Proyecto Final Machine Learning

Titanic disaster

**Integrante:** Sebastián Salazar Pulgar

**Profesor:** Jorge Vergara

# **Introducción**

En este proyecto se busca implementar y comparar dos reconocidos algoritmos aplicados en el Machine Learning, para esto ocuparemos como set de datos una base de datos con los sobrevivientes al desastre del crucero Titanic.

Desarrollar un modelo de aprendizaje. con este set de datos es considerado un “Hola Mundo” al campo del Machine Learning por lo que para obtener un poco de complejidad y para que este proyecto sea mas interesante compararemos el rendimiento y distintos indicadores de los siguientes algoritmos de aprendizaje: Random Forest y Gradient Boosted. Además, como el set de datos a trabajar es un set pequeño ocuparemos el algoritmo de Cross Validation para evaluar rendimiento entre los modelos antes de entrarlos.

# **Marco Teórico**

A continuación, detallaremos los conceptos, algoritmos y métodos usados para el desarrollo de este proyecto:

## **Clasificación vs Regresión**

La clasificación se refiere a la tarea de proporcionar características de algoritmo de aprendizaje automático y hacer que el algoritmo coloque las instancias / puntos de datos en una de muchas clases discretas. Las clases son representaciones naturaleza categórica, no es posible clasificar una instancia como parcialmente una clase y parcialmente otra.

Las regresiones se realizan cuando la salida del modelo de aprendizaje automático es un valor real o un valor continuo. Tal ejemplo de estos valores continuos podría ser, distancias, pesos, cantidad de un componente en una sustancia… etc.

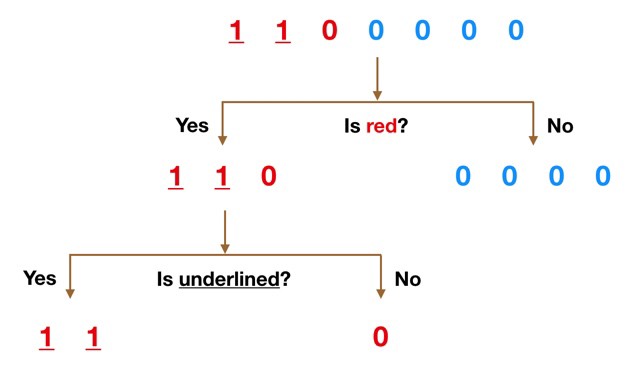
Para efectos de que el set de datos ocupados en esta instancia es de carácter discreto, es decir, categórico ocuparemos métodos de clasificación para resolver el problema

## **Random Forest**

El algoritmo Random Forest o en español Bosques Aleatorios tiene su versión como algoritmo como clasificador o como algoritmo de regresión, en este proyecto nos enfocaremos en su versión de clasificador, ya que es lo que necesitamos para resolver el problema del titánic.

Para poder entender el algoritmo de Random Forest primero debemos tener bien claro que es un árbol de decisión. En términos prácticos un árbol de decisión es un esquema donde una condición tiene dos respuestas: si y no, dependiendo de la respuesta pasaríamos a evaluar otra condición, así sucesivamente hasta llegar a un resultado.

Así se vería un árbol de decisiones:



Random Forest consiste en una gran cantidad de arboles de decisión aleatorios que operan como un conjunto pero que no tienen ninguna correlación entre ellos. Cada árbol da como resultado una predicción de una clase, la clase que se repite mas veces es la clase que se convierte en la predicción real de nuestro modelo.



El concepto fundamental detrás de este algoritmo es “La sabiduría de las Multitudes”, dentro del campo de las ciencias de la computación este concepto es poderoso, en el caso de Random Forest se puede decir que funciona tan bien por la gran cantidad de arboles que este maneja y la baja correlación que estos tienen.

¿Como asegura Random Forest que los arboles sean individuales y no tengas correlaciones entre si? Ocupa dos métodos:

### **Bagging:**

Los árboles de decisión son muy sensibles a los datos en los que están capacitados; pequeños cambios en el conjunto de entrenamiento pueden dar como resultado estructuras de árbol significativamente diferentes. Random Forest aprovecha esto al permitir que cada árbol individual muestree al azar del conjunto de datos con reemplazo, lo que da como resultado diferentes árboles.

### **Feature Randomness:**

En un árbol de decisión normal, cuando es el momento de dividir un nodo, consideramos todas las características posibles y elegimos la que produce la mayor separación entre las observaciones en el nodo izquierdo y las del nodo derecho. Por el contrario, cada árbol en un Random Forest puede elegir solo de un subconjunto aleatorio de características. Esto obliga a una variación aún mayor entre los árboles del modelo y, en última instancia, da como resultado una menor correlación entre los árboles y una mayor diversificación.

## **Gradient Boosting Machine**

Gradient Boosting Machine es un conjunto de algoritmos usados en el area del Machine Learning que combinan modelos de aprendizajes debiles para crear un modelo de aprendizaje solido.

La idea detrás de Gradient Boosting es tomar un algoritmo de aprendizaje débil y hacer una serie de ajustes que mejoren la fortaleza de este, este metodo se basa en la idea de Probability Approximately Correct Learning (PAC).

Esta idea se realizo en el algoritmo Adaptrive Boosting (AdaBoost), para AdaBoost, muchos modelos débiles se crean al inicializar muchos algoritmos de árbol de decisión que solo tienen una sola división.

Imagen que contiene pared

Descripción generada automáticamente

Las observaciones en el conjunto de entrenamiento son ponderadas por el algoritmo, y se asigna más peso a las instancias que son difíciles de clasificar.

