Detección de Fraude en los Tratamientos Médicos

Joaquín Tomas Lisotti

Facultad CAECE, sede C.A.B.A

Seminario I de Ciencia de Datos

Lic. Alejandro Víctor Bartolomeo

23 de Julio de 2024

Resumen

Con la finalidad de proporcionar ayuda a las empresas médicas que tienen problemas respecto a los fraudes, se ha desarrollado un proceso de modelado mediante la metodología CRISP-DM el cual permitió desarrollar un modelo con gran efectividad, el cual permite a las empresas identificar fraudes con un AUC-ROC de 0.97, indicando una gran capacidad del modelo para distinguir entre casos fraudulentos y no fraudulentos.

Contexto Importante

Dado a los aumentos de los costos de la atención sanitaria, es importante para los paises y empresas realizar un uso lo más adecuado posible de los recursos escasos, el financiamiento, principal recurso cuenta con varias problemáticas. Por tal motivo, en este trabajo, nos enfocaremos en uno de los grandes problemas que tienen los centros médicos, el fraude, “El tamaño del mercado mundial de detección de fraude sanitario se estima en 2,32 mil millones de dólares en 2024 y se espera que alcance los 6,35 mil millones de dólares en 2029” (Mordor intelligence, 2024).

Objetivo del análisis

El objetivo del análisis realizado es desarrollar y evaluar un modelo predictivo a fin de poder detectar aquellos casos potencialmente fraudulentos y así ayudar a reducir las pérdidas financieras relacionadas con el fraude médico, consiguiendo una buena precisión para la detección de actividades fraudulentas se puede ayudar a las organizaciones a realizar una mejor distribución de sus recursos y por ende, brindar una mejor atención.

Metodología

El análisis se realizó basándose en una combinación del método científico y la metodología CRISP-DM (Cross Industry Standard Process for Data Mining). Tomando como base 5 pasos: (1) Comprensión del negocio y de los datos, (2) Preparación de los datos, (3) Modelado, (4) Evaluación de efectividad del modelo, (5) Despliegue (teórico). Se realizaron

7 iteraciones a la metodología, realizando cambios en cuanto a las variables utilizadas, los modelos utilizados y cambios en las implementaciones de los modelos.

Materiales Utilizados

Se utilizó el Data set dado por el instructor, llamado “medicaldata”, el análisis y modelado se hizo utilizando un Jupyter Notebooks en Visual Studio Code con código python y con las siguientes librerías Pandas, Time, Matplotlib, Seaborn y Sklearn; Se utilizaron las siguientes medidas para verificar la precisión y el rendimiento de los modelos: Accuracy, Recall, F1-score, AUC-ROC, Elapsed Time

Primera Iteración

Se realizó una categorización de la variable “FRAUD_LABEL”, la cual contenía valores booleanos, reemplazándolos por 0 (false) y 1 (true), el análisis permitió entender la distribución de las distintas características, se encuentran la mitad de los datos con valor TRUE y la otra mitad FALSE. El comportamiento de muchas variables depende de si el registro es fraudulento o no, estas variables serán buenas para diseñar el modelo predictivo para detectar fraudes. Se realizaron 4 modelos: logistic regression model, decisión tree model, random forest model, Gradient boosting model.

El modelo con mayor Accuracy (0.85) y mayor AUC-ROC (0.9325) fue el Gradient Boosting, aunque el mismo fue el que tomo más tiempo para ser procesado 0.2 segundos, seguido por Random Forest muy de cerca con Accuracy de 0.825 y AUC-ROC de 0.92 y un tiempo de ejecución de 0.10 segundos. El modelo con el peor desempeño fue la regresión logística con 0.75 de Accuracy y 0.90 de AUC-ROC.

Imagen 1.

Valores de AUC-ROC para cada modelo utilizado.

