los datos de entrenamiento, los llamados "vectores de soporte", los cuales determinan estos márgenes. Se demostró que si los vectores de entrenamiento son separados sin errores por un hiperplano, el valor esperado de la probabilidad de cometer error en un test de ejemplo está limitada por la relación

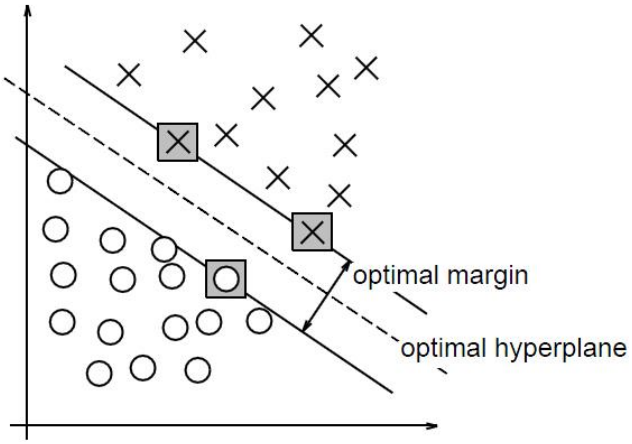
entre el valor esperado del número de vectores de soporte y el número de vectores de entrenamiento. [10]

Figura 7. Posibles soluciones al hiperplano



Fuente: http://mindthegap.googlecode.com/files/AnotherIntroductionSVM.pdf

Figura 8. Ejemplo de un problema separable en un espacio bidimensional



Fuente: http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf

1.5.2 Árboles de decisión

Para ilustrar como funciona un árbol de decisión, considere una versión simple de un problema de clasificación de vertebrados. En vez de clasificar en distintos grupos de especies, se clasifican los vertebrados como mamíferos y no mamíferos.

Suponga que una nueva especie es descubierta por científicos. Cómo se le puede decir a los científicos si es o no un mamífero? Una aproximación es plantear una serie de preguntas acerca de las características de la especie. La primera pregunta que se debe plantear es si la especie es de sangre fría o caliente. Si es de sangre fría entonces definitivamente no es mamífero. De lo contrario es un ave o un mamífero. En este último caso hay que hacer una pregunta de seguimiento: Las hembras de la especia dan a luz a sus crías?. Aquellas que dan a luz son definitivamente mamíferos, mientras aquellas que no, parecen ser que no son mamíferos (con la excepción de los mamíferos que ponen huevos tales como el ornitorrinco).

El ejemplo anterior ilustra cómo se resuelve un problema de clasificación haciendo una serie de preguntas cuidadosamente elaboradas sobre el registro de pruebas. Cada vez que se recibe una respuesta, una pregunta de seguimiento se hace para llegar a una conclusión sobre la etiqueta de la clase del registro. La serie de preguntas y sus posibles respuestas se pueden organizar en forma de un árbol de decisión, que es una estructura jerárquica que consta de nodos y aristas dirigidas. En la figura 8 se muestra el árbol de decisión para este ejemplo. Los arboles de decisión tienen tres tipos de nodos

- Un **nodo raíz** que no tiene aristas entrantes y cero o más aristas salientes.

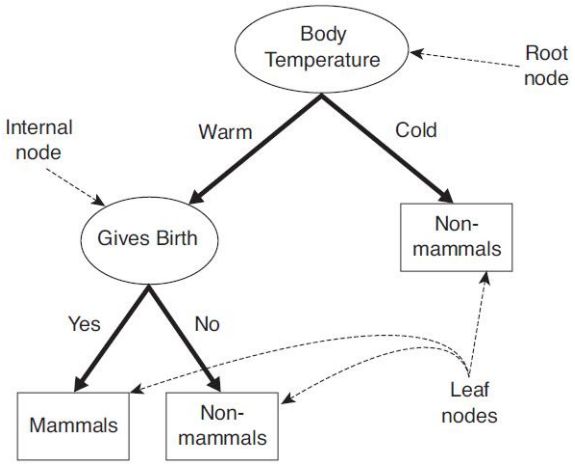
- **Nodos internos**, cada uno de los cuales tiene exactamente una arista entrante y dos o más saliendo.

- **Hoja** o **nodos terminales**, cada uno de los cuales tiene exactamente una arista entrante y no tiene aristas de salida.

En los arboles de decisión, a cada nodo hoja se le asigna una etiqueta de clase. Los nodos no terminales, los cuales incluyen la raíz y otros nodos internos, contienen condiciones de prueba de atributos para separar registros que tienen diferentes características. Por ejemplo, el nodo raíz en la figura 8, usa el atributo “temperatura del cuerpo” para separar vertebrados de sangre caliente de los de sangre fría. Ya que todos los vertebrados de sangre fría no son mamíferos, un nodo hoja etiquetado como “no mamífero” es creado como nodo hijo a la derecha del nodo raíz. Si el vertebrado es de sangre caliente, un atributo subsecuente, “da a luz”, es usado para distinguir mamíferos de otras criaturas de sangre caliente, que son más que todo aves.

La clasificación de un registro de prueba es sencillo una vez que un árbol de decisión se ha construido. Empezando desde el nodo raíz, se aplica la condición de prueba al registro y se sigue a la rama apropiada con base a la salida del test. Esto lleva a otro de los nodos internos en el cual una nueva condición de prueba es aplicada, o a un nodo hoja. La etiqueta de clase asociada al nodo hoja es entonces asignada al registro. [11]

Figura 9. Árbol de decisión para el problema de clasificación de mamiferos

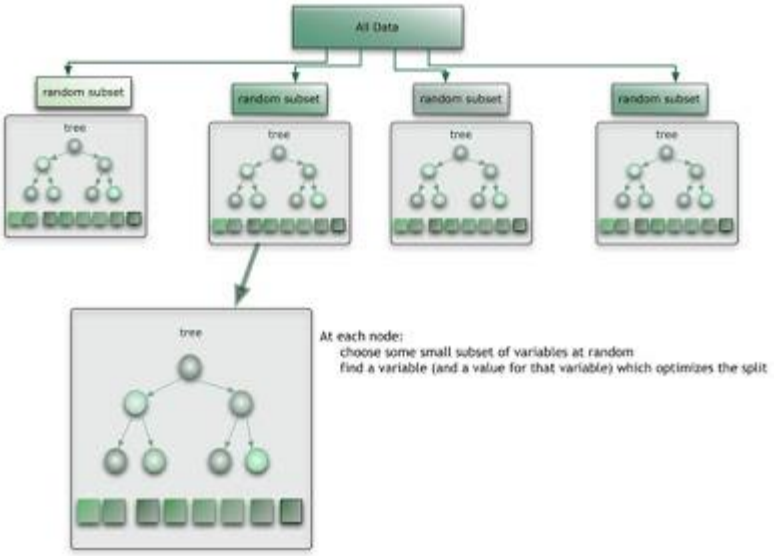


Fuente: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

1.5.3 Random forest

Random forest es una combinación de árboles predictores en la que cada árbol depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de estos.

Figura 10. Ejemplo visual de la interpretación de un random forest



Fuente: http://randomforest2013.blogspot.com.es/2013/05/randomforest-definicion-random-forests.html

Este algoritmo mejora la precisión en la clasificación mediante la incorporación de aleatoriedad en la construcción de cada clasificador individual. Esta aleatorización puede introducirse en la partición del espacio (construcción del árbol), así como en la muestra de entrenamiento. El Random Forest comienza con una técnica de aprendizaje automático estándar llamada “árbol de decisión”, que, en cuanto al conjunto, corresponde a un aprendizaje. En un árbol de decisión, una entrada se introduce en la parte superior y hacia abajo a medida que atraviesa el árbol de los datos, los cuales se acumulan en conjuntos más pequeños.

El Método Random Forest es fácil de aprender y de usar, tanto por profesionales como por personas con menos experiencia. Con poca investigación y tiempo de desarrollo, puede ser utilizado por personas con poca base estadística. En pocas

palabras, este método permite hacer de manera segura predicciones más precisas y sin la mayoría de los errores básicos comunes a otros métodos. [12]

Principales beneficios:

- Precisión.
- Funciona de manera eficiente con bases de datos de gran tamaño.
- Maneja miles de variables de entrada sin necesidad de borrado.
- Da estimaciones de qué variables son importantes en la clasificación.
- Genera una estimación objetiva interna de la generalización de errores.
- Proporciona métodos eficaces para estimar datos que faltan.
- Los “bosques” generados se pueden guardar para uso futuro en otros datos.
- Los prototipos se calculan de manera que proporcionan información acerca de la relación entre las variables y la clasificación.

- Calcula proximidades entre pares de casos que se pueden utilizar en la agrupación, la localización de los valores extremos, o (en escala) dan interesantes vistas de los datos

- Ofrece un método experimental para la detección de interacciones de variables.

1.6 HERRAMIENTAS PARA LA SELECCIÓN Y EVALUACION DE MODELOS

1.6.1 Grid search

Grid search es usado para optimizar los parámetros de un clasificador. Estos parámetros que no están directamente aprendidos dentro de los clasificadores, pueden ser configurados mediante la búsqueda de un espacio de parámetros para la mejor puntuación del cross-validation. Cualquier parámetro proporcionado al construir un estimador (o clasificador), puede ser optimizado de esta forma.

Tales parámetros se refieren a menudo como hiperparámetros (particularmente en el aprendizaje bayesiano), distinguiéndolos de los parámetros optimizados en el procedimiento de ML.

