Reto equipo 4: Supervivencia en el Titanic

Ana Cardenas, Diego Rodríguez
Elias Garza, José Romo

13 de septiembre de 2023

Resumen

Este proyecto de investigación se enfoca en el análisis de datos y la construcción de modelos de Machine Learning aplicados al conjunto de datos del Titanic. A través de un proceso exhaustivo de limpieza, preprocesamiento y selección de modelos, se busca desarrollar un enfoque integral para predecir la supervivencia de pasajeros en el histórico naufragio del Titanic. Se exploran modelos de Regresión Logística, Naive Bayes, Random Forest y Support Vector Machine, con énfasis en la optimización de hiperparámetros y la evaluación de métricas de desempeño. La interfaz gráfica desarrollada permite la interacción de los usuarios con los modelos. Los resultados revelan que el modelo Random Forest con profundidad máxima 10 se destaca con un accuracy del 84

1. Limpieza y procesamiento de datos

En el ámbito del aprendizaje automático y la ciencia de datos, la calidad de los datos es un factor crítico para el éxito de cualquier proyecto. La base de datos del Titanic, disponible en la plataforma Kaggle (https://www.kaggle.com/competitions/titanic/), es un recurso ampliamente conocido y utilizado que ofrece una rica fuente de información sobre los pasajeros del famoso naufragio del RMS Titanic. Este conjunto de datos es un escenario ideal para desarrollar habilidades en limpieza, tratamiento y transformación de variables, con el objetivo de preparar los datos de manera efectiva para su uso en modelos de machine learning clasificadores.

Esta primera parte del reporte se centra en el proceso de preparación de datos aplicado a la base de datos del Titanic. Exploraremos las etapas clave de la limpieza de datos, el tratamiento de valores faltantes y atípicos, así como la transformación de variables para garantizar que los datos estén listos para alimentar modelos de clasificación. Además, analizaremos las características relevantes de este conjunto de datos, que incluyen información sobre los pasajeros, como su edad, género, clase de cabina y si sobrevivieron o no al naufragio.

La base de datos del Titanic es ampliamente utilizada en la comunidad de ciencia de datos y aprendizaje automático como un punto de partida para proyectos de clasificación y predicción. Su disponibilidad en Kaggle la hace accesible para estudiantes y profesionales

por igual, y proporciona una excelente oportunidad para aplicar técnicas de limpieza y preparación de datos en un contexto real.

1.1. Descripción de la Base de Datos

La base de datos contiene la información de los pasajeros del Titanic de 1912, embarcación que terminó siendo hundida por su colisión con un iceberg. En la base se tienen diversos datos entre ellos la clasificación de si este viajero sobrevivió al accidente o falleció. Entre las demás variables disponibles se encuentran:

1. **PassengerId** [int]: Es un identificador único asignado a cada pasajero.
2. **Survived** [bool]: Indica si el pasajero sobrevivió o no al naufragio. Un valor de 0 significa que no sobrevivió, y un valor de 1 significa que sí sobrevivió.
3. **Pclass (Passenger Class)** [int]: Representa la clase en la que el pasajero viajaba. Puede tener los valores 1 (primera clase), 2 (segunda clase) o 3 (tercera clase).
4. **Name** [str]: El nombre del pasajero.
5. **Sex** [str]: El género del pasajero, que puede ser 'male' (hombre) o 'female' (mujer).
6. **Age** [float]: La edad del pasajero en años. Algunas edades pueden estar representadas como fracciones si los pasajeros eran bebés.
7. **SibSp (Siblings/Spouses Aboard)** [int]: El número de hermanos o cónyuges que el pasajero tenía a bordo.
8. **Parch (Parents/Children Aboard)** [int]: El número de padres o hijos que el pasajero tenía a bordo.
9. **Ticket** [str]: El número de la boletería del pasajero.
10. **Fare** [float]: El precio del boleto que el pasajero pagó.
11. **Cabin** [str]: El número de cabina en la que el pasajero estaba alojado. Algunos valores pueden estar ausentes debido a la falta de registros.
12. **Embarked** [char]: El puerto donde el pasajero embarcó. Puede tener los valores 'C' (Cherbourg), 'Q' (Queenstown) o 'S' (Southampton).

Cabe mencionar que además de estas columnas principales, puede haber otras columnas secundarias en la base de datos con información adicional, pero estas son las principales que se utilizan comúnmente en análisis y modelado predictivo.

1.2. Limpieza y transformación de las variables

Para la exploración inicial se revisará los tipos de datos y sus valores faltantes.

```
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
```

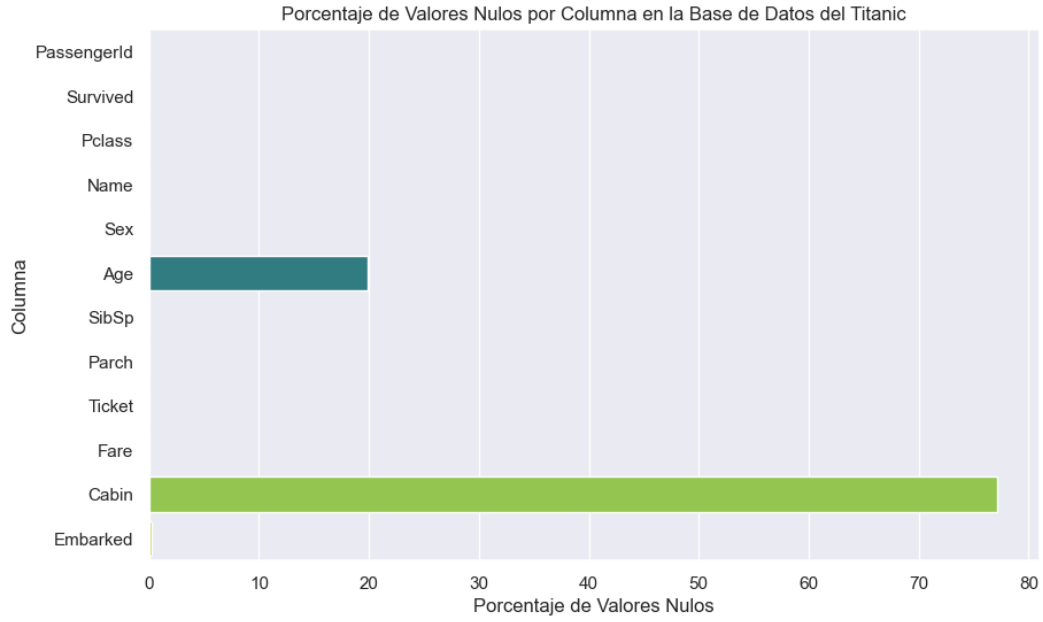

Figura 1: Porcentaje de datos faltantes en variables

Para mejorar la calidad y la eficiencia de nuestros modelos de Machine Learning, decidimos realizar una selección de características, excluyendo las siguientes columnas de la base de datos:

- **Passenger Id**: Esta columna representa un número de identificación único asignado a cada pasajero y no guarda ninguna relación significativa con la supervivencia de los mismos.
- **Cabin**: La columna 'Cabin' contiene una cantidad considerable de datos nulos, lo que dificulta su uso efectivo en nuestros modelos. Es posible que en futuras pruebas consideremos transformar esta columna en una variable binaria para distinguir entre aquellos pasajeros que tenían una cabina asignada y aquellos que no.
- **Name**: Aunque el nombre de los pasajeros no debería influir directamente en su supervivencia, hemos decidido mantener la posibilidad de agrupar a las personas por apellido en futuros análisis, ya que es probable que las familias viajaran juntas y su supervivencia podría estar relacionada. Sin embargo, esta acción se realizará si se considera necesario en el futuro.
- **Ticket**: Similar al nombre, la columna 'Ticket' en principio no debería afectar la supervivencia. Sin embargo, hemos optado por mantener la opción de agrupar a las personas que compraron boletos consecutivos, ya que es probable que fueran conocidos o amigos y viajaran juntos en el barco al momento de la evacuación.

