Primer Entrega
Taller de Aprendizaje Automático

Instituto de Ingeniería Eléctrica

Federico Bello

28 de septiembre de 2025

Taller 1 - Titanic

Analizando la salida del metodo *info* con los datos de train, se observa que los atributos se dividen en 3 tipos. Los atributos *Surivived, Pclass, SibSp y Parch* son de tipo *integer*, luego, *Name, Sex, Ticket, Cabin y Embarked* son de tipo *object* y por ultimo, *Age, y Fare* son de tipo *float*.

Descripción de los atributos:

- **Survived:** 1 si el pasajero sobrevivió, 0 si no. (atributo a predecir).
- **Pclass** indicador del nivel socio-económico (1 alto, 2 medio, 3 bajo).
- **Sex:** masculino o femenino
- **Age:** edad en años.
- **Sibsp:** cantidad de hermanos/as y esposos/as abordo.
- **Parch:** cantidad de hijos/as y padres/madres abordo.
- **Ticket:** numero de ticket.
- **Fare:** precio del pasaje.
- **Cabin:** numero de cabina.
- **Embarked:** puerta de embarcación (C: Cherbourg, Q: Queenstown o S: Southampton)

En un primer razonamiento, los atributos que mas parecerían importar son: *Sex, Cabin y Age*, ya que parecería ser de gran importancia la ubicación del barco, el estado físico de la persona (o si es un niño) y el género. Sin embargo, de 891 datos, simplemente se tiene el número de cabina de 204, imposibilitando sacarle utilidad a este atributo. Además, se ve como algunos atributos no parecen aportar nada al problema, ejemplos de esto son el nombre del pasajero o el numero de ticket.

Utilizando el método *corr* es posible ver la correlación entre todos los atributos numéricos del problema, por lo que, filtrando por nuestra variable a predecir, es posible analizar la correlación que cada atributo numérico tiene con dicha variable. Haciendo uso de esto, se obtienen los valores de la figura 1 y la tabla 1. Se observa como los atributos *Fare y PClass* tienen una alta correlación (0,257 y $-0{,}338$ respectivamente). En el caso particular de *Pclass* tiene sentido analizar la correlación ya que, si bien es un atributo categórico, cuanto menor el valor, mayor el poder económico. Es por esto que una correlación negativa implica que a un mayor poder adquisitivo, una mayor probabilidad de supervivencia. Este caso puede verse también siguiendo una estrategia similar a la que se explicara para el caso del atributo *Sex*. Por otro lado, la edad (en contra

de la intuición inicial) y la cantidad de parientes abordo no parece influir de forma considerable (correlación cercana a 0). También se intento combinar estos últimos dos atributos en uno solo, que fuera la cantidad de parientes abordo. Esto, no mejoro la correlación entre los atributos ni el desempeño del modelo.

Para el atributo *Sex* (análogo para *Pclass*) es posible ver que porcentaje de hombres y que porcentaje de mujeres sobrevivieron. Realizando esto, se llega a que 74,20 % de las mujeres sobrevivieron, y que solo un 18,89 % de los hombres lo hicieron, llegando así a ver que hay una fuerte correlación (en el conjunto de entrenamiento) entre ser mujer y sobrevivir y ser hombre y no hacerlo.

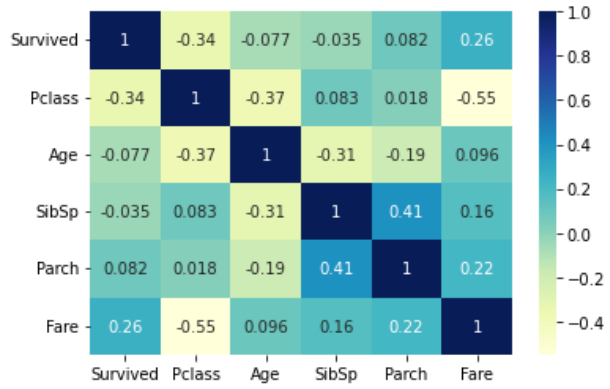

Figura 1: Matriz de correlación

Atributo	Correlación
Pclass	-0.338481
Age	-0.077221
SibSp	-0.035322
Parch	0.081629
Fare	0.257307
Survived	1.0000

Cuadro 1: Correlación de los atributos con *Survived*

Para solucionar en problema en cuestión, se generaron tres *pipelines* distintos. El primero, solamente se queda con la columna *Sex*, utilizando una codificación ordinal. Se genera el modelo utilizando regresión logística con una *Grid Search* para encontrar el parámetro de regularización (*C)* óptimo. El segundo, descarta los atributos *Ticket, Cabin y Name*, rellena los datos faltantes en los atributos numéricos con la mediana y para los atributos categóricos utiliza la codificación ordinal. El modelo se genera igual que en el caso anterior. Por ultimo, el tercer modelo, basado en el anterior, le agrega al mismo un escalado de los datos. Se utiliza una estandarización para *Age y Fare* y un escalado entre 0 y 1 para *SibSp y Parch*. La razón de esto último es que es un tipo ordinal de pocos valores, por lo que haciendo esto se asegura que no se pierda la distancia relativa entre los valores. Además, a diferencia de los anteriores, el modelo se genera utilizando maquinas de vectores de soportes, repitiendo la búsqueda de los parámetros con la grilla.