Un grid search consiste de:

- Un estimador (como un árbol de decisión)

- Un espacio de parámetros

- Un método de busqueda
- Un esquema de cross-validation
- Una función de puntuación. [15]

1.6.2 K-fold cross-validation

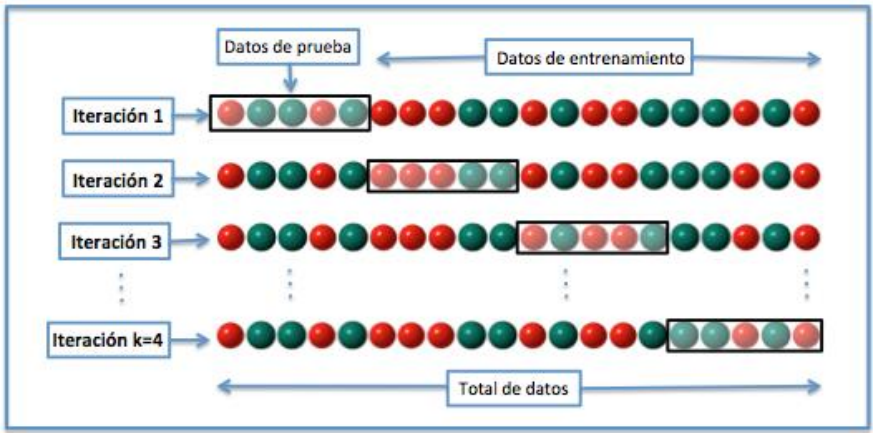
Es común que en los problemas de clasificación, el conjunto de datos etiquetados para entrenamiento y validación no sea lo suficientemente grande. La solución para esto es utilizar el método k-fold cross validation, el cual funciona dividiendo los datos etiquetados en *K* grupos de datos iguales X_i, $i = 1,\ldots,K$. Genera cada grupo, conservando una de las partes como un conjunto de validación y combina las restantes *K-1* partes para formar el conjunto de entrenamiento. Hace esto *K* veces, y cada vez que lo hace, saca a parte una de las *K* partes para obtener *K-1* grupos. [13][14]

Figura 11. Conjunto de datos divididos en K pares

$$\begin{aligned} \mathcal{V}_1 &= \mathcal{X}_1 \quad & \mathcal{T}_1 &= \mathcal{X}_2 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ \mathcal{V}_2 &= \mathcal{X}_2 \quad & \mathcal{T}_2 &= \mathcal{X}_1 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K \\ &\vdots \\ \mathcal{V}_K &= \mathcal{X}_K \quad & \mathcal{T}_K &= \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_{K-1} \end{aligned}$$

Fuente: Ethem Alpaydin, “Introduction to Machine Learning”, 2014

Figura 12. Validación cruzada de k-fold con k = 4

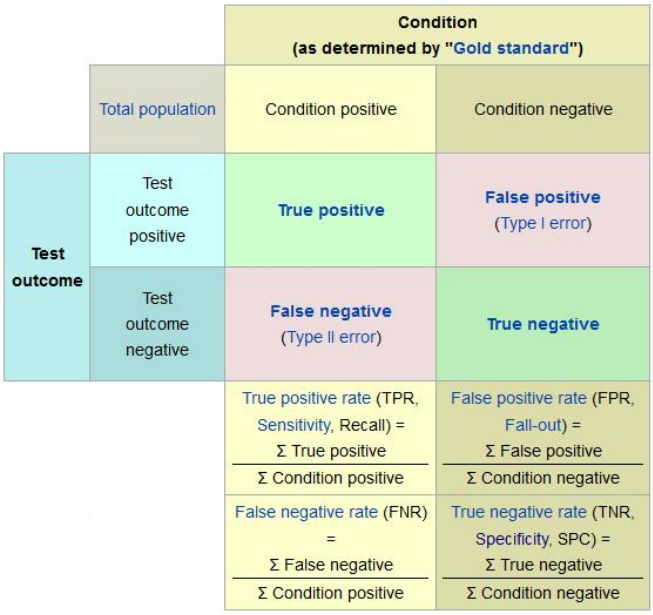


Fuente: http://es.wikipedia.org/wiki/Validaci%C3%B3n_cruzada

1.6.3 Sensibilidad y especificidad

Son medidas estadísticas del desempeño de una prueba de clasificación binaria. La sensibilidad mide la habilidad del modelo para clasificar un registro positivamente, mientras que especificidad mide la habilidad de clasificar un registro negativamente. Con estas medidas se puede construir lo que se conoce como matriz de confusión.

Figura 13. Matriz de confusión



		Condition (as determined by "Gold standard")	
	Total population	Condition positive	Condition negative
Test outcome	Test outcome positive	**True positive**	**False positive** (Type I error)
	Test outcome negative	**False negative** (Type II error)	**True negative**
		True positive rate (TPR, Sensitivity, Recall) = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR, Fall-out) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$
		False negative rate (FNR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR, Specificity, SPC) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$

Fuente: http://en.wikipedia.org/wiki/Sensitivity_and_specificity#Sensitivity

En la cual:

- True positive (TP): valores correctamente identificados
- False positive (FP): valores incorrectamente identificados
- True negative (TN): valores correctamente rechazados
- False negative (FN): valores incorrectamente rechazados

Por lo tanto se puede definir:

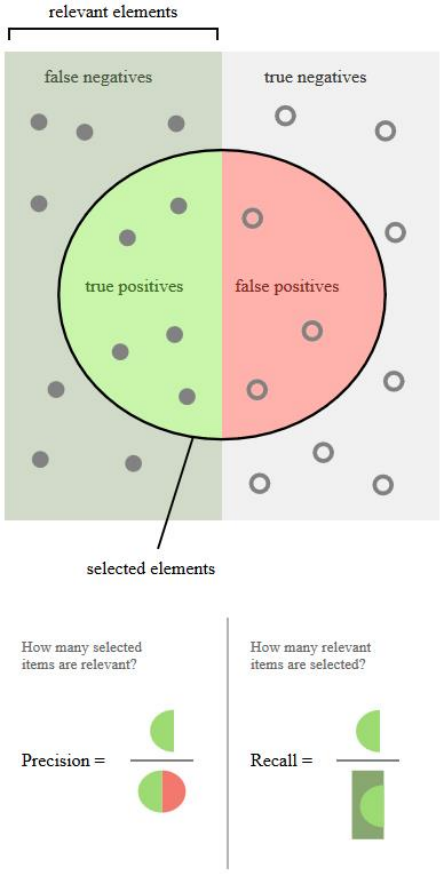
$$Sensibilidad = \frac{TP}{TP + FN}$$

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

1.6.4 Precisión y exhaustividad

Es una métrica del rendimiento de sistemas de búsqueda y recuperación de información y reconocimiento de patrones. En un entorno de recuperación de datos, un sistema puede recuperar algunos datos registros relevantes (TP) pero probablemente no todos (FN); también podría erróneamente recuperar datos no relevantes (FP). Entonces, *precisión* es el número de registros recuperados y que son relevantes, dividido por el número total de registros recuperados. Si precisión es 1, todos los registros recuperados pueden ser relevantes, pero puede haber registros que son relevantes pero no recuperados. Exhaustividad es el número de registros relevantes recuperados, dividido por el número total registros relevantes, incluso si exhaustividad es 1, todos los registros relevantes pudieron ser recuperados pero también debe haber registros irrelevantes que son recuperados.[14]

Figura 14. Precisión y exhaustividad



Fuente: http://en.wikipedia.org/wiki/Precision_and_recall

2 DESARROLLO DEL PROYECTO

2.1 METODOLOGÍA

El desarrollo del proyecto se realizará guiado por los principios de las metodologías ágiles. Estos métodos de ingeniería del software están basados en el desarrollo iterativo e incremental, esto quiere decir que el desarrollo del software se hace con interacciones llamadas sprints en las que se analiza, diseña, desarrolla y se produce una versión del software que cuenta con funcionalidades desarrolladas en el sprint y que con cada sprint se producen y se van agregando cada vez más funcionalidades hasta completar el software. La idea de esta metodología es que con cada sprint se produzca una pequeña parte del software que pueda ser probada por el cliente de modo que si hay errores estos sean corregidos en el siguiente sprint, además se desarrollan e implementan nuevas funcionalidades al software y se produce otra versión de este más completa y mejorada con las correcciones de los errores que el cliente haya detectado.

Utilizando estas técnicas en éste proyecto de grado, el trabajo se divide en paquetes de trabajo (PT) y, dentro de cada PT, varias actividades (A) que produce entregables (E)

2.1.1 PT1: Exploración preliminar datos Forbush

Actividades

A1.1: Obtención y normalización de datos

Se obtienen los datos de los detectores, y se procede a hacerles un análisis inicial.

A1.2: Prototipo bolsa de características (BoF)

Se implementa la bolsa de caracteristicas sobre los datos para producir los datasets

A1.3: Uso preliminar de clasificadores

Se eligen e implementan los primeros clasificadores para observar su desempeño.

Entregables

E1.1: Scripts cálculo representación BoF

E1.2: Reporte resultados preliminares de clasificación

2.1.2 PT2: Construcción del framewok

Actividades

A2.1: Diseño base del framework

Se especifican las pautas a seguir en el desarrollo del proyecto

A2.2: Iteración 1: (revisión diseño, implementación y prueba).

Se produce la primera versión testeable (Framework v0.1) que tentativamente incluirá funcionalidades como normalización de datos, aplicación del BoF y producción de datasets

A2.3: Iteración 2: (revisión diseño, implementación y prueba)

Se hace revisión del diseño para corregir errores, se continúa con el desarrollo de las funcionalidades del software y se hace otra entrega testeable (Framework v.0.2).

A2.4: Iteración 3: (revisión final del diseño, implementación, prueba y creación del modelo)

Se hacen los últimos cambios del sistema, se desarrollan e implementan las últimas funcionalidades del sistema y se crea el modelo para su posterior evaluación.

Entregables

E2.1: Bosquejo del Framework

E2.2: Framework v0.1 (funcionalidad parcial pero ejecutable)

E2.3: Framework v0.2 (funcionalidades corregidas y adicionales)

E2.4: Framework v1.0

2.1.3 PT3: Experimentación

Actividades

A3.1: Experimentación y recogida de resultados

Se desarrolla el modelo y se pone a prueba. Se recogen los resultados para su validación.

Entregables

E3.1: Reporte del framework desarrollado.