Esta selección de características tiene como objetivo simplificar el conjunto de datos y centrarnos en las variables más relevantes para nuestro análisis y modelado predictivo.

1.3. Transformación de datos

Podemos quitar algunos outliers. Sin embargo, si vemos los diagramas de caja podemos observar que los valores atipicos importantes estan en edad y costo del boleto y no consideramos que estos sean valores verdaderamente atipicos. En el caso de la edad es muy posible que haya pesonas mayores que haya que considerar y tambien eliminar a el grupo de personas que pago considerablemente más esta mal ya que justo son un grupo importante a considerar. De hecho pueden ser atipicos porque son menos la cantidad de personas de clase alta pero es importante para la supervicencia este factor.

Es de interés remover los registros que tienen un valor de 'Fare' muy por arriba de la media. En la gráfica podemos notar como hay unos registros puntuales incluso por encima de 500 los cuales se consideran podría afectar en los cálculos de las medias y generar más ruido en los modelos que ayudar. Son 3 registros con el valor de 'Fare' que se desea remover.

Tenemos algunos datos faltantes en la variable de edad y la variable de Embarked. No queremos perder esos datos por lo que en el caso de la variable edad la rellenaremos con el promedio de la columna usando un k-nearest-neighbors para rellenar con el promedio de grupos similares. Por parte de la variable de Embarked usaremos la moda. Por otra parte, los modelos que vamos a utilizar necesitan valores numericos por lo que vamos a codificar las variables categóricas. En este caso utilizaremos un One-Hot encoding para generar las columnas necesarias para las variables 'Embarked' y 'Sex'.

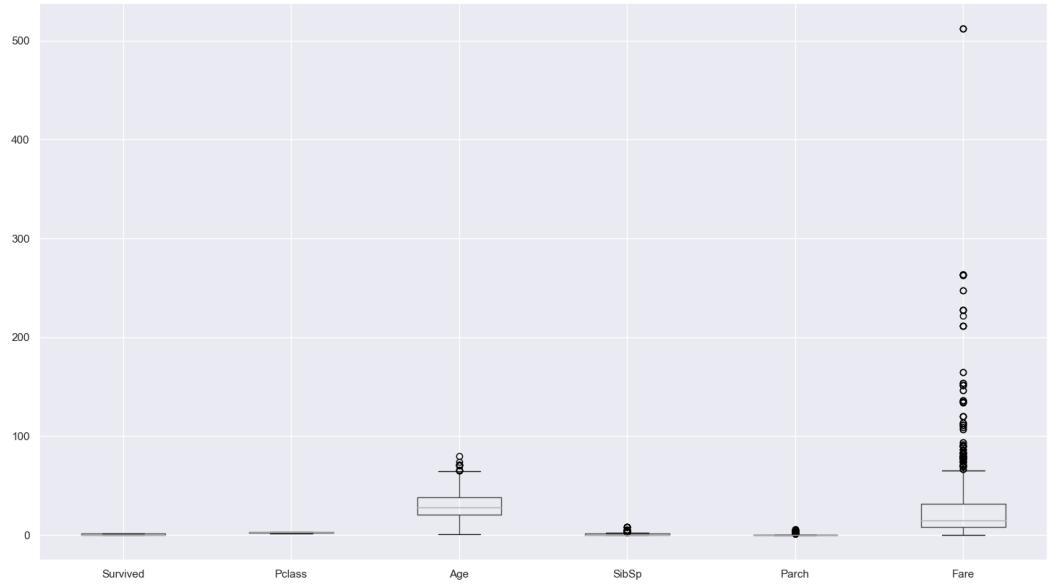

Figura 2: Boxplot inicial de variables numéricas

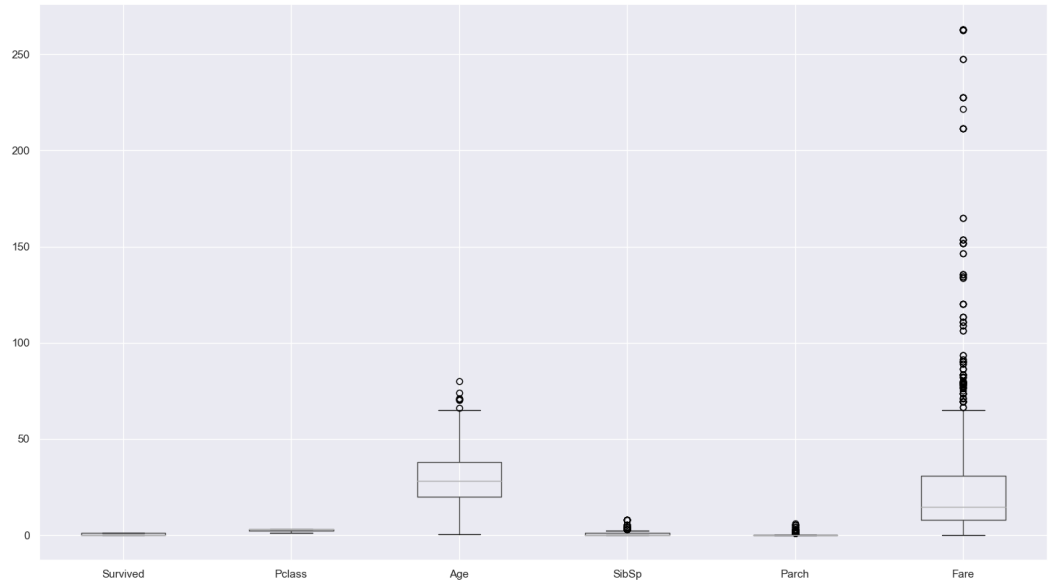

Figura 3: Boxplot de variables numéricas tras remover outliers seleccionados

De manera que corrigiendo esos puntos en la base de datos tenemos las siguientes variables.

```
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Survived    888 non-null    int64
 1   Pclass      888 non-null    int64
 2   Age         888 non-null    float64
 3   SibSp       888 non-null    int64
 4   Parch       888 non-null    int64
 5   Fare        888 non-null    float64
 6   Embarked_C  888 non-null    bool
 7   Embarked_Q  888 non-null    bool
 8   Embarked_S  888 non-null    bool
 9   is_male     888 non-null    bool
```

Por último, analizaremos la relación de las variables para determinar si conservamos todas o descartamos algunas.