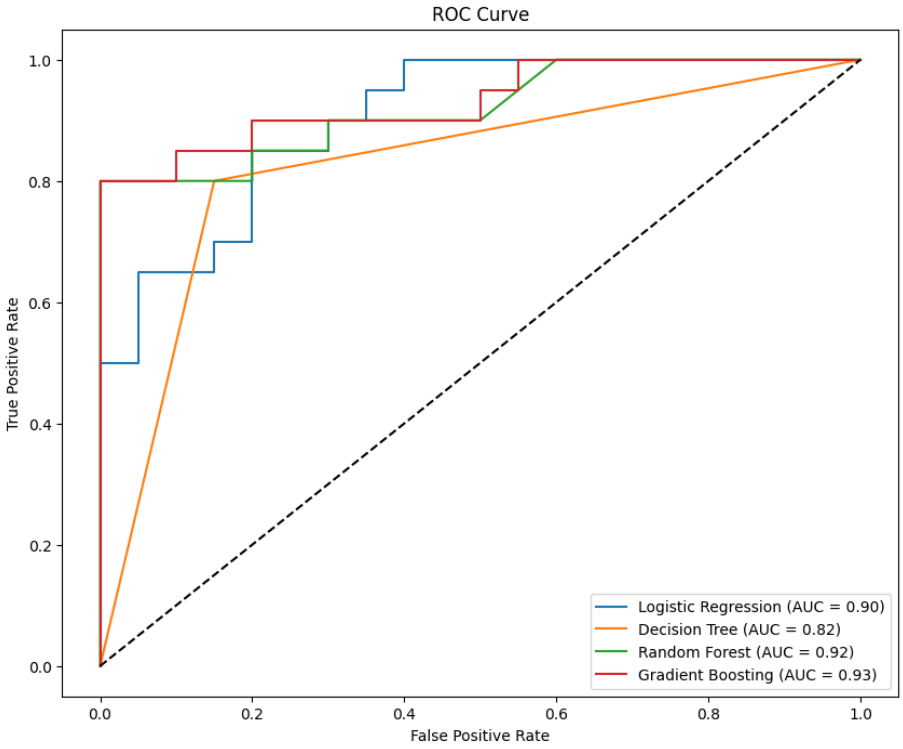


Imagen 2.

Valores de las métricas de evaluación para cada modelo utilizado.

Model	Accuracy	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	AUC-ROC
Logistic Regression	0.75	0.708333	0.85	0.772727	0.8125	0.65	0.722222	0.9025
Decision Tree	0.825	0.809524	0.85	0.829268	0.842105	0.8	0.820513	0.825
Random Forest	0.825	0.809524	0.85	0.829268	0.842105	0.8	0.820513	0.92
Gradient Boosting	0.85	0.818182	0.9	0.857143	0.888889	0.8	0.842105	0.9325

Segunda Iteración

Para el modelo de Decision Tree hay 5 variables las cuales tienen 0.0 de importance respecto a la variable objetivo (FRAUD_LABEL) por ello, se excluyeron; Se realizó con todos los modelos a fin de verificar si esta modificación significara alguna mejora en las medidas de rendimiento.

Imagen 3.

Importancia de las variables para el modelo Decision Tree.

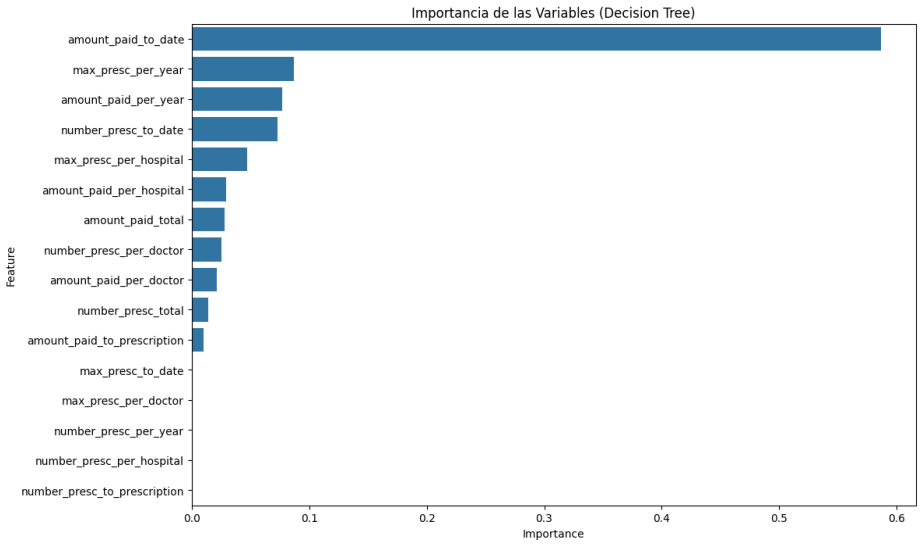


Imagen 4.