E3.2: Reportes experimentación (desempeño de detección de Forbush)

2.2 PLANTEAMIETO DEL PROBLEMA

El grupo Halley en colaboración con el proyecto LAGO, en su estudio de eventos Forbush (FD), tienen a su disposición una red de monitores ubicados latitudinalmente y de forma estratégica por Centro y Sur América [16]. Se dice que es de forma estratégica debido a que esta la necesidad de ubicar los monitores en lugares de gran altitud ya que esto permite al detector, tomar una mejor medida de la energía de las partículas. Otro hecho que hay que tener en cuenta es que los FD son detectados con menor intensidad en los detectores que se encuentran más cercanos al eje del Ecuador, debido a que las partículas se ven afectadas por el campo magnético de la tierra.

Actualmente, el proyecto LAGO está en proceso de expansión de su red de monitores en el cual busca tener un despliegue que abarca desde Centro América hasta la Antártida.

Esto acarrea 2 problemas para el proyecto. El primero es que aunque se quiere aumentar el despliegue de la red de monitores, no se cuenta con una gran capacidad de almacenamiento de los datos que constantemente están produciendo los detectores. El segundo problema es que debido a las necesidades de ubicar los detectores a lo largo del globo a nivel latitudinal, y en lugares con gran altitud, se hace difícil trasladarse a la ubicación de los detectores para el monitoreo de las señales obtenidas en busca de eventos Forbush FD.

En este trabajo, se planteó la necesidad de desarrollar por medio de algoritmos de machine learning, un modelo predictivo que permita detectar eventos forbush en señales procesadas de los detectores disminuyendo así la necesidad de guardar grandes cantidades de datos, optimizando la capacidad de almacenamiento que se

tiene al dejar solo las porciones de señales que poseen FD. Otra ventaja que se obtiene es que el modelo al ser entrenado con la señal de un detector, aprende a detectar un FD en esa señal (siempre y cuando sea posible su detección), sin importar la ubicación del detector sobre el globo.

2.3 PLANTEAMIENTO DE LA SOLUCION

Al inicio de este proyecto, el grupo Halley proveyó 3 señales obtenidas de 3 detectores. Uno del proyecto LAGO, “lago.dat”, una señal de un monitor solar, “solar-auger.dat”, y una señal del detector McMurdo “mcmu.dat”. Estos señales tienen el tiempo marcando en Unix timestamp. Es decir, que es el tiempo en segundos que han transcurrido desde la media noche UTC del 1 de enero de 1970 [17]

2.3.1 Análisis de las señales obtenidas

Se procedió entonces a graficar las señales para tener una idea general de su forma y características y detectar posibles anomalías.

Figura 15. Señal “lago”

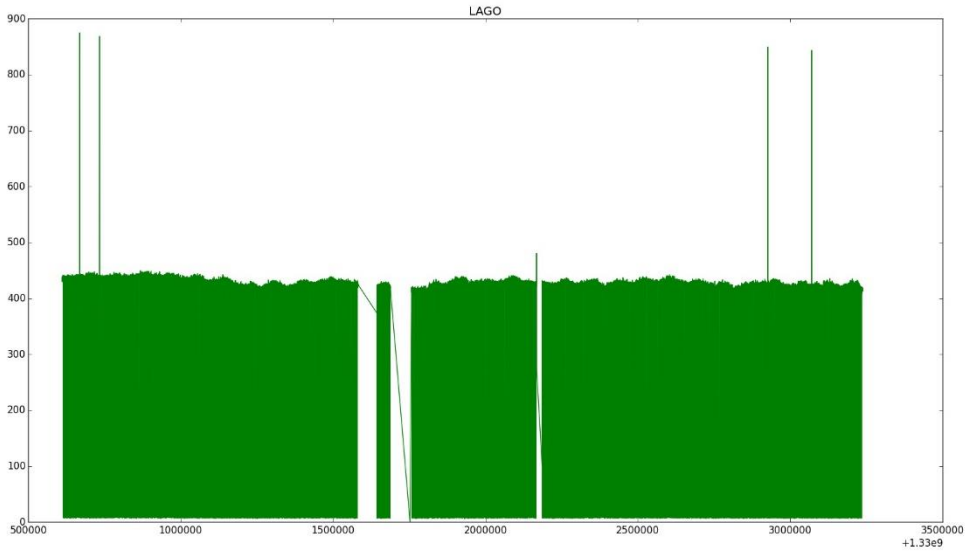


Figura 16. Señal “solar”

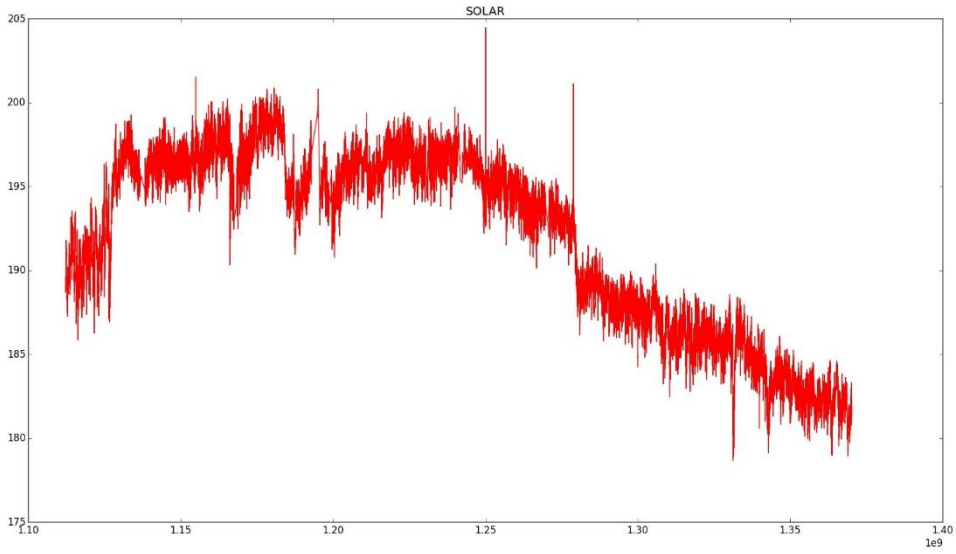
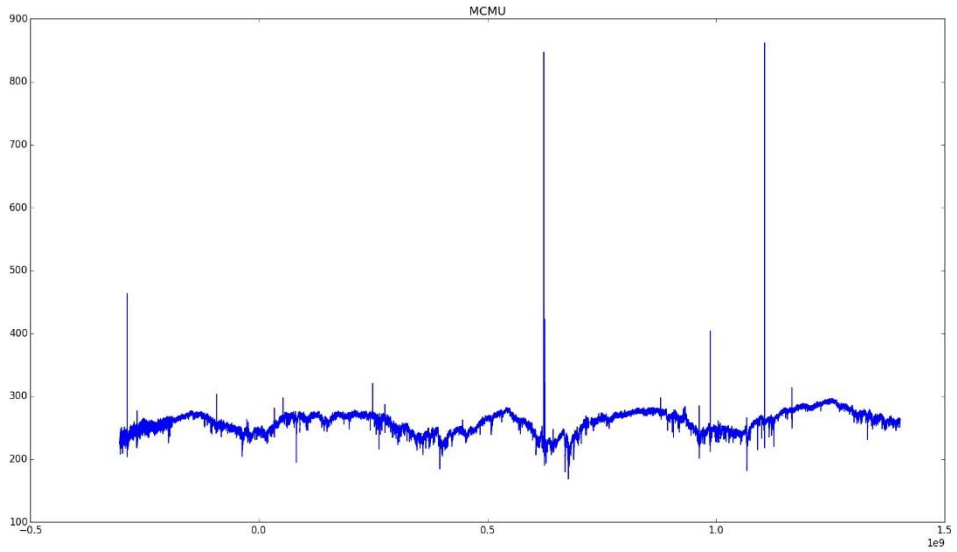


Figura 17. Señal “mcmu”



Se puedo observar que la señal de lago tiene demasiado ruido (figura 14). Para intentar arreglar esto, se creó un script que elimina estos picos al dejar solo los datos que están en un rango. Para saber este rango en la señal de lago, hay que hacer una ampliación a la señal para observar el rango en el que se encuentran la mayoría de los datos. Al hacer esto, se puede notar que los valores en los que se puede definir el rango, valor mínimo = 400, valor máximo = 500. (Ver figura 17)

La señal “solar” no tiene picos como la señal “lago”, pero se aplicó un suavizado para normalizarla y lo mismo se hizo con la señal “mcmu”. Una vez terminado el análisis sobre las señales, se procede a aplicar el “prototipo bolsa de características”.

Figura 18. Señal lago ampliada

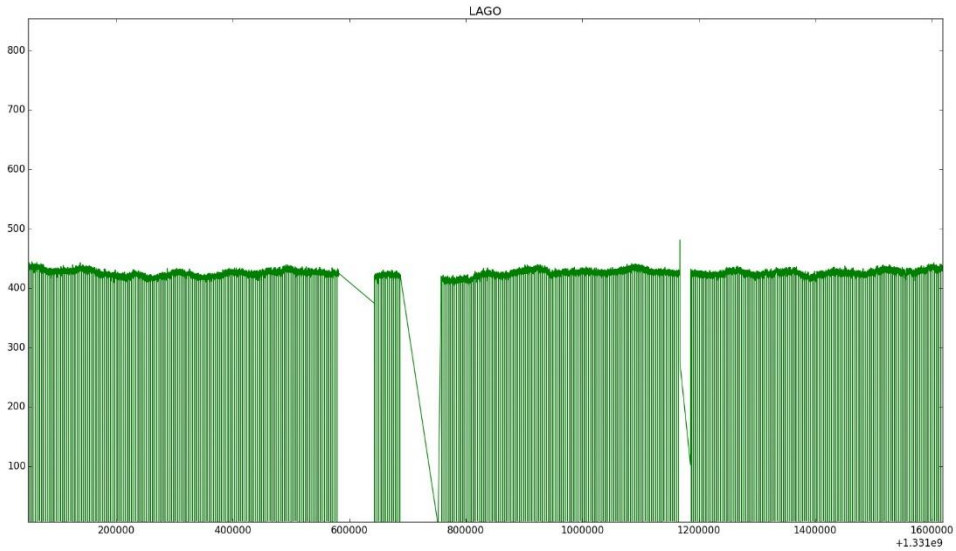
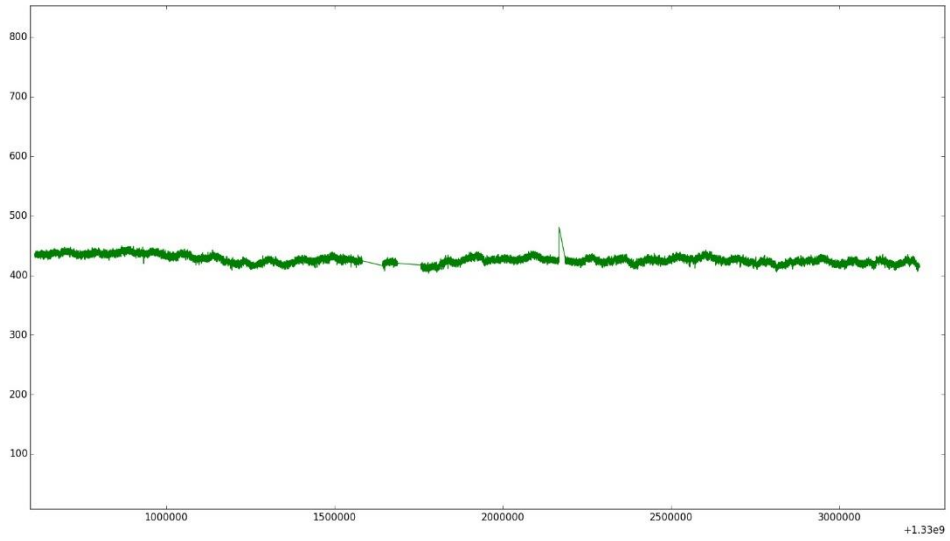


