Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Inteligencia Artificial Avanzada para la Ciencia de Datos I

Ingeniería en Ciencias de Datos y Matemáticas

Selección, configuración y entrenamiento del modelo Reto

Profesores
Daniel Otero Fadul
Hugo Terashima Marin

Equipo 4

Niza Alvarado Rendon	A01367547
Renata Vargas	A01025281
Andrea Galicia Jimenez	A01177643
Ana Daniela López Dávila	A00831568
Alejandro Murcia Alfaro	A00828513
Iván L. Hernández Buda	A01412476

Monterrey, Nuevo León. 01 de septiembre de 2023

1. Introducción

La tecnología y la digitalización de la información ha provocado que, a día de hoy, se manejan cantidades inmensas de datos. Esto ha transformado el proceso por el que tomamos decisiones. Bajo este nuevo contexto, la ciencia de datos y los modelos de aprendizaje automático han ido tomando relevancia en la automatización y mejora de la eficiencia.

Entre los nuevos desafíos que enfrentamos esta la predicción de eventos futuros basada en datos históricos. Es por esto que abordamos .El hundimiento del Titanic", una predicción de la supervivencia de los pasajeros abordo del Titanic. Aplicando técnicas avanzadas de modelado predictivo, buscaremos la razón o los patrones que respondan a la pregunta ¿Quiénes sobrevivieron y quiénes no?

En este reporte, se exploran las decisiones tomadas durante el proceso de selección y preparación de datos. Asimismo, se dan a conocer los modelos de aprendizaje automático utilizados junto con la justificación de los parámetros escogidos para su entrenamiento. Por último, se evalúan los modelos con la finalidad de observar el comportamiento y escoger un modelo definitivo para la continuidad del proyecto.

2. Problemática

Desde el sentido común, se asume que algunos de los factores que influyen en la supervivencia de los pasajeros son: género, clase socioeconómica, edad y familiares abordo. Sin embargo, dentro de la base de datos nos encontramos con más variables que podrían tener un impacto significativo. Por esto es necesario hacer una extensa exploración sobre la relación de nuestras variables para reconocer cómo es que interactúan.

Como parte de la problemática, sera un desafío reconocer si existen o no patrones en los datos. Es crucial que, de identificar patrones, entendamos cómo es que estos afectan dentro del contexto de la situación. Asimismo, se espera identificar a que variables, no tomadas en cuenta, se les atribuye el error de los modelos probados.

Para dar inicio al desarrollo del proyecto, es necesario identificar cuál es el tipo de modelo que será necesario para resolver la problemática. Dado que tratamos de predecir si un pasajero sobrevivió o no al acontecimiento, se entiende que el modelo deberá ser de **clasificación binaria**. Esto se representa de la siguiente forma:

- Sobrevivió: 1
- No sobrevivió: 0

Es así como el modelo, en función de los atributos, realizará una predicción binaria sobre la supervivencia de una persona específica abordo del Titanic. Los modelos de clasificación binaria más utilizados son la

Regresión Logística, Support Vector Machine, Random Forest, entre otros. Sin embargo, la limpieza de los datos será una parte importante al momento de evaluar nuestros modelos. A continuación se describen los datos utilizados para el proyecto.

3. Base de Datos

Para esta asignatura se trabajaron con dos base de datos que contienen información general sobre las personas que se encontraban en el titanic, un csv para entrenar, 'train.csv' y otro para hacer pruebas 'test.csv'. En ellas podemos notar diferentes variables como lo son:

- **PassengerId:** variable entera que representa el ID de cada pasajero
- **Survived:** variable entera, de categoria binaria, la cual representa las personas que sobrevivieron (1) y fallecieron (0).
- **Pclass:** el ticket que representa la clase del pasajero, 1 = 1era clase, 2 = 2nda clase, 3 = 3era clase. Hay que tener en cuenta que las clases hacen referencia al estatus socio-economico del pasajero teniendo que 1era clase = sociedad de alta clase, 2nda clase = sociedad de media clase y 3ra clase = sociedad de baja clase.
- **Name:** nombre del pasajero. Algo notorio en esta columna es que cada nombre tiene un título asociado (como Mr., Mrs., Master., Miss., Bissette y Bishop). Estos títulos pueden proporcionarnos pistas sobre la edad de la persona. Por ejemplo:
  1. 'Master.' generalmente indica niños varones entre 0 a 12 años.
  2. 'Mrs.' se refiere a una mujer casada, lo que sugiere que es poco probable que tenga menos de 16 años (edad de consentimiento).
- **Sex:** sexo del pasajero
- **Age:** edad en años del pasajero.
- **SibSp:** número de hermanas, hermanos, esposos o esposas en el barco
- **Parch:** número de padres de familia y niños en el barco
- **Ticket:** número de ticket
- **Fare:** tarifa del boleto
- **Cabin:** número de cabina del pasajero
- **Embarked:** puerta de embarcación por donde ingresaron los pasajeros. (C = Cherbourg, Q = Queenstown, S = Southampton)