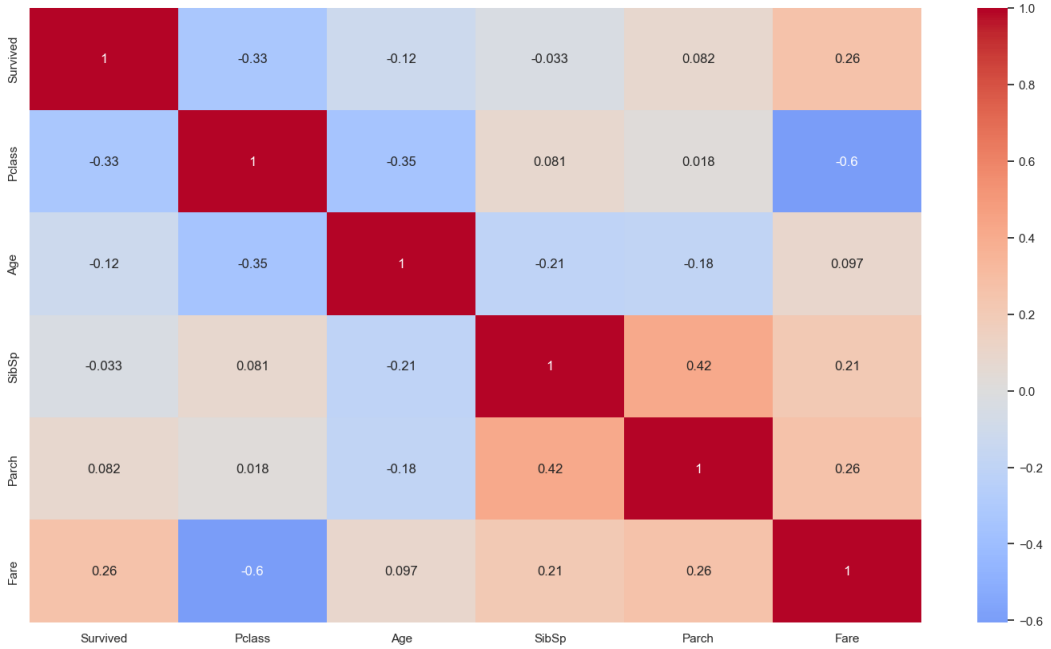

Figura 4: Correlación de variables

Ya que ‘Pclass‘ y ‘Fare‘ tienen una correlación considerada alta (0.6) se debe elmiminar del data set una de ellas. Por multicolinearidad, se dropeará la columna ded ‘Fare‘ ya que tiene menor correlación con la variable objetivo. De manera que nuestro dataframe final limpio para usarse en los modelos clasificadores es el siguiente:

Survived	**Pclass**	**Age**	**SibSp**	**Parch**	**Emb_C**	**Emb_Q**	**Emb_S**	**is_male**
0	3	22.0	1	0	False	False	True	True
1	1	38.0	1	0	True	False	False	False
1	3	26.0	0	0	False	False	True	False
1	1	35.0	1	0	False	False	True	False
0	3	35.0	0	0	False	False	True	True

Cuadro 1: Datos limpios para entrenar los modelos.

2. Reto Selección, configuración y entrenamiento del modelo

2.1. Variables

Las variables que se seleccionaron ("**Survived**", "**Pclass**", "**Age**", "**SibSp**", "**Parch**", "**Emb_C**", "**Emb_Q**", "**Embarked_S**", "**is_male**") se eligieron en función de su potencial para influir en la supervivencia de los pasajeros y su capacidad para aportar información significativa al modelo. Las demás variables se eliminan debido a su falta de relevancia o a la dificultad de interpretación. Esta selección cuidadosa de variables contribuye a la creación de un modelo más eficiente y efectivo para predecir la supervivencia de los pasajeros del Titanic.

Dado que la base de datos ya está limpia y contiene variables como “Survived”, “Pclass”, “Age”, “SibSp”, “Parch”, “Embarked_C”, “Embarked_Q”, “Embarked_S” y “is_male”, tenemos un conjunto de datos que combina variables numéricas y categóricas codificadas en formato numérico o binario. Vamos a analizar cómo estas variables se ajustan a los siguientes modelos de clasificación: Regresión Logística, Random Forest, Support Vector Machine (SVM) y Gaussian Naive Bayes.

1. **Regresión Logística**:
   - **Compatibilidad con datos**: La regresión logística es adecuada para datos que contienen variables numéricas y categóricas. Puede manejar variables binarias (como “is_male”) y variables continuas (como “Age”).
   - **Ventajas**: Es un modelo simple e interpretable. Ofrece probabilidades de clasificación. Funciona bien cuando las clases son linealmente separables.
   - **Desventajas**: Puede no funcionar bien si las relaciones son altamente no lineales. No captura interacciones complejas entre características.
2. **Random Forest**:
   - **Compatibilidad con datos**: Los Random Forests son versátiles y pueden manejar una combinación de características numéricas y categóricas.
   - **Ventajas**: Pueden manejar relaciones no lineales. Son robustos frente a overfitting. Pueden manejar variables irrelevantes.

- **Desventajas**: Pueden ser más difíciles de interpretar que modelos simples. Pueden sobreajustar en conjuntos de datos pequeños.

3. **Support Vector Machine (SVM)**:
   - **Compatibilidad con datos**: Los SVM pueden manejar tanto características numéricas como categóricas codificadas.
   - **Ventajas**: Pueden manejar relaciones no lineales utilizando el truco del kernel. Son eficaces en espacios de alta dimensión. Buen rendimiento en problemas con separación clara.
   - **Desventajas**: Puede ser computacionalmente costoso en conjuntos de datos grandes. La selección del kernel y los parámetros puede ser un desafío.
4. **Gaussian Naive Bayes**:
   - **Compatibilidad con datos**: El Naive Bayes es adecuado para características numéricas y categóricas. La asunción de independencia condicional entre características puede ser cuestionable en ciertos casos.
   - **Ventajas**: Simple y rápido. Puede funcionar bien incluso con supuestos incumplidos. Bueno para datos pequeños.
   - **Desventajas**: Supone independencia condicional, lo que puede no ser cierto en la realidad. No captura interacciones entre características.

En resumen, todas las variables seleccionadas (“Survived”, “Pclass”, “Age”, “SibSp”, “Parch”, “Embarked_C”, “Embarked_Q”, “Embarked_S”, “is_male”) son compatibles con los modelos de clasificación mencionados. La elección del modelo dependerá de la naturaleza de los datos y de la complejidad de las relaciones entre las características. La Regresión Logística es una buena opción inicial debido a su simplicidad y interpretabilidad. Los Random Forests son adecuados para relaciones no lineales y para evitar overfitting. Los SVM son útiles cuando las clases están bien separadas. El Naive Bayes es una opción simple que funciona bien en casos con pocos datos. La selección del modelo debe considerar tanto el rendimiento predictivo como la interpretabilidad de los resultados.

1. **Random Forest**:

   Se ajustará el parámetro `max_depth`, que controla la profundidad máxima de los árboles en el bosque. Al modificar este valor, se puede controlar la complejidad de los árboles y, por lo tanto, el potencial de overfitting.

2. **Support Vector Machine (SVM)**:

   Se variará el parámetro `kernel`, que determina el tipo de función kernel utilizada para transformar los datos en un espacio de mayor dimensión. Las opciones comunes son lineal, polinómica y radial (RBF). Además, se podría ajustar el grado (`degree`) en caso

de usar un kernel polinómico, que controla la complejidad de las transformaciones polinómicas.

3. **Naive Bayes**:

   Se ajustará la variable `var_smoothing` que controla la suavización (smoothing) de las probabilidades de características. En Naive Bayes, la probabilidad de que una característica específica ocurra en una clase puede volverse cero si no se ha observado en el conjunto de entrenamiento. La suavización (smoothing) permite evitar este problema al agregar una cantidad pequeña a todas las frecuencias de características, evitando así probabilidades nulas y mejorando la generalización del modelo. Al ajustar `var_smoothing`, podemos equilibrar la influencia de las características poco comunes y encontrar un compromiso óptimo entre suavización y capacidad de adaptación a los datos.

4. **Logistic Regression**:

   Se variará `C` y `penalty`. **C (Inverse of Regularization Strength)** controla el nivel de regularización en el modelo. Un valor más bajo de `C` aumenta la regularización, lo que previene el sobreajuste, mientras que un valor más alto permite un ajuste más flexible a los datos de entrenamiento. **Penalty (Type of Regularization)** determina el tipo de regularización aplicada (L1 o L2). L1 tiende a conducir a la selección de características al hacer que algunos coeficientes sean exactamente cero, lo que puede ser útil para reducir la complejidad del modelo. L2 penaliza coeficientes grandes, evitando que tomen valores extremadamente altos.

Ajustar estos hiperparámetros permite encontrar un equilibrio entre la capacidad del modelo para ajustarse a los datos de entrenamiento y su capacidad de generalización a nuevos datos, lo que es esencial para un rendimiento óptimo en la regresión logística.
Al comparar las diferentes configuraciones de hiperparámetros para cada modelo, se busca identificar cuál es el conjunto de parámetros que produce el mejor rendimiento en la predicción de la supervivencia de los pasajeros del Titanic. Este proceso es fundamental para asegurarse de que los modelos estén optimizados y puedan generalizar bien a datos nuevos y no vistos durante el entrenamiento.