Figura 19. Señal lago normalizada



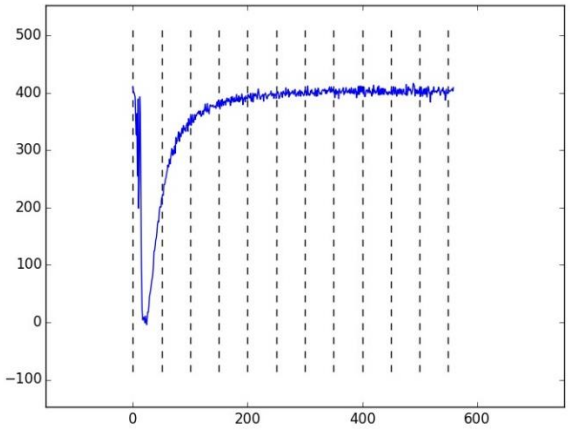
2.3.2 Prototipo bolsa de características (Bof)

Este es el proceso por el cual se transforma la señal para producir una representación de las características de la señal de modo que pueda ser procesada por un algoritmo de ML.

En éste proceso se inició con una discretización de la señal. El proceso de discretización se hizo de la siguiente manera:

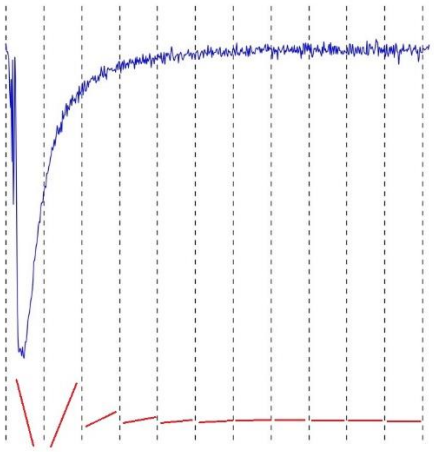
Se divide la señal en fragmentos pequeños llamados “subventanas”, Luego se crean grupos de subventanas los cuales fueron llamados “ventanas”. Las ventanas y subventanas pueden ir superpuestas.

Figura 20. Division de un fragmento de señal en subventanas y todas forman una ventana



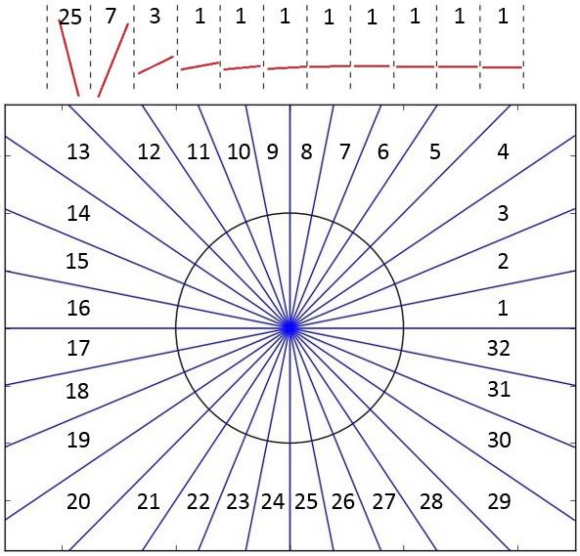
Luego, a cada subventana se le aplica una regresión lineal para obtener la pendiente de la recta que representa a esa subventana.

Figura 21. Regresión lineal aplicada a cada subventana



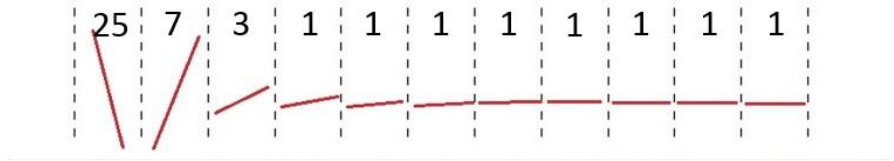
Una vez calculadas las pendientes, se discretizan sobre un plano el cual es dividido en un numero definido de cuadrantes.

Figura 22. Discretización sobre un plano dividido en 32 cuadrantes



Se crea un vector de tamaño *n,* en donde *n* es el número de cuadrantes en los que se dividió el plano, y el valor de cada uno de sus elementos, es la suma de pendientes que cayeron en ese plano. Nótese que cada vector representa una ventana.

Figura 23. Ejemplo de creación de un vector



8,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0

Se procede ahora a hacer pruebas del método sobre las señales para analizar los resultados.

En estos resultados se observó que casi todas las pendientes quedaban en solo 2 cuadrantes (ver ANEXO A), no importa que tan grande sean los tamaños de las ventanas y subventanas, el resultado es siempre el mismo. Esto es debido a que la señal usa el formato de tiempo Unix, el cual hizo que el rango de valores que toma el tiempo sea extremadamente grande en comparación con el rango de valores que toman las medidas. Para solucionar esto basta solo con escalar la señal de modo que el rango de los valores en un eje (el tiempo), no sea excesivamente grande al rango de valores que toman las medidas.

Ya solucionado el problema con las pendientes en el discretizado, se opta por agregar otros valores a los vectores de la discretización, que ayudarán en las métricas de desempeño de los clasificadores

Estos valores son:

- El dato inicial y final de la porción de señal que representa cada ventana.

0,432,8,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0

- EL numero ID del FD que representa. Nótese que para esto se necesita un archivo llamado “Ground Truth” en el cual se especifican los valores que delimitan a los FD .

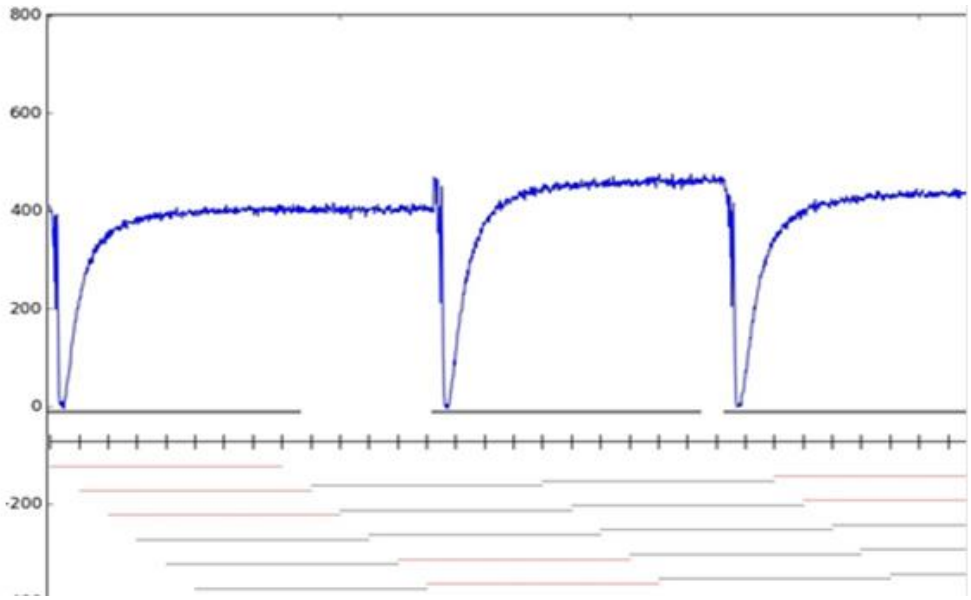
0,432,7,8,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0

- Una etiqueta (1 o 0) que indica si la ventana tiene un porcentaje *p* mínimo dentro de un FD. Por ejemplo, si *p* = 80, quiere decir que si el 80% o más de una ventana está dentro de un FD, el vector que representa a la ventana se etiqueta con 1, de lo contrario se etiqueta con 0.

0,432,7,8,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1

Al final se obtiene un dataset con las características de cada ventana con respecto a las pendientes que la componen y demás datos que indican su posición en la señal y si representa un evento FD o no.

Figura 24. Ejemplo de una señal en la que se muestra la división en subventanas y en ventanas



2.3.3 Archivo Ground Truth

Este archivo contiene los datos que delimitan cada uno de los FD. Son 3 valores (separados por comas) los que describen la duración y el tiempo en el que se presentaron estos eventos, así como un valor ID que identifica a cada evento, Este archivo es de extensión .csv.

Este archivo está estructurado de la siguiente forma: el primero es el ID del evento, el segundo valor corresponde al dato que marca el inicio del evento y el último valor corresponde al dato que marca el final de un FD. En la tabla 1 se muestra un ejemplo. Hay que tener en cuenta que los datos de inicio y final están en Unix Timestamp, pero pueden estar en cualquier otro formato. Lo único que importa es que el formato del tiempo tanto en las señales como en el archivo ground truth sean el mismo.