Analizando un poco mas a detalle ambas bases de datos, pudimos notar que el test existen 418 registros mientras que en el train existen 891. El propósito de usar conjuntos de entrenamiento (train) y prueba (test) en una base de datos es entrenar un modelo de aprendizaje automático con datos conocidos (train) y luego evaluar su capacidad para hacer predicciones precisas en datos desconocidos (test). El conjunto de entrenamiento se utiliza para entrenar al modelo, mientras que el conjunto de prueba se utiliza para medir su rendimiento en datos no vistos, lo que ayuda a garantizar que el modelo pueda generalizar correctamente y evitar problemas de sobreajuste. Esta división es esencial para evaluar y ajustar los modelos de manera efectiva.

4. Modelos de Machine Learning

a) Support Vector Machine

Preprocesamiento de Datos:

Se carga primeramente los datos limpios de entrenamiento guardados en un archivo '.csv'. Posteriormente, se eliminan variables categóricas irrelevantes (como 'Ticket', 'PassengerId', 'Name', 'CabinEncoded' y 'Deck') para preparar el conjunto de datos para el modelado. Las variables categóricas sobrantes se convierten en variables dummy utilizando la función pd.getDummies. Se grafica el mapa de calor de correlación entre las variables sobrantes (o relevantes) para identificar relaciones entre características.

Selección de características:

Se calcula la correlación de cada característica con la variable objetivo 'Survived' y se almacena en 'corrSurvived'. Se consideran las características con una correlación menor a 0.2 con 'Survived' para eliminar (featuresToDrop). Estas características de baja correlación se eliminan del conjunto de datos, dando lugar a dataFiltered. Se calcula una nueva matriz de correlación (corMatrixFiltered) basada en las características restantes de alta correlación. Se grafica el mapa de calor de esta matriz de correlación para visualizar las relaciones entre las variables restantes.

Construcción del Modelo SVM:

1. Se importa el modelo SVM de sklearn.svm.

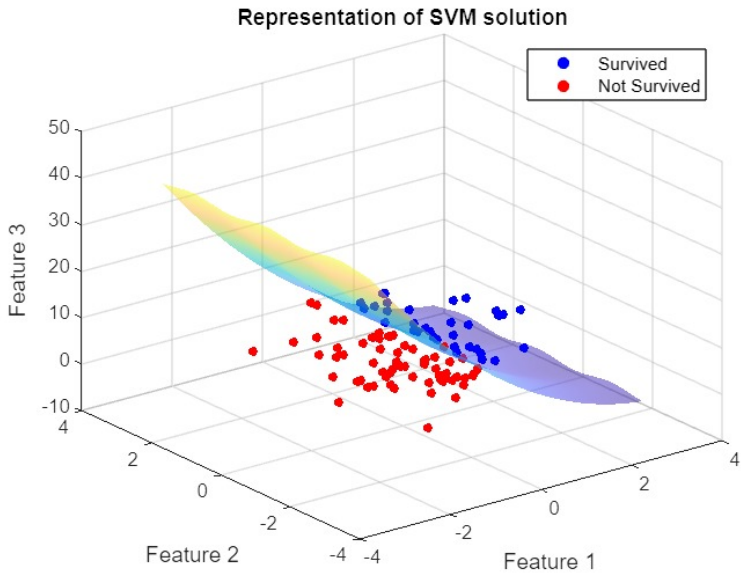
2. Se crea una instancia del modelo SVM con hiperparámetros especificados, como:

   - **C**: Parámetro de penalización del término de error (parámetro de regularización)
   - **kernel**: Especifica el tipo de kernel (por ejemplo, 'poly' para un kernel polinómico)
   - **gamma**: Coeficiente del kernel (auto para 1/n_características)

- **class_weight**: Equilibrar clases ajustando los pesos de manera inversamente proporcional a las frecuencias de clase
- **random_state**: Semilla para el generador de números aleatorios para reproducibilidad

3. El modelo SVM se entrena en los datos de entrenamiento utilizando `.fit()`.

4. Evaluación del Modelo:



Cálculo de precisión

Se hacen predicciones en los datos de prueba utilizando el modelo SVM entrenado. Se calcula la precisión del modelo utilizando 'accuracy_score' de sklearn.metrics. La puntuación de precisión se imprime en la consola.

En resumen, este código carga un conjunto de datos limpios, los preprocesa eliminando variables irrelevantes y creando variables dummy para características categóricas. Luego realiza una selección de características basada en la correlación con la variable objetivo y aplica un modelo SVM con hiperparámetros especificados para clasificación binaria (sobrevivió o no sobrevivió).