En AdaBoost, las predicciones se realizan por mayoría de votos, y las instancias se clasifican según la clase que recibe la mayoría de los votos de los alumnos débiles.

El algoritmo de Gradient Boosting Classifier es la combinacion de el metodo de AdaBoost y el metodo del gradiente, esto permite minimizar la pérdida o la diferencia entre el valor de clase real del ejemplo de entrenamiento y el valor de las clase de prediccion.

## **Cross Validation**

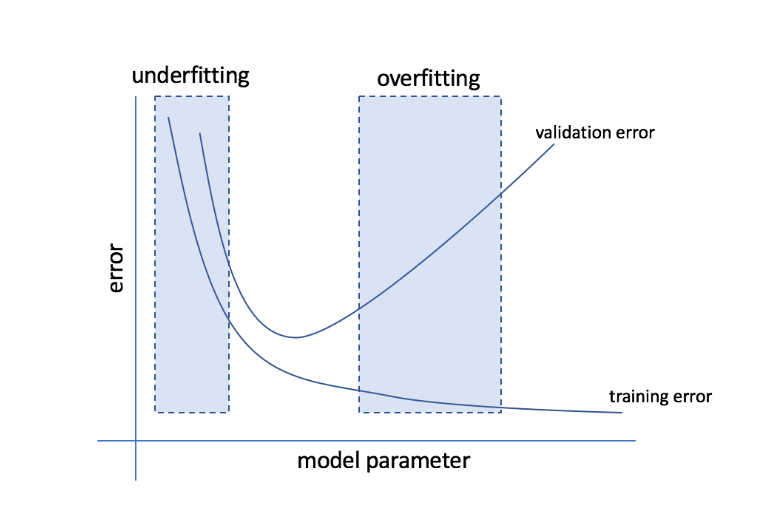
Cross Validation o Validacion Cruzada es uno de los conjuntos metodos mas importantes para poder saber si el modelo ocupado se ajusta a el conjunto de datos que ocuparemos, de esta manera podremos saber la precision respecto al set de datos y si el modelo no esta captando ruido de este mismo.

Cross Validation nos permite saber si un modelo se generalizara a un conjunto de datos, este se usa principalmente en casos donde el objetivo es la prediccion de algo y uno necesita estimar con que precision se desempeñara el modelo predictivo. Con esto podremos limitar problemas como el Overfitting (sobreajuste) y Underfitting (Desajuste) en la generealizacion del modelo.

### **¿Qué es overfitting y underfitting?**

La falta de ajuste se refiere a no capturar suficientes patrones en los datos. El modelo funciona mal tanto en el entrenamiento como en el conjunto de prueba.

El sobreajuste se refiere: a) captura de ruido yb) captura de patrones que no se generalizan bien a datos invisibles. El modelo se desempeña bien con el conjunto de entrenamiento pero mal en el conjunto de prueba.



En Cross Validation existen diferentes estrategias de validación basadas en el número de divisiones que se realizan en el conjunto de datos:

### **Dividir el conjunto de datos en Test y Train**

En esta estrategia, dividimos los datos en dos conjuntos: train y test para que la muestra entre train y test no se superpongan, si lo hacen, simplemente no podemos confiar en nuestro modelo. Esa es la razón por la cual es importante no tener muestras duplicadas en nuestro conjunto de datos. Antes de hacer nuestro modelo final, podemos volver a entrenar el modelo en todo el conjunto de datos sin cambiar ninguno de los hiperparámetros del modelo.

Pero esta estrategia tiene una gran desventaja, ya que si el conjunto de datos fuera muy pequeño podriamos tener datos generalizados y esto resultaria en un Overfitting de parte del modelo.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

### **Dividir el conjunto de datos en K grupos (K-Fold)**

Cross Validation K-Fold es un metodo que nos separa el conjunto de datos de entrenamiento en k conjuntos de validacion, con esto podremos medir el ajuste de los datos por separado para luego sacar un promedio de la precision de la prediccion para cada uno de estos conjuntos (modelos).

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Este método es una buena opción cuando tenemos una cantidad mínima de datos y obtenemos una diferencia lo suficientemente grande en calidad o diferentes parámetros óptimos entre pliegues. Como regla general, se eligen k = 5 o k = 10, ya que se ha demostrado empíricamente que estos valores producen estimaciones de error de prueba que no sufren ni un sesgo excesivamente alto ni una varianza alta.

Como el set de datos ocupado es pequeño, ocuparemos este metodo de Cross Validation para validar nuestros modelos y identificar cual se comporta mejor con este.

### **Dividir el conjunto de datos en K grupos pero K es igual al numero de datos de entrenamiento**

Es un caso especial de Kfold cuando K es igual al número de muestras en nuestro conjunto de datos. Esto significa que iterará a través de cada muestra en nuestro conjunto de datos cada vez que use el objeto k-1 como muestras de tren y 1 objeto como conjunto de prueba.

Imagen que contiene captura de pantalla

Descripción generada automáticamente

Este método puede ser útil si tenemos muy pocos datos y un modelo lo suficientemente rápido como para volver a entrenar.

### **Estratificación**

Tambien es un caso especial de K-fold en donde se logra una distribucion objetico similar a la de las distintas divisiones del conjunto de datos de entrenamiento, esto es util ya que en conjuntos dde entrenamientos muy pequeños es posible que la distribucion objetivo sea distinta a la de las distintas partes.

Esta estrategia es útil para:

* Pequeños conjuntos de datos
* Conjuntos de datos no balanceados
* Clasificación multiclase

En general, para un gran conjunto de datos equilibrado, la división de estratificación será bastante similar a una división aleatoria simple (K-fold) .