Tabla 1. Estructura del archivo “Ground Truth”

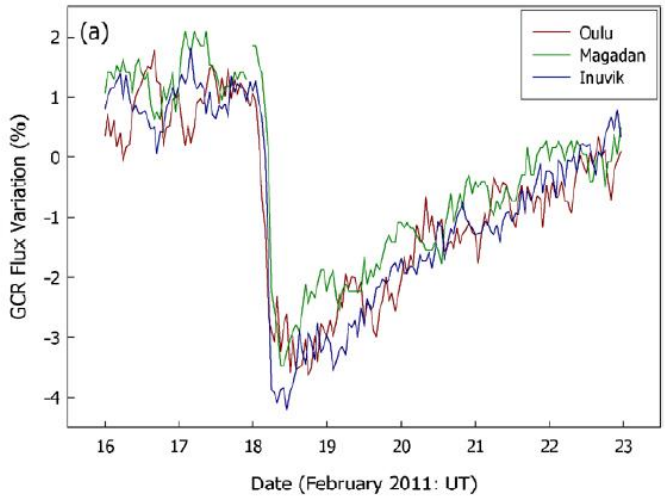
ID del evento	Inicio del FD	Final del FD
1	-315619200	-314755200
2	-313027200	-311472000
3	-299376000	-290563200

Desafortunadamente, en este punto del proyecto, no se contaba todavía con los datos que marcaban los FD de las señales de los detectores, por lo que no se podía crear el archivo ground truth lo que hacía imposible hacer pruebas para escoger clasificadores. Para solucionar esto momentáneamente, se creó un script que simulaba una señal con eventos forbush el cual produce 2 archivos, uno con la señal simulada y otro que es el ground truth. Una vez obtenidos estos archivos, se procedió a hacer pruebas para escoger el clasificador.

2.3.4 Señal simulada

Para crear la señal simulada se comenzó con la búsqueda de una función que simulara un evento forbush.

Figura 25. Evento forbush



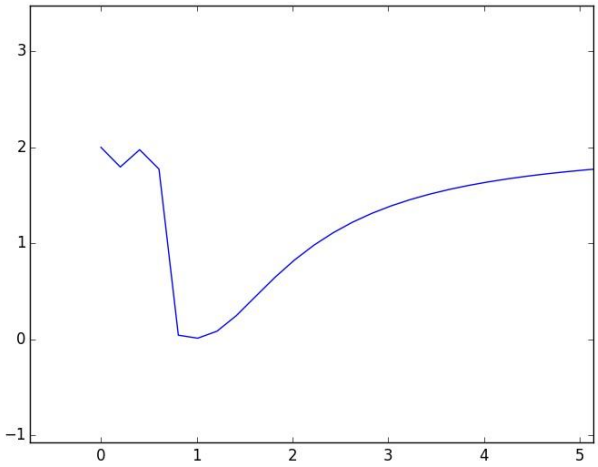
Fuente: https://tallbloke.wordpress.com/2012/08/13/forbush-decreases-caused-by-cmes-are-globally-simultaneous/

Una función que puede representar un evento forbush es:

$$\cos\left(\sin\left(\frac{1}{(-x+.01)^2}\right)^2 * 3.53\right)$$

cuya grafica es

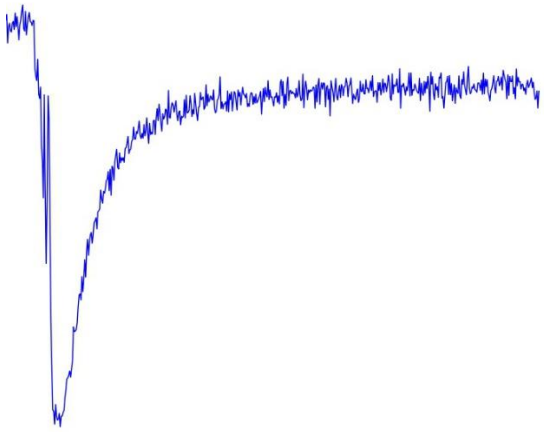
Figura 26. Función que simula evento forbush



Una vez encontrada la función, se procede a crear un script que permita manipular esta función de modo que permita multiplicarla y que cada elemento multiplicado permita escalarlo en tanto en altura como longitud. Entre cada FD simulado se agregar un separador, una línea recta que también pueda ser escaldo en cuanto a su longitud de modo que simule un tiempo largo o corto entre cada evento.

Para finalizar, se le agrega un nivel de noise, de modo que tenga la apariencia de una señal obtenida de un detector y se guarda la señal y su ground truth.

Figura 27. Forbush decrease (FD) simulado



2.3.5 Selección del clasificador

Para resolver este problema de clasificación, se evaluaron 3 clasificadores, el SVC, decision tres y random forest, con el método cross-validation (CV), y un valor de k-fold = 10. Esto quiere decir que el CV evalua iterativamente cada clasificador 10 veces y obtendrá una medida de la clasificación. Para obtener el valor final de la puntuación para cada clasificador, se calcula la media (mean) y la desviación estándar (std) con los 10 valores obtenidos por el CV para cada clasificador

Tabla 2. Resultados de clasificación

	SVM	**Decision trees**	**Random Fores**
CV-mean	0.8650	0.8555	0.8514
CV-std	0.038	0.032	0.035

Con los resultados obtenidos, se optó por usar el algoritmo de SVM al tener un mejor desempeño en la clasificación aun cuando no se ha hecho la búsqueda de sus parámetros óptimos. Esto indica que este algoritmo de clasificación es el que mejor encaja en este problema ya que puede tener un mejor nivel de precisión, y desempeño, ya que es el que hace un mejor ajuste sobre el dataset obtenido de la discretizacion.

Acá se completa el paquete de trabajo 1, en el que se hace entrega del script que produce la discretización y el reporte con los resultados de los clasificadores.

En este punto ya se cuenta con el desarrollo total del script que hace la discretización pero aún no se provee de los datos que definen los FD para poder usar las señales de los detectores. Por lo tanto se opta por continuar el desarrollo del proyecto usando el script que produce señales simuladas.

2.3.6 Barrido de parámetros del clasificador

La barrida de parámetros se hace con el fin de encontrar los parámetros del clasificador que mejor se ajusten a los datos, de modo que le permitan al clasificador, crear una mejor generalización y hacer mejores predicciones.

Para hacer un barrido de parámetros, se utiliza la funcion *GridsearchCV* de la libreia scickit learn. Esto se hace pasándole a la función grid, el clasificador sobre el cual se va a hacer la búsqueda, y sus parámetros en foma de diccionario.

Primero, se invoca la librería:

from sklearn.grid_searsh import GridSearchCV

Luego, se define el clasificador.

svr = SVC()

Se definen los parametros y los valores sobre los que se va buscar en forma de diccionario. En las SVM los parámetros que se suelen buscar son el “kernel”, la toleracia y el “valor gamma”:

params = [{‘kernel’: [‘rbf’,’linear’], ‘gamma’: [0, .1, .001, .0001], ‘tol’: [.1,.01,.001]}

Ahora se pasa el clasificador y la variable “params” como parámetros al gridsearch. La función GridsearchCV tiene integrada la funcionalidad del cross-validation de modo que es un parámetro que se puede definir también, solo hay que especificar el valor de k-fold.

clf = GridSearchCV(svr, params, cv=5)

Se ejecuta el grid search sobre los datos de entrenamiento

Clf.fit(x_train, y_train)

Finalmente se obtienen los parámetros del clasificador que mejor encajaron sobre los datos.

*print "best parameters:",clf.best_params_*

2.3.7 Medida de desempeño del clasificador

Para la medir el desempeño del clasificador se usó el cross validation, los valores TPR, TNR, FPR, FNR, accuracy y los valores de precision and recall. Para obtener un valor de desempeño con el cross validation, basta con recordar que el cross validation divide el dataset en k partes, entrena con las k-1 partes y predice con la

que se deja a parte, esto se repite k veces, por lo que al final se tiene k valores. Para medir la el valor de la predicción, se calcula la media y la desviación estándar con los k valores obtenidos. En la figura 25 se puede observar el valor de las predicciones con todas las pruebas.

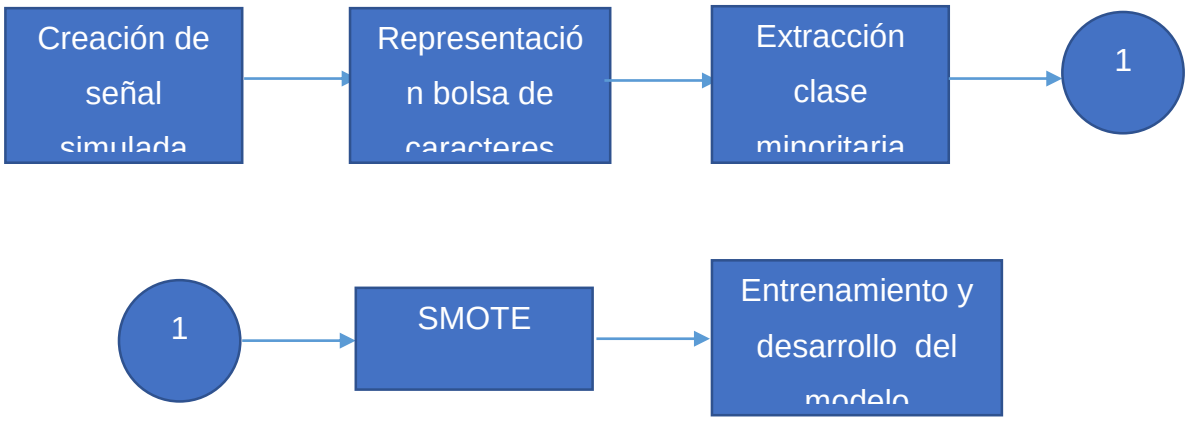
Figura 28. Desempeño del clasificador

```
0's: 1652    1's: 1611
----------------
training with crossvalidation
[ 0.97865854  0.96636086  0.96932515  0.9601227   0.95705521  0.94785276
  0.96319018  0.95092025  0.96932515  0.97239264]
mean 0.963520344193
std 0.00917076415071
----------------
0's in test: 332    1's in test: 321
TRUE POSITIVE RATE (TPR):  97.4683544304
FALSE POSITIVE RATE (FPR):  2.37388724036
TRUE NEGATIVE RATE (TNR):  96.1424332344
FALSE NEGATIVE RATE (FNR):  4.11392405063
ACCURACY:  96.7840735069

             precision    recall  f1-score   support

    class 0       0.96      0.98      0.97       332
    class 1       0.97      0.96      0.97       321

avg / total       0.97      0.97      0.97       653
```

Fuente: Autor

2.3.8 Diseño del Framework

Figura 29. Framework



Este framework comineza con la ejecución del script “create_signal.py” ya que este crea una señal simulada, junto con su archivo ground truth. Esto se hizo debido a que solo se contaba con las señales de los detectores pero no con su archivo ground truth.