La precisión del modelo se evalúa utilizando los datos de prueba. El objetivo es construir un modelo predictivo que clasifique con precisión los resultados de supervivencia basados en las características seleccionadas y el algoritmo SVM.

b) Random Forest

Preprocesamiento de Datos:

Para la limpieza de este modelo, realizamos una serie de modificaciones en los dos sets de entrenamiento y test. Para comenzar, nos enfocamos en la variable de Edad y notar que había muchos valores nulos (177 en el set de entrenamiento y 86 en el de prueba). Esto fue considerando que la edad era la principal variable en un pasajero para determinar su supervivencia. Como eso es un porcentaje sumamente representativo del set de entrenamiento, usaremos Random Forest para llenar estos valores . Para esto, haremos uso de las variables dummy para las variables categóricas y de esta manera puedan ser procesadas por el modelo. Después de eso, quitaremos las columnas que no utilizamos para el análisis: Name, Ticket y Cabin.

Modelo

El modelo Random Forest es el conjunto de múltiples árboles de decisión donde cada árbol está entrenado para un subconjunto de los datos, para que al final se unan para formar una predicción o una clasificación mucho más fundamentada.

La formulación básica del algoritmo Random Forest se puede expresar como sigue:

$$Y = f(X_1, X_2, \ldots, X_k) + \epsilon$$

donde Y es la variable objetivo, $X_1, X_2, \ldots, X_k$ son las características y ϵ es el error aleatorio.

El modelo se entrena utilizando el método `fit` en el conjunto de entrenamiento.

```
rf.fit(X_train, y_train)
```

Las métricas de evaluación utilizadas para evaluar el rendimiento del modelo son la precisión, la sensibilidad (recall) y la puntuación F1. Estas métricas se calculan como sigue:

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

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Donde TP es Verdadero Positivo, TN es Verdadero Negativo, FP es Falso Positivo y FN es Falso Negativo.

El modelo de Random Forest es un algoritmo robusto y versátil para tareas de clasificación y regresión. Su capacidad para manejar un gran número de características y su flexibilidad en la configuración de parámetros lo convierten en una excelente opción para una variedad de problemas de aprendizaje automático.

Los parámetros del modelo de Random Forest se eligieron de la siguiente manera:

- **n_estimators=200**: Se eligieron 200 árboles para asegurar que el modelo sea lo suficientemente complejo como para capturar patrones en los datos.
- **max_depth=10**: Se limitó la profundidad máxima de los árboles a 10 para evitar el sobreajuste.
- **min_samples_leaf=2**: Se requieren al menos 2 muestras para formar una hoja, lo que ayuda a evitar árboles demasiado complejos.
- **min_samples_split=2**: Se requieren al menos 2 muestras para dividir un nodo, lo que también ayuda a evitar el sobreajuste.
- **bootstrap=False**: Se decidió no utilizar bootstrap para darle al modelo la oportunidad de utilizar todo el conjunto de datos para el entrenamiento.

El modelo de Random Forest funciona construyendo múltiples árboles de decisión durante el entrenamiento y luego promediando los resultados para mejorar la precisión y controlar el sobreajuste. En el contexto del Titanic, cada árbol toma en cuenta diferentes combinaciones de características y condiciones para hacer una predicción sobre si un pasajero sobrevivirá o no.

Por ejemplo, un árbol podría considerar principalmente la clase del pasajero y el sexo, mientras que otro podría centrarse en la edad y el número de hermanos o cónyuges a bordo. Al final, el modelo toma la "votación"de todos los árboles para hacer una predicción final.

El modelo logró una precisión de aproximadamente 0.8268 en el conjunto de validación y un puntaje de 0.7887 en la competencia de Kaggle, lo que indica un rendimiento bastante bueno. Las métricas adicionales como la precisión, la sensibilidad y la puntuación F1 también mostraron resultados prometedores.

c) K-Nearest neighbor

Preprocesamiento de Datos: Para la limpieza de este modelo se realizaron diversas modificaciones específicamente en nuestra base de datos de entrenamiento, "train". Primeramente se combinaron algunas variables que podrian manejarse juntas para evitar datos atípicos innecesarios como lo son combinar los padres de familia junto con los hermanos asumiendo que son de la misma familia. Después se procedió a eliminar columas que no son necesarias para este modelo como el nombre de los pasajeros, el ticket, la tarifa del boleto y la cabina de cada pasajero. Finalmente para tener un poco mas de orden con los datos restantes al momento de limpiarlos, se decidió por crear unas variables dummies para poder tener

la información por categorías cambiando los tres tipos de embarcamiento (S, C, Q) por números (1, 2, 3) y de igual manera pero con el genero (masculino = 1 y femenino=0). Todo este mismo procedimiento se realizo de igual manera para la base de datos "test"para así utilizarla en la implementación del modelo.