2.2. Métricas

Para evaluar y comparar el rendimiento de los modelos, se realizará un proceso de ajuste de hiperparámetros utilizando diferentes configuraciones para cada uno. En particular, se variarán los siguientes parámetros de los modelos:

2.3. Split de dataset

Se usará una separación 80/20 para entrenar los modelos. La razón detrás de esta división 80/20 es encontrar un equilibrio entre el tamaño suficiente del conjunto de entrenamiento para

que el modelo pueda aprender de manera efectiva y un conjunto de prueba lo suficientemente grande como para proporcionar una evaluación sólida del rendimiento del modelo. Esta técnica ayuda a evitar el sobreajuste, ya que el modelo no ve los datos de prueba durante el entrenamiento y, por lo tanto, se evalúa en datos no vistos.

2.4. Support Vector Machine

Para el modelo de SVM usaremos distintos kernels y distintos grados para el kernel polinomial. Los kernels en SVM son funciones que transforman los datos originales en un espacio de mayor dimensión donde las clases sean más fácilmente separables. Diferentes kernels representan diferentes formas de separación de clases y pueden capturar relaciones más complejas entre los datos. Los kernels comunes incluyen:

- **Linear Kernel**: Para separaciones lineales.
- **RBF Kernel**: Para separaciones no lineales y complejas.
- **Polynomial Kernel**: Para relaciones polinómicas de grado superior.

El objetivo de probar diferentes kernels es identificar cuál de ellos se ajusta mejor a la naturaleza de los datos y produce una mejor separación de las clases. Al probar varios kernels, podemos entender cómo cada uno se adapta a la estructura subyacente de los datos y cuál proporciona un mejor rendimiento en el problema específico de predicción de supervivencia en el Titanic.

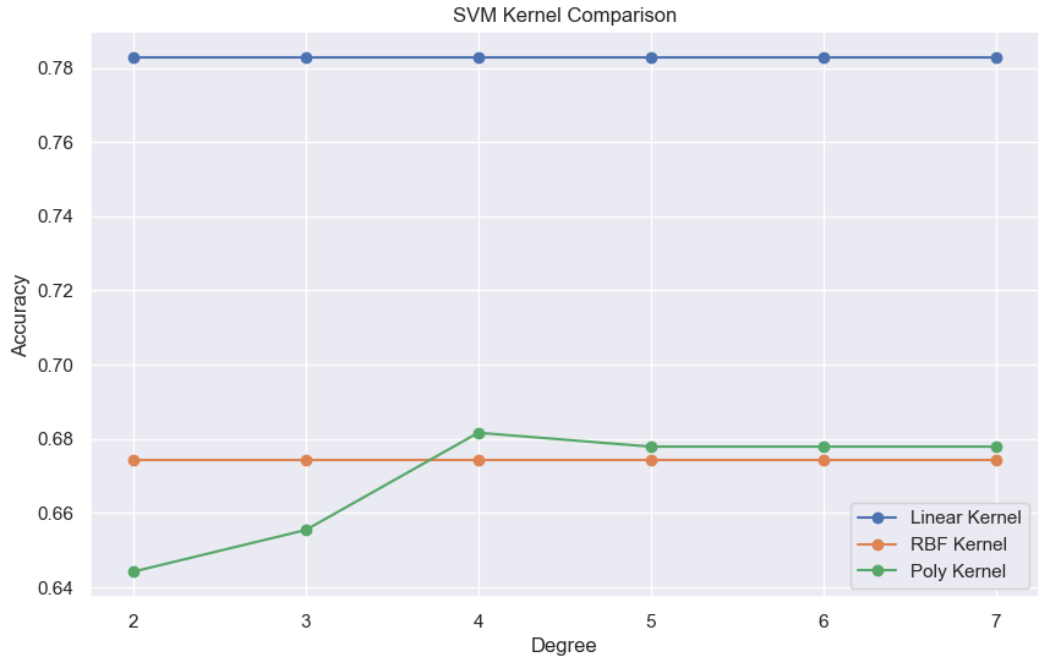

Figura 5: Rendimiento del SVM cambiando parámetros

El mejor modelo en cuanto accuracy para el Support Vector Machine resultó ser el de Kernel lineal. Este modelo tiene las siguientes métricas:

```
Informe de Clasificación:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84       172
           1       0.72      0.63      0.67        95

    accuracy                           0.78       267
   macro avg       0.77      0.75      0.76       267
weighted avg       0.78      0.78      0.78       267
```

- **Accuracy (Precisión Global):** El modelo SVM tiene una precisión global del 78 %, lo que significa que acertó correctamente la supervivencia o no supervivencia de los pasajeros en aproximadamente el 78 % de las predicciones en el conjunto de prueba. Esto indica un rendimiento decente pero puede haber margen para mejoras.
- **Precision (Precisión):** La precisión para la clase 0 (no sobrevivientes) es del 81 %, lo que indica que el 81 % de las predicciones positivas para esta clase fueron correctas. La precisión para la clase 1 (sobrevivientes) es del 72 %, lo que significa que el 72 % de las predicciones positivas para esta clase fueron correctas. En general, el modelo tiene una precisión bastante equilibrada en ambas clases, aunque es ligeramente mejor en la clasificación de no sobrevivientes.
- **Recall (Recuperación o Sensibilidad):** El recall para la clase 0 (no sobrevivientes) es del 87 %, lo que indica que el 87 % de los verdaderos no sobrevivientes fueron identificados correctamente por el modelo. Para la clase 1 (sobrevivientes), el recall es del 63 %, lo que significa que el 63 % de los verdaderos sobrevivientes fueron identificados correctamente. Esto sugiere que el modelo es mejor para identificar no sobrevivientes que sobrevivientes.
- **F1-Score (Puntuación F1):** El F1-score es una medida que combina precisión y recall en una sola métrica. El F1-score para la clase 0 es 0.84 y para la clase 1 es 0.67. Un F1-score más alto indica un mejor equilibrio entre precisión y recall.

En resumen, el modelo SVM parece funcionar razonablemente bien en términos de accuracy, con un rendimiento ligeramente mejor en la clasificación de no sobrevivientes. Sin embargo, puede haber margen para mejorar la capacidad del modelo para identificar correctamente a los sobrevivientes.

2.5. Gaussian Naive Bayes

El Gaussian Naive Bayes es un algoritmo que se basa en el teorema de Bayes y supone independencia entre características. Aunque simplista, puede ser útil en la clasificación del

Titanic debido a su eficacia en datos con múltiples características categóricas. Su enfoque rápido y adecuado para conjuntos pequeños, junto con su capacidad para manejar relaciones simples entre características, lo convierte en una opción viable para la predicción de supervivencia en este contexto.

Se ajustará la variable 'var_smoothing' que controla la suavización (smoothing) de las probabilidades de características. En Naive Bayes, la probabilidad de que una característica específica ocurra en una clase puede volverse cero si no se ha observado en el conjunto de entrenamiento.