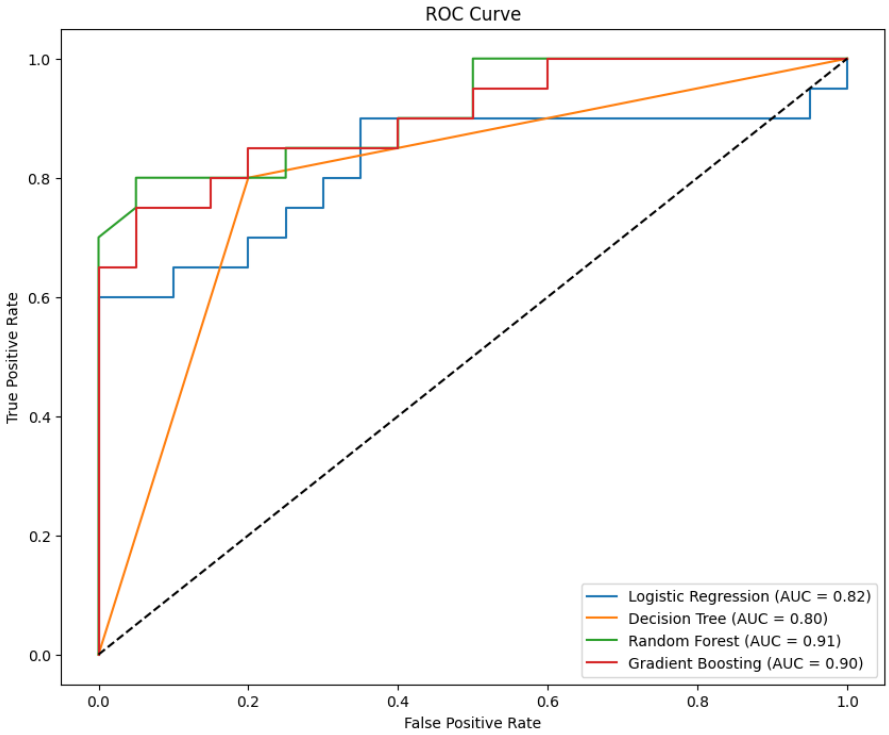
Valores de las métricas de evaluación para cada modelo utilizado.

Model	Accuracy	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	AUC-ROC
Logistic Regression	0.75	0.708333	0.85	0.772727	0.8125	0.65	0.722222	0.825
Decision Tree	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Random Forest	0.825	0.809524	0.85	0.829268	0.842105	0.8	0.820513	0.91375
Gradient Boosting	0.825	0.842105	0.8	0.820513	0.809524	0.85	0.829268	0.9025

La eliminación de las variables significo una disminución de AUC-ROC inclusive para el modelo de Árbol de decisión, pasando de 0.82 a 0.80.

Imagen 5.

Valores de AUC-ROC para cada modelo utilizado.