Una vez obtenida la señal y su ground truth, se pasan como parametros al script “create_bof_representation.py”, el cual procede a procesar la señal y el ground truth de acuerdo a los parametros estipulados para crear un archivo que contiene el dataset con la señal discretizada.

Si al analizar el archivo con la discretización se desea hacer un over-sampling se procede entonces a separar las clases (clase mayoritaria y minoritaria) presentes en el dataset en archivos diferentes. Para hacer esto, se ejecuta el script “extract_minority_class.py”, el cual se le pasa como uno de sus parámetros el archivo con la discretización y este se encarga de dividir las clases y guardarlas en archivos diferentes.

Ya al tener las clases separadas en archivos diferentes, se procede a ejecutar el script “smote.py”. Este script se encarga de hacer el over-sampling y reconstruir el dataset, con los datos creados por el método usado en el smote (knn).

Una vez se obtenga el archivo con el dataset reconstruido del smote, se procede a entrenar un clasificador, se hace el barrido de parametros con el grid search y se crea el modelo.

Scripts y parametros:

2.3.8.1 Creación de señal simulada

Create_signal.py: Este script, crea una señal simulada de acuerdo con los parámetros especificados y produce 2 archivos. Uno con la señal simulada (signal.dat) y otro con su ground truth (gt_data.csv). Tiene 8 parametros los caules son:

- min_width_signal: ancho mínimo (en número de datos) de los eventos forbush simulados.
- max_width_signal: ancho máximo (en número de datos) de los eventos forbush simulados.
- min_height_signal: alto mínimo (en número de datos) de los eventos forbush simulados.
- max_height_signal: alto máximo (en número de datos) de los eventos forbush simulados.
- min_width_separator: ancho mínimo (en número de datos) del separador entre eventos.
- max_width_separator: ancho máximo (en número de datos) del separador entre eventos.
- noise: Nivel de ruido en la señal.
- nb_cycles: número de ciclos o eventos en la señal.

2.3.8.2 Recreación bolsa de caracteres

create_bof_representation.py: Este script toma como entrada, los archivos signal.dat y gt_data.csv para procesarlos y crear el archivo con el dataset con la discretización. (discretization.dat). Tiene 9 parametros los cuales son:

- input_file: Archivo de entrada que contiene la señal.
- gt_file: Archivo de entrada que contiene el ground truth
- output_file: Archivo de salida que contendrá la discretización.
- sw_size: tamaño de subventanas (en número de datos)
- sw_offset: offset de las subventanas (en número de datos)
- w_size: tamaño de las ventanas (en número de ventanas)
- w_offset: offset de las ventanas (en número de ventanas)
- nb_quadrants: Numero de cuadrantes sobre los que se quiere discretizar.
- min_fit_pct: porcentaje mínimo que una ventana debe tener dentro de un FD para ser etiquetado como contenedor de un evento.

2.3.8.3 Extracción clase minoritaria

extract_minority_class.py: Este script toma como entrada, el archivo con la discretización para extraer y guardar en archivos separados la clase mayoritaria y la clase minoritaria. Tiene 3 parámetros:

- input_file: Archivo de entrada que contiene la discretización.
- Output_file_0: Archivo de salida que contendrá el dataset con etiqueta 0
- Output_file_1: Archivo de salida que contendrá el dataset con etiqueta 1

2.3.8.4 Smote

Smote.py: Este script toma como entrada, el archivo con la clase mayoritaria, el archivo con la clase minoritaria y le aplica un oversampling. El archivo de salida contiene el dataset reconstruido con la clase mayoritaria y la clase minoritaria junto con su oversampling. Tiene 5 parámetros:

- Input_file_mayor: Archivo de entrada que contiene la clase mayoritaria
- Input_file_minor: Archivo de entrada que contiene la clase minoritaria
- N: porcentaje (en múltiplos de 100) de nuevos datos de la clase minoritaria

- k: número de vecinos cercanos para el método knn

- output_file: Archivo de salida que contendrá el dataset con las clases ya balanceadas.

2.3.8.5 Entrenamiento y desarrollo del modelo

Entrenamiento y desarrollo del modelo: En este módulo, se comienza dividiendo el dataset en 2 partes, entrenamiento y test (por lo general 80% y 20% respectivamente). Se define el clasificador sin parámetros de modo que cuando se ejecute el grid search, se obtengan los mejores parámetros que encajen sobre los datos.

Se crea el diccionario con los parámetros a los que se les va a hacer un barrido, se vuelve a definir el clasificador pero con los parámetros encontrandos con el grid search. Se valida el modelo con el cross-validation calculando la media y la desviación estándar de sus valores. Se entrena el modelo con el dataset de entrenamiento y luego se pone a prueba con el de test. Se calcula el desempeño del modelo con las métricas estipuladas TPR, TNR, FPR, FNR, precision and recall, y las métricas desarrolladas para medir y validar la señal que el modelo decide guardar.

2.3.9 Revision del diseño y funciones nuevas

Aunque se tiene métricas para medir el desempeño del modelo, se aplicaron 3 nuevas métricas para medir el desempeño del algoritmo basado en la señal que el algoritmo decide guardar.

Estas nuevas métricas son:

- **Porcentaje de señal que contiene eventos:** Este porcentaje se calcula para tener una base confiable con la cual comparar el porcentaje de señal relevante que el algoritmo decide guardar.

- **Porcentaje de señal que el algoritmo decide guardar:** Este porcentaje muestra la cantidad de señal que el algoritmo decide guardar por que predijo que tenían eventos forbush (FD)

- **Porcentaje de señal que tiene eventos y el algoritmo decide guardar:** Este porcentaje representa la porción de señal que es relevante, es decir, es la porción de señal que el algoritmo decidió almacenar y que realmente contienen FD.

Estas métricas se decidieron implementar por que muestran el desempeño del algoritmo con respecto a la señal que se almacena y que al final, es la métrica más importante debido a la limitación del grupo Halley para almacenar grandes cantidades de datos, lo que hace que sea de gran importancia optimizar el espacio de almacenamiento, guardando solo las porciones de señales que contienen evento forbush.

Para tener cierto control sobre los datos guardados, se implementa un valor de sensibilidad el cual, permite controlar la cantidad de datos que se guardan.

Acá se implementan las nuevas métricas en el último entregable y al ya haberse provisto el archivo con los datos que definen los FD de la señal mcmu, se procede a hacer pruebas.

El framework queda implementado en un ipython notebook.

2.4 PRUEBA Y ANÁLISIS DE RESULTADOS

En este momento ya se contó con el archivo que contenía los datos de los FD en la señal de McMurdo, por lo tanto, se procedió a crear su ground truth y a aplicar el framework sobre la señal. Los resultados se pueden observar en la figura 26.

Figura 30. Resultados con señal mcmu

```
0's in train: 22775    1's in train: 4299
----------------
----------------
0's in test: 5845    1's in test: 923
TP 647    - %: 70.0975081257
FP 2361    - %: 40.3934987169
TN 3484    - %: 59.6065012831
FN 276    - %: 29.9024918743

             precision    recall  f1-score   support

    class 0       0.93      0.60      0.73      5845
    class 1       0.22      0.70      0.33       923

avg / total       0.83      0.61      0.67      6768


events in test: 36
predicted events 36
no predicted events 0
         event id        windows per event
----------------
----------------
percentage of signal containing events 0.164478213829
percentage of signal that algorithm decides to save 0.753172027147
percentage of signal that contains events and the algorithm decides to save 0.883839138885
```

Estos resultados son preliminares ya que el objetivo es usarlos como referencia en la calibración de los parámetros de los scripts tales como el tamaño de las ventanas y subventanas en el script “representación bolsa de caracteres”, o los parámetros del clasificador tales como el valor “C” de tolerancia para detener el criterio.

Al observarse que modificando los parámetros presentes en este framework no modifican los resultados obtenidos, se decidió hacer un análisis del archivo ground truth.

Para esto, se graficaron los eventos especificados como decrecimientos forbush para permitir una comparación visual de estos. Ver ANEXO B.

Esto nos permitió detectar que el propio criterio que el grupo Halley usaba para delimitar eventos forbush necesitaba ajustes. Se presentaron estas gráficas al grupo, quienes confirmaron el error en el delimitado y se comenzó con el proceso de corrección el cual, al finalizar este proyecto no había terminado.

3 CONCLUSIONES

- Se descubrió la necesidad de ajustes en el criterio de los físicos para delimitar eventos forbush.

- Se desarrolló un algoritmo que permite procesar señales y representar sus características adecuadamente en datasets que permiten entrenar y validar algoritmos de clasificación.

- Se construyó un framework que integró los procesos necesarios para crear datasets, entrenar clasificadores y validar modelos adecuadamente.

4 RECOMENDACIONES

- Se recomienda hacer un filtrado y corrección de los datos suministrados como ground truth ya que estos no permiten hacer una validación correcta del desempeño de los clasificadores.

- Se recomienda probar el framework desarrollado en hardware con características al menos parecidas a las de los dispositivos sobre los que se ejecutaría el framework de modo que se pueda hacer un análisis de los requerimientos necesarios para su ejecución.