Informe de Clasificación:

	precision	recall	f1-score	support
0	0.84	0.87	0.85	172
1	0.75	0.69	0.72	95
accuracy			0.81	267
macro avg	0.79	0.78	0.79	267
weighted avg	0.81	0.81	0.81	267

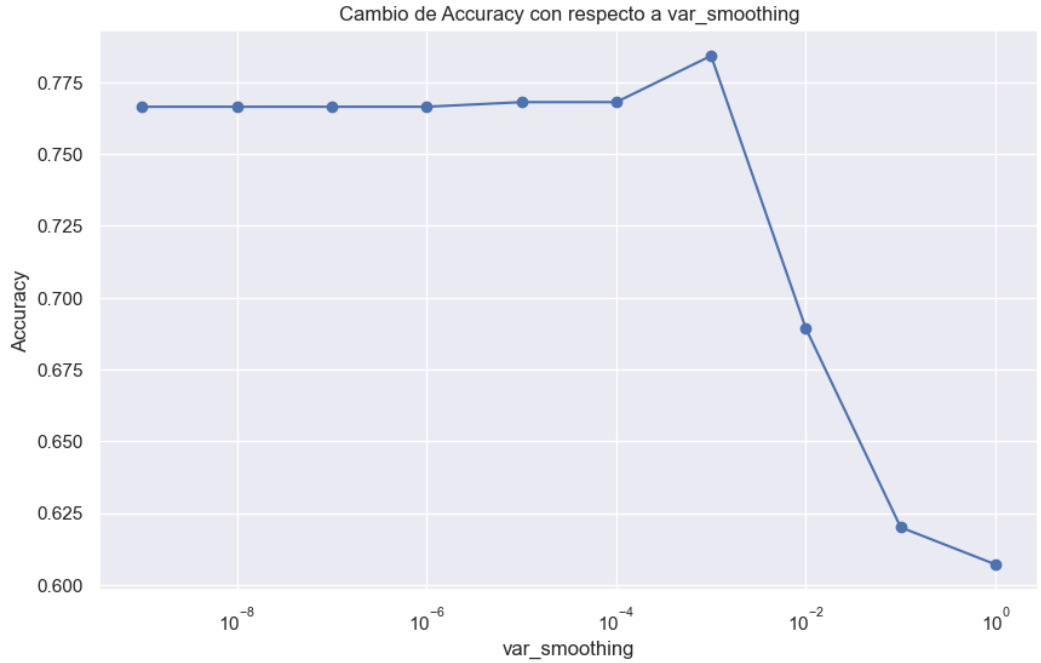

Figura 6: Rendimiento del Naive Bayes cambiando parámetros

- **Precision (Precisión)** para la clase 0 (no sobrevivientes) es del 84 %, lo que significa que el 83 % de las predicciones positivas para esta clase fueron correctas. Para la clase 1 (sobrevivientes), la precisión es del 75 %.

- **Recall (Recuperación o Sensibilidad)** para la clase 0 es del 87 %, lo que indica que el 90 % de los verdaderos no sobrevivientes fueron identificados correctamente por el modelo. Para la clase 1, el recall es del 69 %.
- **F1-Score (Puntuación F1)** es de 0.85 para la clase 0 y 0.72 para la clase 1.
- **Accuracy:** El modelo tiene una precisión global (accuracy) del 81 %, lo que significa que aproximadamente el 81 % de las predicciones son correctas en el conjunto de prueba (En la gráfica no llega a tocar ese punto ya que se visualiza el promedio del cross-fold validation, lo que provoca que pueda mostrarse menor o mayor dependiendo de los 5 test.)

En resumen, el modelo Naive Bayes tiene un rendimiento bastante sólido en la clasificación del conjunto de datos del Titanic, con una precisión global del 81 %. El modelo es mejor para clasificar no sobrevivientes (clase 0) que sobrevivientes (clase 1), como se evidencia en un recall más alto para la clase 0. El F1-Score muestra un buen equilibrio entre precisión y recall para ambas clases.

2.6. Logistic Regression

La regresion logística lo que hace es meter una función linear como parametro dentro de una función sigmoide lo que la manda a un rango entre 0 y 1. Se variará ‘C‘ y ‘penalty‘. **C (Inverse of Regularization Strength)** controla el nivel de regularización en el modelo. Un valor más bajo de ‘C‘ aumenta la regularización. **Penalty (Type of Regularization)** determina el tipo de regularización aplicada (L1 o L2). L1 tiende a conducir a la selección de características al hacer que algunos coeficientes sean exactamente cero, lo que puede ser útil para reducir la complejidad del modelo. L2 penaliza coeficientes grandes, evitando que tomen valores extremadamente altos.

```
Informe de Clasificación:
              precision    recall  f1-score   support

           0       0.83      0.89      0.86       172
           1       0.77      0.67      0.72        95

    accuracy                           0.81       267
   macro avg       0.80      0.78      0.79       267
weighted avg       0.81      0.81      0.81       267
```

- **Precision (Precisión)** para la clase 0 (no sobrevivientes) es del 83 %, lo que significa que el 77 % de las predicciones positivas para esta clase fueron correctas. Para la clase 1 (sobrevivientes), la precisión es del 81 %.

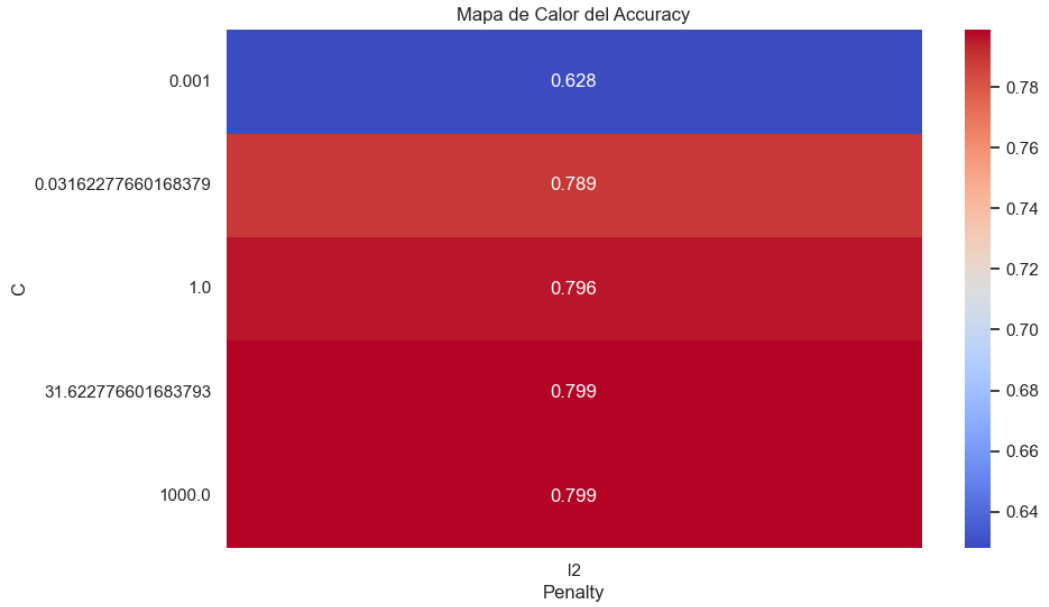

Figura 7: Rendimiento de la regresión logística cambiando parámetros

- **Recall (Recuperación o Sensibilidad)** para la clase 0 es del 89 %, lo que indica que el 67 % de los verdaderos no sobrevivientes fueron identificados correctamente por el modelo. Para la clase 1, el recall es del 68 %.
- **F1-Score (Puntuación F1)** es de 0.86 para la clase 0 y 0.72 para la clase 1.
- **Accuracy:** El modelo tiene una precisión global (accuracy) del 81 %, lo que significa que aproximadamente el 81 % de las predicciones son correctas en el conjunto de prueba. (En la gráfica no llega a tocar ese punto ya que se visualiza el promedio del cross-fold validation, lo que provoca que pueda mostrarse menor o mayor dependiendo de los 5 test pero el reporte y la matriz de confusión lo demuestran.)

En resumen, el modelo de Regresión Logística con los hiperparámetros optimizados muestra un rendimiento sólido en la clasificación del conjunto de datos del Titanic, con una precisión global del 81 %. Al igual que el modelo Naive Bayes, este modelo es mejor para clasificar no sobrevivientes (clase 0) que sobrevivientes (clase 1), como se evidencia en un recall más alto para la clase 0. El F1-Score muestra un buen equilibrio entre precisión y recall para ambas clases.