División de datos de entrenamiento y prueba: Primero, dividimos nuestros datos en dos partes: datos de entrenamiento y datos de prueba. Lo hicimos para poder enseñar a nuestro modelo y luego evaluar cuán bien podemos hacer predicciones en datos nuevos.

División de datos de entrenamiento y prueba: Utilizamos estos datos para enseñarle a nuestro modelo cómo predecir quién sobrevivió y quién no en el barco. En Xtrain, incluimos características como la clase de boleto (Pclass), la edad (Age), el tamaño de la familia (Family) y la tarifa del boleto (Fare). ytrain contenía información sobre si cada pasajero sobrevivió (1) o no (0).

Datos de prueba (Xtest): Estos datos son para evaluar qué tan bien podemos hacer predicciones nuestro modelo en nuevos datos. También tienen las mismas características que Xtrain, pero no tienen las etiquetas de supervivencia.

Entrenamiento de un modelo KNN: Luego, elegimos utilizar un algoritmo llamado K Vecinos Más Cercanos (KNN) para construir un modelo de predicción. El número 17 que ajustamos como n neighbors es una configuración que podemos cambiar según nuestras necesidades.

Entrenamiento del modelo: El modelo KNN se .^{en}trenaütilizando los datos de entrenamiento (Xtrain y ytrain). Esto significa que el modelo aprende de estos datos cómo se relacionan las características (como la clase de boleto, la edad, la familia y la tarifa) con la supervivencia.

Realización de predicciones: Una vez que el modelo está entrenado, lo usamos para hacer predicciones sobre quién sobrevivió en un conjunto de datos de prueba (Xtest). El modelo tomó las características de cada pasajero en Xtest y trató de predecir si sobrevivieron o no. Luego, guardamos esas predicciones en la variable ypred.

Verificación de la longitud de los resultados: Finalmente, verificamos que la cantidad de predicciones en ypred coincidiera con la cantidad de datos en el conjunto de prueba (Xtest). Esto era importante para asegurarnos de que el modelo hiciera una predicción para cada pasajero en el conjunto de prueba. Si la longitud de ypred y Xtest no coincidiera, podría indicar un problema en nuestro código.

Rendimiento: Después de entrenar nuestro modelo K Vecinos Más Cercanos (KNN), evaluamos su rendimiento utilizando una métrica llamada .^{a}ccuracy"(exactitud), que mide la proporción de predicciones correctas en comparación con el total de predicciones realizadas.

En nuestros datos de prueba, el modelo alcanzó una precisión (accuracy) de aproximadamente 0.7017543859649122. Esto significa que acertó 70 %. de las veces al predecir si un pasajero sobrevivió o no en función de las características proporcionadas.

Sin embargo, es importante señalar que al cargar nuestras predicciones en la plataforma Kaggle, obtuvimos un puntaje de precisión de aproximadamente 0.60 en el conjunto de datos de prueba de Kaggle. Esto sugiere que nuestro modelo podría beneficiarse de mejoras adicionales.

En resumen, aunque nuestro modelo tiene un rendimiento decente con una precisión del 70 %., aún estamos trabajando en mejorarlo para lograr resultados aún más precisos, especialmente teniendo en cuenta la puntuación más baja que obtuvimos en Kaggle

5. Análisis

La evaluación de los modelos en Kaggle se realizó al generar un archivo .csv conformado por una columna de Passenger ID y otra columna con las predicciones. A continuación, las mejores evaluaciones de los modelos

Best predictions

- SVM: 0.72
- Random Forest: 0.7887
- KNN: 0.7 en python, 0.6 en Kaggle

El modelo con mejor precisión fue el de Random Forest con precision de 0.7887 y el peor fue

6. Conclusión

Es fundamental emplear modelos de machine learning como K-Means, Support Vector Machines (SVM) y Random Forest debido a su versatilidad y eficacia en la resolución de una amplia gama de problemas. K-Means es útil para la segmentación de datos, SVM brilla en la clasificación y regresión, y Random Forest ofrece alta precisión y resistencia al sobreajuste. Estos modelos son adaptables, interpretables y pueden manejar datos ruidosos, lo que los convierte en herramientas esenciales para la toma de decisiones basada en datos y la solución de problemas en el mundo real.

Con base en los resultados obtenidos, podemos concluir con que el modelo de Random Forest es el que muestra un mejor comportamiento ya que es el que tiene un mejor score en la predicción de la competencia en Kaggle; sin embargo, aun esta pendiente el ajuste de los hiperparametros de los tres modelos para poder concluir de manera definitiva cual sera el modelo final que impleemnatremos.

Bibliografía