- Ya que este framework usa una librería robusta como lo es Numpy, se recomienda hacer un estudio con análisis numéricos para reemplazar los métodos matemáticos usados con esta librería y así, ejecutar sobre otras plataformas con menor capacidad de procesamieto, este framework, sin ocupar muchos recursos del sistema.

REFERENCIAS BIBLIOGRAFICAS

[1] PHILLIPS, Tony. "Afraid of a Solar Flare?" {En línea}. {Abril 2015}. Disponible en: http://science.nasa.gov/science-news/science-at-nasa/2005/07oct_afraid/

[2] NMDB NEST, Base de datos de monitores de neutrones. {En línea}. {Abril 2015}. Disponible en http://previ.obspm.fr/hidden3/search.php

[3] SEGARAN, Toby. Programming: Collective Intelligence. Sebastopol: O'Reilly, 2007. 334p.

[4] MARSLAND, Stephen. Machine Learning: An Algorithm Perspective. Florida: Chapman & Hall/CRC. 2009. 390p

[5] CHAMPANDARD, Alex. Reinforcement Learning. {En línea}. {Abril 2015}. Disponible en http://reinforcementlearning.ai-depot.com/Intro.html

[6] MURPHY, Kevin. Machine Learning: A probabilistic Perspective, Massachusetts: The MIT Press. 2012. 1067p

[7] HASTIE, Trevor. TIBSHIRAN, Robert. FRIEDMAN, Jerome. The Elements of Statistical Learning: Data mining, Inference, and Prediction. Segunda Edición. New York: Springer. 2009. 745p.

[8] BOSWELL, Dustin. Introduction to Support Vector Machines. {En línea}. {Abril 2015}. Disponible en http://dustwell.com/PastWork/IntroToSVM.pdf

[9] GUGGENBERGER, Andre. Another Introduction to Support Vector Machines. {En línea}. {Abril 2015}. Disponible en

http://mindthegap.googlecode.com/files/AnotherIntroductionSVM.pdf

[10] VAPNIK, Vladimir. CORTES, Corinna. Support-Vector Networks. {En línea}. {Abril 2015}. Disponible en

http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf

[11] [Classification: Basic Concepts, Decision Trees, and Model Evaluation]. {En línea}. {Abril 2015}. Disponible en

http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

[12] TIRADOS, Maribel. Algoritmo Random Forest. {En línea}. {Abril 2015}. Disponible en http://www.bigdatahispano.org/noticias/algoritmo-random-forest/

[13] NICOLAS, Patrick. Scala for Machine Learning. Birmingham: Packt Publishing Ltd. 2014. 491p.

[14] ALPAYDIN, Ethem. Introduction to Machine Learning. Tercera Edición. Massachusetts: Thomas Dietterich. 2014. 537p

[15] Scikit-Learn. Machine Learning in Python. [Online] http://scikit-learn.org/stable/

[16] LAGO. The Large Aperture GRB Observatory. [Online]

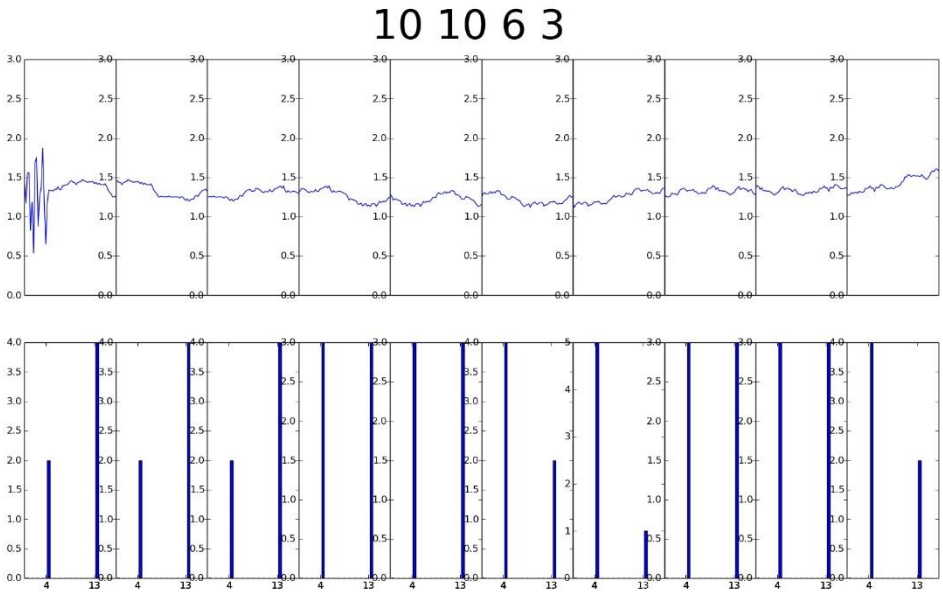
http://labdpr.cab.cnea.gov.ar/lago/

[17] Unix Time Stamp. [Online] http://www.unixtimestamp.com/

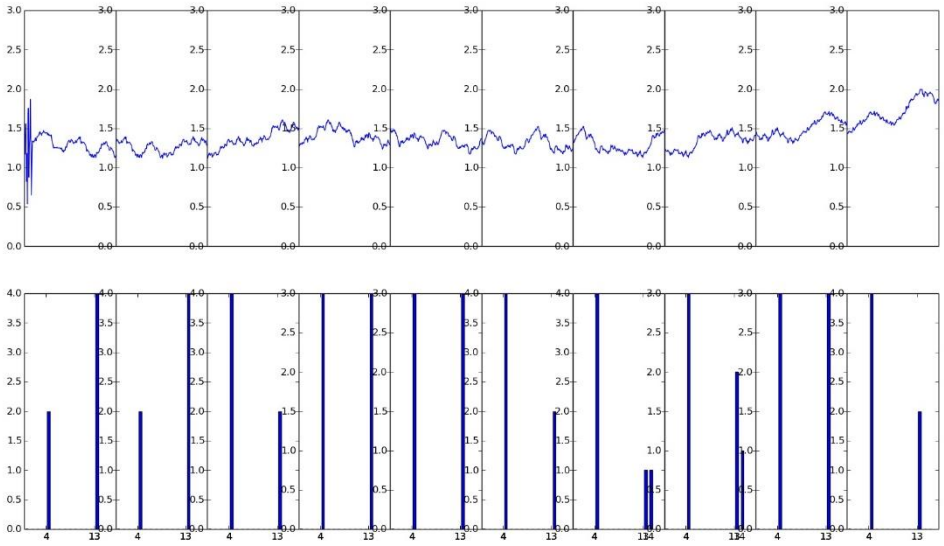
ANEXOS

ANEXO A. Señales de detectores y su discretización

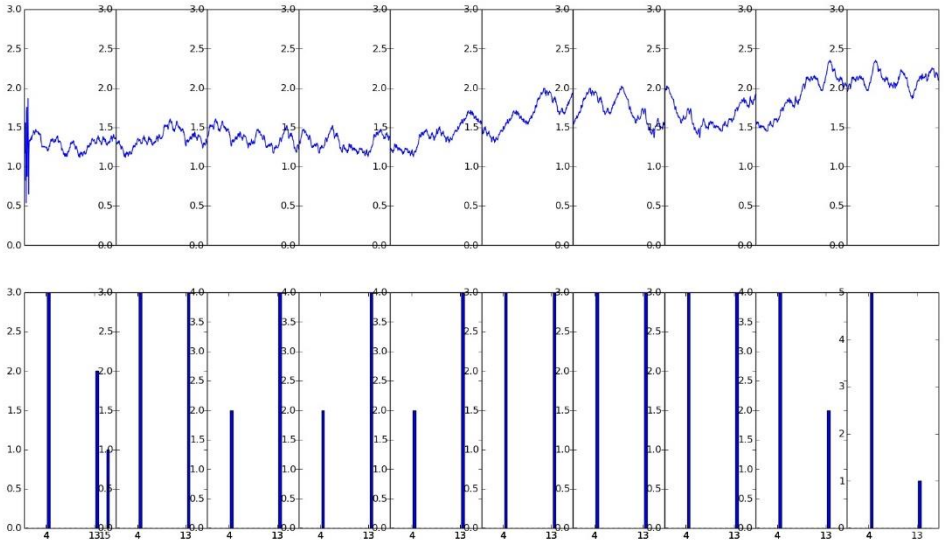
Distribución de las pendientes en la discretización de la señal. Los dos pares de números encima de cada figura son el tamaño de la subventana, el offset de la subventana, el tamaña de la ventana y el offset de la ventana.