El mapa de calor mostraría cómo cambia el accuracy en función de las diferentes combinaciones de los hiperparámetros ‘C‘ y ‘penalty‘. Sin embargo, el modelo tiene problemas usando L1, por lo que únicamente se pudo ejecutar regresión logistica con penalización L2.

2.7. Random Forest

Para el modelo Random Forest variaremos el límite de profundidad de los árboles:

1. **None (Sin límite)**: Al incluir 'None', permitimos que los árboles en el bosque crezcan hasta su máxima profundidad, lo que significa que pueden ser muy profundos y complejos. Esta opción permitirá que el modelo capture detalles finos en los datos, pero también puede llevar al sobreajuste si no se controla adecuadamente.

2. **10, 20, 30, 40, 50**: Estos valores representan límites máximos de profundidad para los árboles. Comenzamos con valores más bajos (10) y aumentamos gradualmente en incrementos de 10 hasta 50. Esto permite explorar diferentes niveles de complejidad del modelo. Los valores más bajos tienden a crear modelos más simples y menos propensos al sobreajuste, mientras que los valores más altos pueden dar como resultado modelos más complejos.

```
Informe de Clasificación:
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       172
           1       0.79      0.72      0.75        95

    accuracy                           0.83       267
   macro avg       0.82      0.81      0.81       267
weighted avg       0.83      0.83      0.83       267
```

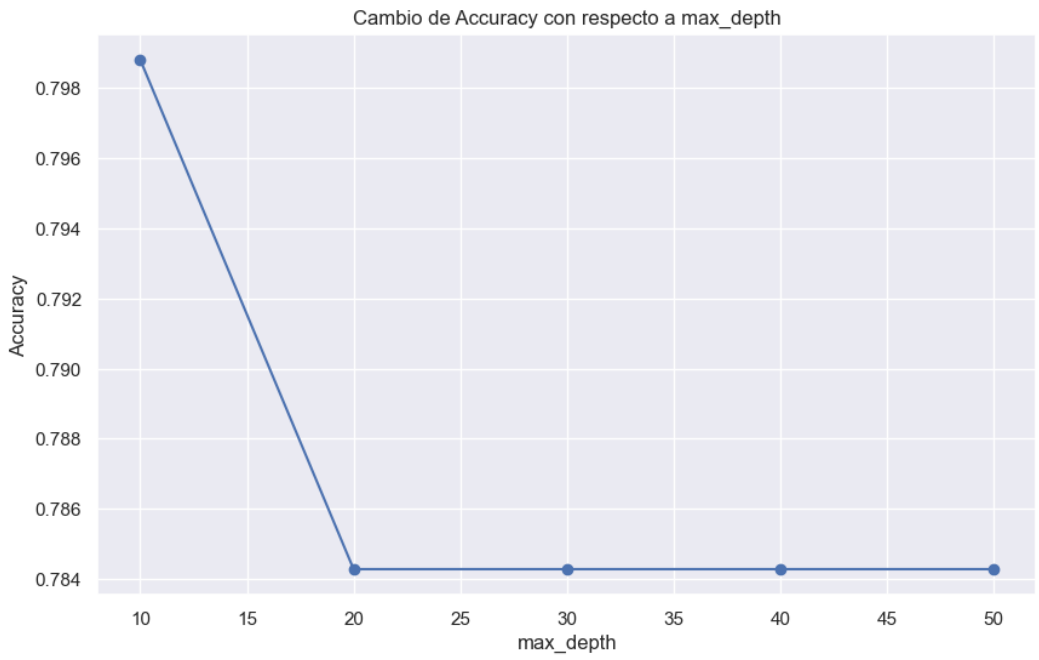

Figura 8: Rendimiento del Random Forest cambiando parámetros

- **Precision (Precisión)** para la clase 0 (no sobrevivientes) es del 85 %, lo que significa que el 85 % de las predicciones positivas para esta clase fueron correctas. Para la clase 1 (sobrevivientes), la precisión es del 79 %.
- **Recall (Recuperación o Sensibilidad)** para la clase 0 es del 90 %, lo que indica que el 90 % de los verdaderos no sobrevivientes fueron identificados correctamente por el modelo. Para la clase 1, el recall es del 72 %.
- **F1-Score (Puntuación F1)** es de 0.87 para la clase 0 y 0.75 para la clase 1.
- **Accuracy:** El modelo tiene una precisión global (accuracy) del 83 %, lo que significa que aproximadamente el 83 % de las predicciones son correctas en el conjunto de prueba.(En la gráfica no se muestra que llegue a este valor debido a que se grafica el promedio del cross-validation de 5 divisiones, lo que puede provocar que vea menor, pero en el reporte se imprime el mejor resultado).

En resumen, el modelo Random Forest con el mejor valor de `max_depth` encontrado en la búsqueda de cuadrícula muestra un rendimiento sólido en la clasificación del conjunto de datos del Titanic, con una precisión global del 83 %. Al igual que los modelos anteriores, este modelo también es mejor para clasificar no sobrevivientes (clase 0) que sobrevivientes (clase 1), como se evidencia en un recall más alto para la clase 0. El F1-Score muestra un buen equilibrio entre precisión y recall para ambas clases.

2.8. Comparativa de Modelos

Support Vector Machine (SVM):

- Accuracy: 0.78
- F1-Score (clase 0/clase 1): 0.84 / 0.67

Naive Bayes:

- Accuracy: 0.81
- F1-Score (clase 0/clase 1): 0.85 / 0.72

Logistic Regression:

- Accuracy: 0.81
- F1-Score (clase 0/clase 1): 0.86 / 0.72

Random Forest:

- Accuracy: 0.83
- F1-Score (clase 0/clase 1): 0.87 / 0.75

Selección de modelo:

1. **Accuracy**: Observamos que Random Forest tiene la mayor precisión global (83 %), lo que indica que es el modelo que acierta más predicciones en el conjunto de prueba en comparación con los otros modelos.

2. **F1-Score (clase 0/clase 1)**: El F1-Score es una métrica que combina precisión y recall. Random Forest tiene un F1-Score ligeramente superior para la clase 1 (sobrevivientes) en comparación con los otros modelos, lo que sugiere que es mejor para identificar correctamente a los sobrevivientes.

3. **Max Depth = 10**: Aunque el Random Forest con `max_depth` de 10 tiene un rendimiento ligeramente menor en términos de precisión en comparación con el Random Forest sin límite de profundidad (`max_depth=None`), es importante considerar que un modelo con una profundidad limitada generaliza mejor y es menos propenso al sobreajuste. Esto es especialmente importante en aplicaciones del mundo real donde se busca un buen equilibrio entre precisión y capacidad de generalización.

En resumen, se selecciona el modelo Random Forest con `max_depth=10` porque tiene una precisión global sólida y un F1-Score equilibrado para ambas clases, lo que sugiere un buen equilibrio entre precisión y capacidad de generalización. Además, la limitación de la profundidad del árbol ayuda a reducir el riesgo de sobreajuste.