Solo género	**Básico**	**Avanzado**
0.8034	0.7865	0.8202
0.8034	0.7865	0.8315
0.7528	0.7809	0.81461
0.7865	0.7697	0.7921
0.7853	0.8192	0.8644

Cuadro 2: Valores de *accuracy* para los distintos pipelines, evaluado con validación cruzada de 5 *folds*

Para evaluar los distintos pipelines, se realiza una validación cruzada de 5 folds, con *accuracy* como medida de desempeño, obteniendo así los valores de la tabla , donde solo género es el primer pipeline descripto, básico el segundo y avanzado el ultimo. Además, se probo realizar una modificación al ultimo pipeline, utilizando *Random Forest* en lugar de las maquinas de vectores de soportes, obteniendo un peor resultado. Por lo tanto, de los modelos propuestos, el ultimo es el que da mejor resultado. Evaluando el mismo en el conjunto de test se obtiene un desempeño de 0.78, como puede verse en la figura . Teniendo en cuenta que en el conjunto de entrenamiento el porcentaje de sobrevivientes es 38,4 %, si asumimos que la proporción se mantiene en el

conjunto de test, se obtiene un mejor resultado que utilizando el modelo mas simple posible (decir que todos murieron).

Figura 2: Desempeño en conjunto de test

Taller 2 - Criticas Cine

Para evaluar los pipelines de las partes 4, 5, 6 y 7 se realizo una validación cruzada de 5 *folds*, utilizando la función *cross_val_score*. Como métricas de desempeño se utilizaron *accuracy, precision y recall*. A continuación se recuerda la definición de cada métrica, donde T_P son las predicciones positivas correctas, T_N las negativas correctas, F_P las falsas positivas (se clasificaron como positivas y eran negativas) y F_N las falsas negativas

- **Accuracy:** proporción de predicciones correctas, es decir, $\frac{(T_P + T_N)}{(Total)}$
- **Precision:** proporción de predicciones positivas correctas, es decir, $\frac{(T_P)}{(T_P + F_P)}$
- **Recall:** proporción de predicciones positivas correctamente identificadas, es decir, $\frac{(T_P)}{(T_P + F_N)}$

Métrica \ Modelo	Parte 4	Parte 5	Parte 6	Parte 7
Accuracy	0.8916	0.8888	**0.8954**	0.8867
Precision	0.8855	0.8827	**0.889**	0.8794
Recall	0.8994	0.8967	**0.9047**	0.8962

Cuadro 3: Desempeño con validación cruzada 5 folds

Métrica \ Modelo	Parte 4	Parte 5	Parte 6	Parte 7
Accuracy	0.9500	0.9490	0.9675	1.0
Precision	0.926	0.9241	0.9510	1.0
Recall	0.9537	0.9537	0.9703	1.0

Cuadro 4: Desempeño en conjunto de entrenamiento

En todos los casos se realiza una búsqueda del hiper-parámetro C de regresión logística con el objetivo de maximizar el *accuracy* mediante validación cruzada. Los resultados obtenidos para las distintas métricas se muestran en las tablas 3 y 4. Notar que para todos los pipelines propuestos se logra maximizar las métricas mediante sobre-ajustar el modelo. La razón de haber elegido el *accuracy* como función de costo es que tanto las criticas positivas como las negativas parecen ser de igual importancia, queriendo minimizar el error total y no un tipo de error en especifico. Si el problema en cuestión fuera otro, uno en el que un tipo de error es mas grave que el otro, seria necesario cambiar la función de costo a *precision* o *recall*. Buscar maximizar

la precisión es buena al querer minimizar los errores tipo 1 (falsos positivos) y el recall para minimizar los errores de tipo 2 (falsos negativos).

En este caso en particular, el modelo de la parte 6 devuelve el mejor *score* en todas las métricas evaluadas, indicando que ponderar la relevancia de las palabras e ignorar las *stop words* es un buen pre-procesamiento de la entrada.

Es por esta razón que el pipeline propuesto comienza por **separar las contracciones**, para así evitar que palabras que son iguales no lo sean para el modelo (por ejemplo, *not* es igual a *n't*). Luego, el texto se **lematiza**, esto implica agrupar las palabras en su raíz común, por ejemplo, unificar las palabras running", ran", and runner.[en] solamente run". Esto, puede llegar a tener ciertas ventajas como por ejemplo disminuir la dimensionalidad de los datos, volviendo el algoritmo mas eficiente. Además, disminuye el ruido en los datos, ya que el modelo se vuelve robusto a los distintos tiempos gramaticales o las distintas conjugaciones de las palabras. En el próximo paso del pipeline, **se eliminan las *stop-words* y se utilizan *n-grams*** Las ventajas de eliminar las *stop-words* fueron mencionadas anteriormente, mientras que los n-gramas tienen la ventaja de agregarle contexto al texto, por ejemplo, la palabra çool"tendría sentido que lleve al modelo a pensar que es una reseña positiva, sin embargo, si viene acompañado de "not cool"se podría llegar a pensar que la reseña debería ser negativa. Esto se tiene en cuenta al utilizar los n-gramas (el ejemplo es simplemente para entender la idea, no van a haber n-gramas con la palabra *not* ya que fueron eliminadas por ser *stop-words*). Por ultimo, se ponderan los n-gramas utilizando **Tf-Idf**.

Conjunto / Métrica	Entrenamiento	Validación	Test
Accuracy	1.0	0.8972	0.902
Precision	1.0	0.8848	0.8597
Recall	1.0	0.9133	0.9119

Cuadro 5: Desempeño del pipeline propuesto en los distintos conjuntos de datos

Se observa entonces que, si bien el pipeline propuesto se sobre-ajusta a los datos, obtiene un resultado en validación cruzada superior a cualquiera de los modelos anteriores. Por ultimo, la tercer columna muestra los valores de las distintas métricas en conjunto de test. Notar que, si el problema fuera otro, donde se busque minimizar la cantidad de falsos negativos, seria necesario retocar el modelo, buscando aumentar la *precision* mediante disminuir el *recall*