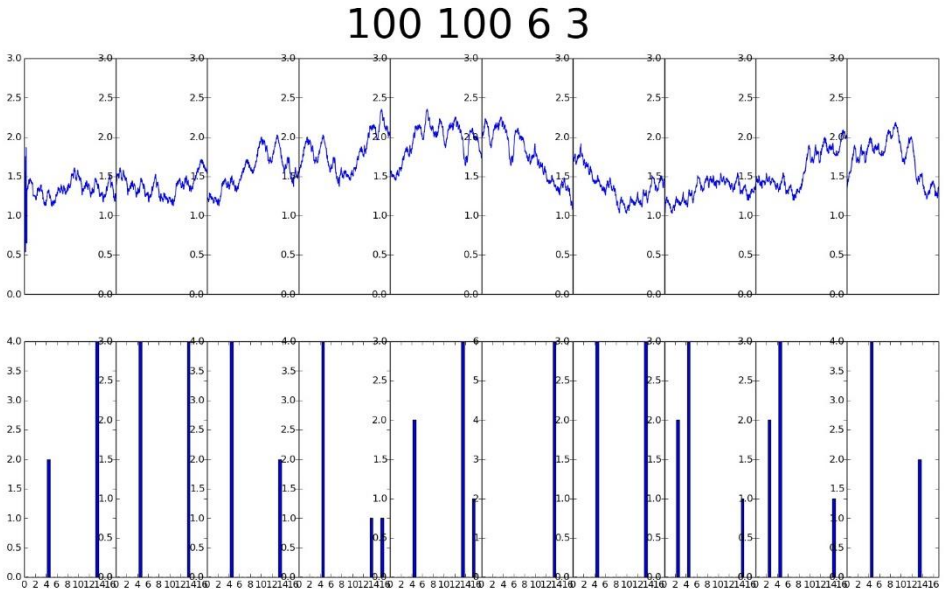


30 30 6 3



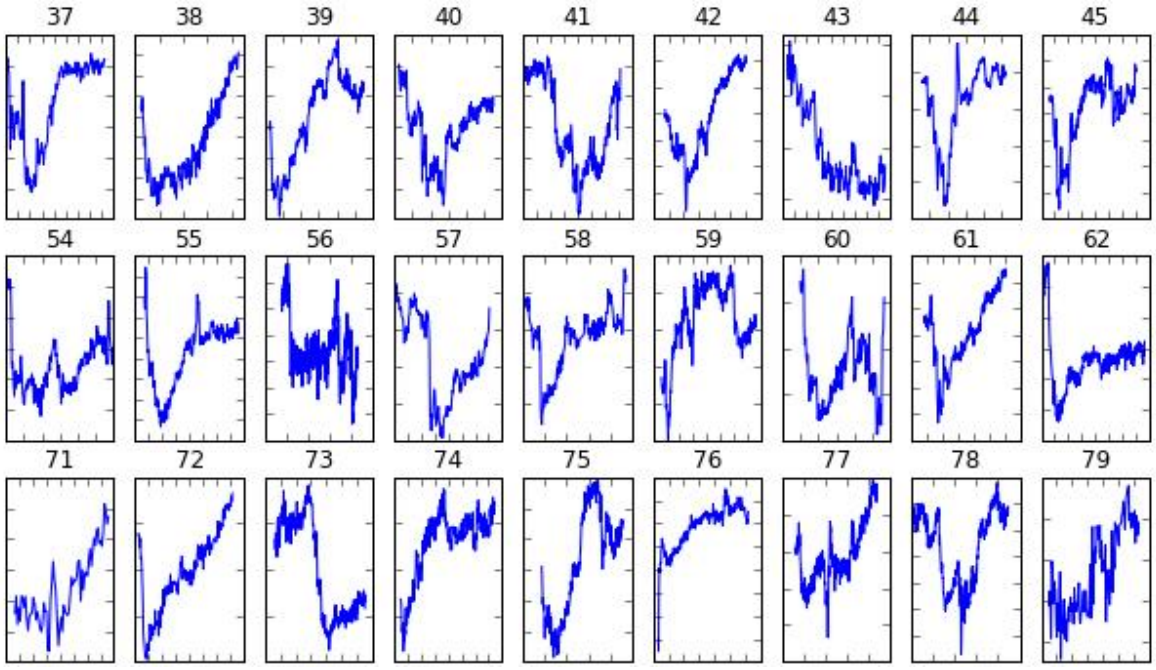
50 50 6 3

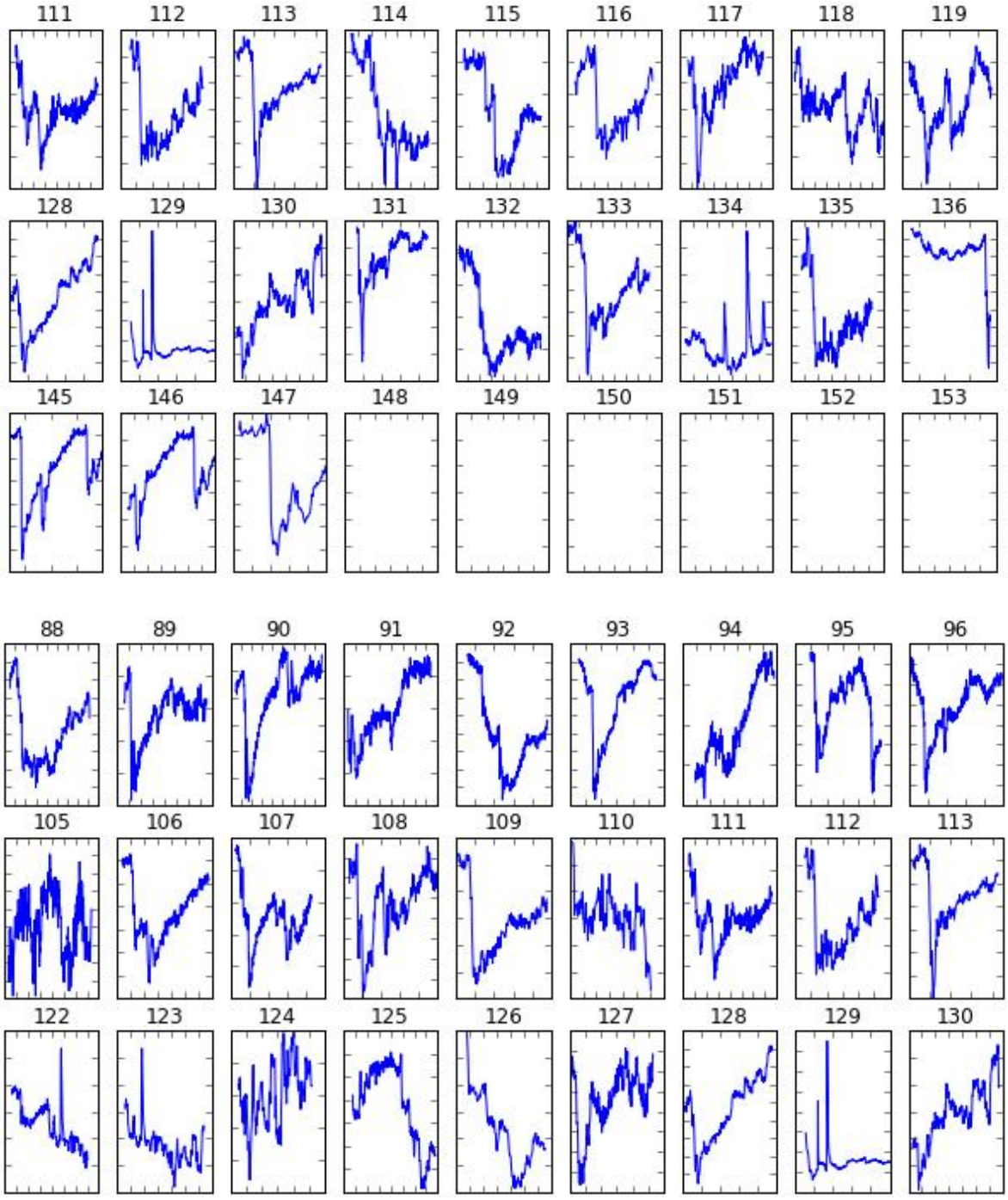


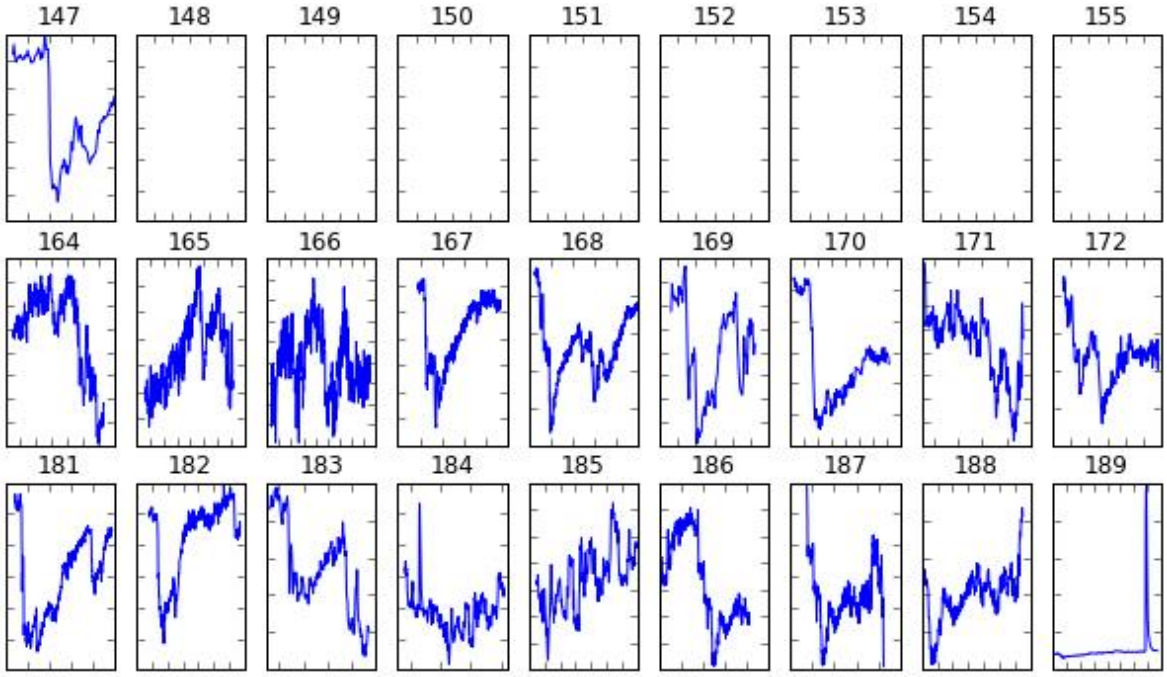

100 100 6 3

ANEXO B. Eventos Forbush anotados por los especialistas

Estos son los eventos delimitados por los especialistas y con los cuales se crea el archivo ground truth. En las gráficas se pueden apreciar porciones de señal correctamente definidas como eventos forbush, así como porciones de señal definidas como eventos forbush y que no lo son. Algunas gráficas están en blanco debido a que cuando se corrige la señal por presión y eficiencia, se pierde parte de ella.




111
112
113
114
115
116
117
118
119
128
129
130
131
132
133
134
135
136
145
146
147
148
149
150
151
152
153
88
89
90
91
92
93
94
95
96
105
106
107
108
109
110
111
112
113
122
123
124
125
126
127
128
129
130


147
148
149
150
151
152
153
154
155
164
165
166
167
168
169
170
171
172
181
182
183
184
185
186
187
188
189