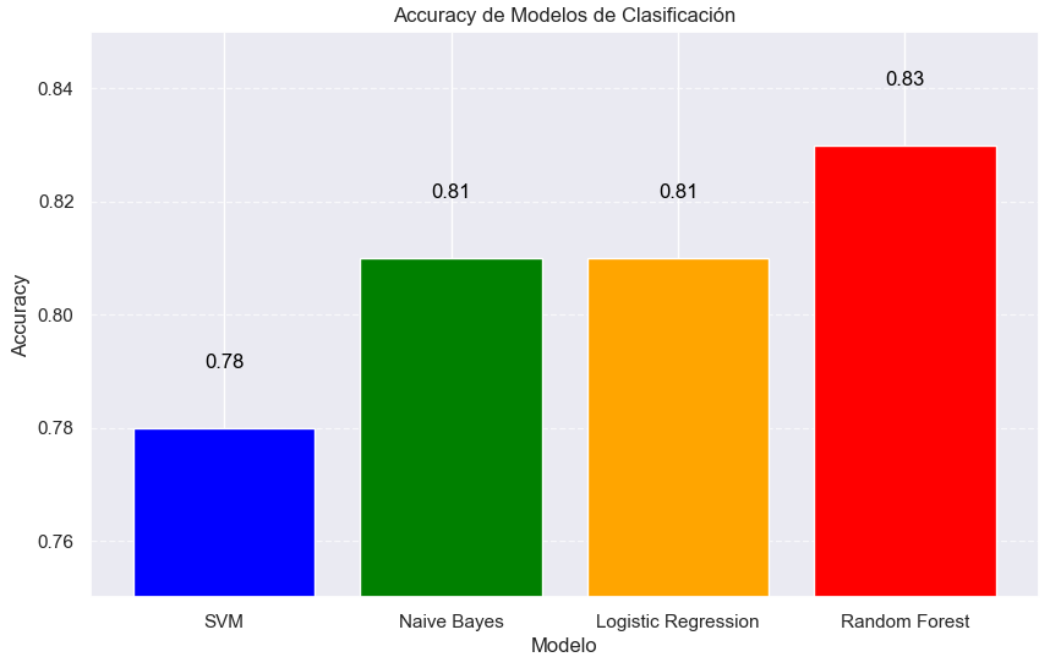

Figura 9: Comparativa de accuracy entre modelos

3. Evaluación y Refinamiento del modelo

Después de evaluar y comparar varios modelos de clasificación, entre ellos Support Vector Machine (SVM), Naive Bayes, Logistic Regression y Random Forest. Tras esta comparativa, seleccionamos el modelo Random Forest con una profundidad máxima de 10 como nuestra elección principal debido a su rendimiento superior en términos de accuracy y capacidad de generalización en relación a los demás modelos.

En esta siguiente sección, nos enfocaremos en la Evaluación y Refinamiento del Modelo Random Forest. A pesar de haber identificado un modelo prometedor, aún hay margen para mejorar su rendimiento mediante la exploración de más parámetros y la aplicación de técnicas de regularización. Exploraremos la afinación de hiperparámetros adicionales, la búsqueda de la mejor configuración y la evaluación de su impacto en las métricas de rendimiento. Con esto, nuestro objetivo es garantizar que nuestro modelo sea lo más preciso y generalizado posible para la tarea de predicción de supervivencia en el desafío del Titanic.

3.1. El detalle de la regularización en random forest

El modelo Random Forest no utiliza regularización de la misma manera que algunos otros modelos, como la Regresión Logística o las Máquinas de Vectores de Soporte (SVM). En cambio, el Random Forest controla el sobreajuste (overfitting) de manera inherente a través de su estructura de conjunto de árboles de decisión. Cada árbol en el conjunto se construye de manera independiente, y el proceso de combinación de los resultados de estos árboles tiende a reducir el sobreajuste de manera automática.

Hay algunas técnicas que se pueden considerar para 'regularizar' un modelo Random Forest:

1. **Limitar la Profundidad de los Árboles**: Esto se logra ajustando el hiperparámetro `max_depth` o `max_leaf_nodes`. Reducir la profundidad máxima de los árboles puede ayudar a prevenir el sobreajuste y hacer que el modelo sea más generalizado.

2. **Aumentar el Número de Árboles**: Agregar más árboles al conjunto (a través del hiperparámetro `n_estimators`) puede ayudar a mejorar la generalización del modelo. Sin embargo, esto también puede aumentar el tiempo de entrenamiento.

3. **Submuestreo de Características (Feature Sampling)**: El Random Forest permite submuestrear las características en cada árbol (a través del hiperparámetro `max_features`). Reducir la cantidad de características consideradas en cada árbol puede ayudar a evitar el sobreajuste.

4. **Bootstrap Aggregating (Bagging)**: El Random Forest utiliza un enfoque de bagging al entrenar sus árboles. Esto implica tomar muestras aleatorias con reemplazo de los datos de entrenamiento para entrenar cada árbol. Esta técnica reduce la varianza y puede ayudar a prevenir el sobreajuste.

5. **Out-of-Bag (OOB) Score**: El Random Forest proporciona una métrica de evaluación llamada "out-of-bag score". Esta métrica se basa en las muestras de entrenamiento que no se utilizaron para entrenar cada árbol. Puede servir como una medida de validación cruzada interna para evaluar el rendimiento y la generalización del modelo.

Aunque el Random Forest no utiliza regularización en el sentido tradicional, se puede ajustar sus hiperparámetros y considerar técnicas específicas para controlar el sobreajuste y mejorar su capacidad de generalización.

3.2. Optimización de hiperparámetros

Para refinar el modelo de random forest con `max depth = 10` ya seleccionado anteriormente, procederemos a hacer una busqueda ajustando los valores para `n_estimators` y `max_features` en Random Forest. Estos hiperparámetros son importantes porque tienen un impacto significativo en el rendimiento y la capacidad de generalización del modelo.

1. `n_estimators` (Número de Árboles en el Bosque):
    - **Subajuste vs. Sobreajuste**: El número de árboles en el bosque, controlado por `n_estimators`, es un hiperparámetro crítico. Un número pequeño de árboles puede llevar al subajuste, lo que significa que el modelo no captura adecuadamente los patrones en los datos. Por otro lado, un número excesivamente grande de árboles puede aumentar el tiempo de entrenamiento sin mejorar significativamente la generalización. La optimización de este hiperparámetro nos ayuda a encontrar un equilibrio adecuado.
2. `max_features` (Submuestreo de Características):
    - **Reducción de la Correlación entre Árboles**: `max_features` controla la cantidad de características consideradas en cada división de un nodo. Al ajustar este hiperparámetro, podemos reducir la correlación entre los árboles del bosque. Esto es importante porque reduce la varianza del modelo y ayuda a prevenir el sobreajuste. Submuestrear las características a menudo mejora la capacidad de generalización.

La optimización de estos hiperparámetros nos permite encontrar la combinación óptima que equilibra el sesgo y la varianza, lo que resulta en un modelo Random Forest que puede hacer predicciones precisas en nuevos datos no vistos.

```
Informe de Clasificación:
              precision    recall  f1-score   support

           0       0.85      0.90      0.88       172
           1       0.80      0.72      0.76        95

    accuracy                           0.84       267
   macro avg       0.83      0.81      0.82       267
weighted avg       0.83      0.84      0.83       267
```

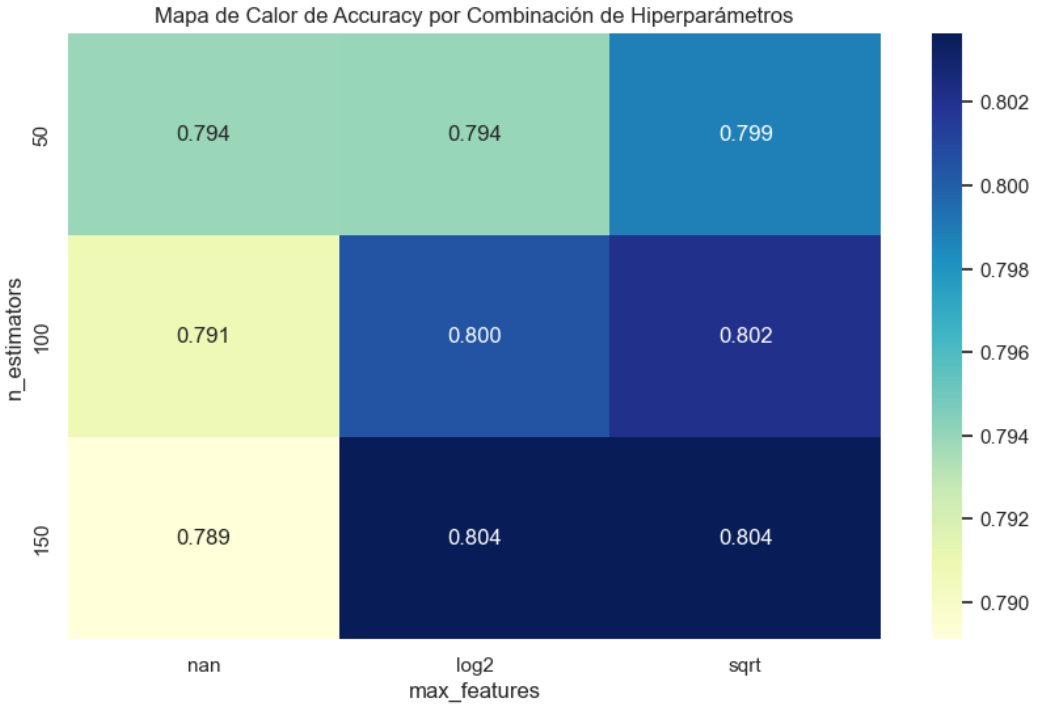

Figura 10: Comparativa de Accuracy con distintos parametros de Random Forest

- El modelo Random Forest con los mejores hiperparámetros (`max_features='sqrt'` y `n_estimators=150`) logra un accuracy del 84 %, lo que significa que aproximadamente el 84 % de las predicciones fueron correctas en el conjunto de prueba.
- La matriz de confusión muestra que el modelo hizo 155 predicciones correctas para la clase 0 (no sobrevivientes) y 68 predicciones correctas para la clase 1 (sobrevivientes). Cometió 17 falsos positivos y 27 falsos negativos.
- En cuanto al informe de clasificación, el modelo tiene una precisión promedio del 83 %, lo que indica un buen rendimiento general. La puntuación F1 es del 88 % para la clase 0 y del 76 % para la clase 1.

- Los mejores parámetros encontrados durante la optimización fueron `max_features='sqrt'` y `n_estimators=150`. Esto significa que el modelo utiliza la raíz cuadrada del número de características para cada división de nodo y consta de 150 árboles en el bosque.

En resumen, el modelo Random Forest con estos hiperparámetros ajustados logra un buen rendimiento en la tarea de predicción de supervivencia en el dataset del Titanic, con un buen equilibrio entre precisión y capacidad de generalización.

3.3. Sesgo y Varianza

- **Accuracy (Precisión):** El modelo tiene un accuracy del 84 %, lo que indica que la mayoría de las predicciones son correctas en el conjunto de prueba. Sin embargo, el accuracy por sí solo no nos dice si hay sesgo o varianza.
- **Matriz de Confusión:** La matriz de confusión muestra que el modelo cometió 17 falsos positivos y 27 falsos negativos. Esto sugiere que el modelo no es perfecto y tiene algunas confusiones, pero no parece haber un desequilibrio extremo.
- **Informe de Clasificación:** El informe de clasificación proporciona métricas adicionales. El recall (sensibilidad) para la clase 1 (sobrevivientes) es del 72 %. Esto significa que el modelo identifica correctamente alrededor del 72 % de los verdaderos sobrevivientes en el conjunto de prueba. Un recall relativamente menor podría indicar cierto sesgo hacia la clase mayoritaria (en este caso, la clase 0, que representa a los no sobrevivientes).

Basándonos en estos resultados, podemos hacer el siguiente diagnóstico:

1. **Sesgo (Underfitting):** El modelo no parece estar sufriendo de underfitting, ya que logra un buen accuracy general y no se observa un rendimiento extremadamente pobre en ninguna métrica. Sin embargo, el recall relativamente bajo para la clase 1 sugiere que el modelo podría estar perdiendo algunos casos de verdaderos sobrevivientes, lo que podría indicar un ligero sesgo hacia la clase mayoritaria.
2. **Varianza (Overfitting):** No parece haber evidencia clara de overfitting en este modelo. El hecho de que el modelo tenga un buen rendimiento en el conjunto de prueba y no muestre una gran discrepancia entre el rendimiento en el conjunto de entrenamiento y prueba sugiere que no hay una varianza excesiva.

En resumen, el modelo Random Forest parece tener un buen equilibrio entre sesgo y varianza, con un rendimiento general sólido en la tarea de predicción de supervivencia en el dataset del Titanic. El recall ligeramente menor en la clase 1 podría ser mejorado con ajustes adicionales, pero en general, no parece haber un problema significativo de sesgo o varianza en este caso.

4. Resumen Final e Interfaz Gráfica

Para abordar el problema de predicción de supervivencia en el Titanic, se realizaron investigaciones en libros y recursos técnicos relacionados con machine learning y clasificación, así como en artículos científicos que aborden problemas similares de predicción en datos de pasajeros de barcos. También se investigó la documentación y tutoriales relacionados con las bibliotecas y algoritmos utilizados, como Random Forest, Regresión Logística, Support Vector Machine y Naive Bayes. Además, se consultaron fuentes de información disponibles en Canvas relacionadas con el desafío específico para comprender mejor el contexto y las expectativas del proyecto.En cuanto a la interfaz grafica, streamlit es una librería de python la cual facilita la creación de aplicaciones web con visualización también en dispositivos móviles de una manera sencilla. Streamlit le permite a los desarrolladores de Python convertir/transformar sus scripts a aplicaciones web sin necesidad de conocimientos en desarrollo web o de interfaz. Streamlit no solamente nos permite crear aplicaciones web, también podemos crear interfaces intuitivas, visualizar datos, actualizar los datos en tiempo real (actualiza la interfaz cada vez que un usuario cambia datos o parámetros), es fácil de personalizar, y se puede integrar a bases de datos o APIs externos con las cuales se pueden crear visualizaciones dinámicas.

Se llevaron a cabo varios experimentos para desarrollar un modelo de machine learning efectivo para predecir la supervivencia en el Titanic. Estos experimentos incluyeron la exploración y preprocesamiento de datos, la selección de variables relevantes (Survived, Pclass, Age, SibSp, Parch, Embarked_C, Embarked_Q, Embarked_S, is_male), la construcción de modelos utilizando Random Forest, Regresión Logística, Support Vector Machine y Naive Bayes, y la evaluación de estos modelos utilizando métricas como el accuracy. Los resultados de los experimentos mostraron que el modelo Random Forest obtuvo el mejor rendimiento, con una precisión del 84 % cuando se utilizó una profundidad de 10, 150 estimadores (n_estimators) y un max_features de ‘sqrt’. Esto significa que el modelo es capaz de predecir con precisión si un pasajero del Titanic habría sobrevivido o no en función de las características proporcionadas. La interpretación de estos resultados sugiere que las variables seleccionadas desempeñan un papel importante en la predicción de la supervivencia.

Con el modelo de Random Forest entrenado y con un rendimiento satisfactorio, se procedió a su serialización para guardarlo en caché de Streamlit. Luego, se creó una interfaz web en Streamlit que permite a los usuarios ingresar datos de un pasajero del Titanic y obtener una predicción de supervivencia en tiempo real utilizando el modelo entrenado. Los usuarios pueden seleccionar la clase, edad, cantidad de hermanos o cónyuges, cantidad de padres o hijos, costo del boleto, puerto de embarque y sexo del pasajero, y el modelo proporcionará una predicción de supervivencia. El código está hosteado por la disponibilidad de cómputo en la nube de streamlit lo cual nos permite generar un link al cual podemos entrar desde cualquier navegador (Click Aquí para probar interfaz).

Figura 11: Interfaz Gráfica del modelo clasificador del Titanic

Repositorio e interfaz:
https://github.com/EliasGarzaV/RetoTitanicBloque1
https://retotitanicclaseia.streamlit.app/