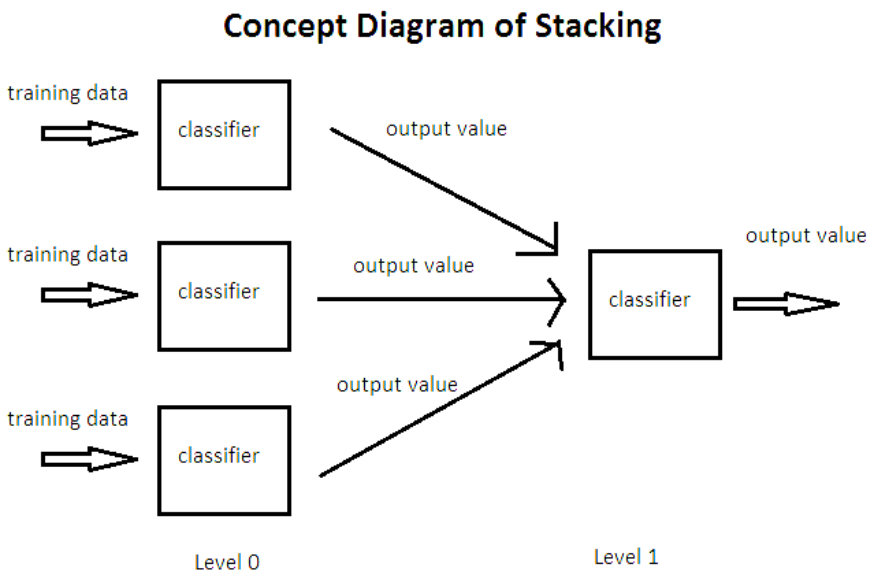
Tercera Iteración

Si bien se obtuvo una pequeña mejora en el rendimiento también se perdió precisión en los distintos modelos. Siendo el modelo Gradient Boosting (el modelo con mayor precisión en la primera iteración) el más afectado, pasando de un Accuracy de 0.85 a 0.825.

Se realizó una investigación sobre posibles modelos que pudieran obtener mejores resultados para el fin propuesto, se encontró el modelo Stacking Classifier, según Bantu, “El apilamiento es un paradigma diferente. El objetivo del apilamiento es explorar un espacio de diferentes modelos para el mismo problema. La idea es que se puede atacar un problema de aprendizaje con diferentes tipos de modelos que sean capaces de aprender una parte del problema, pero no todo el espacio del problema. Por lo tanto, puede crear varios alumnos diferentes y utilizarlos para crear una predicción intermedia, una predicción para cada modelo aprendido. Luego agrega un nuevo modelo que aprende de las predicciones intermedias para el mismo objetivo.” (BANTU, 2020)

Imagen 4.

Diagrama del modelo Stacking.



Se hicieron uso de 4 clasificadores en nivel 0: Logistic Regression, Decision Tree, Random Forest y Support Vector Machine y de clasificador de nivel 1: Histograma basado en Gradient Boosting Classifier, se obtuvo un Accuracy de 0.975.

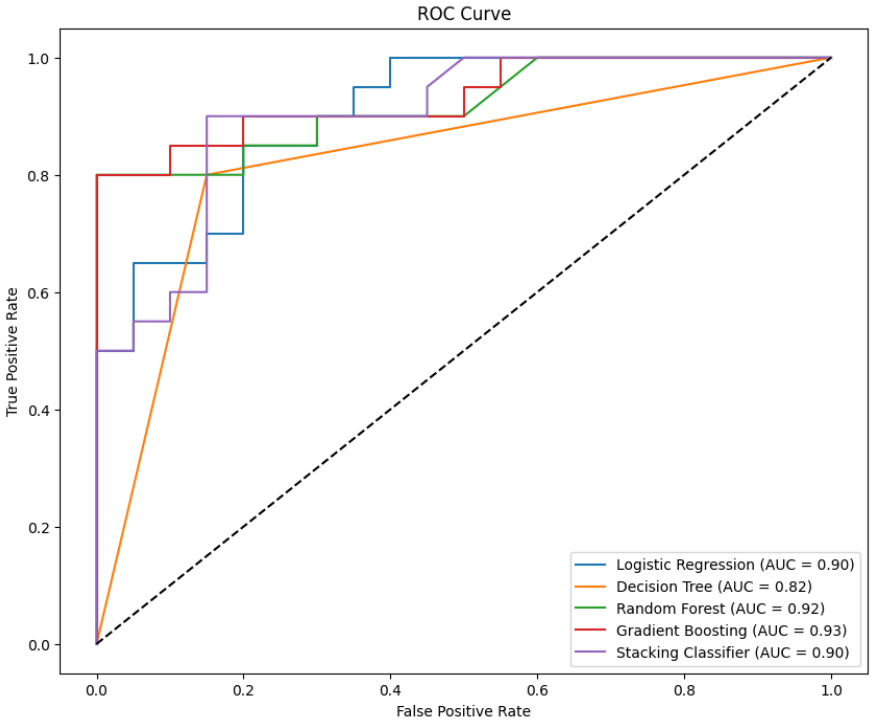
Imagen 6.

Valores de las métricas de evaluación para cada modelo utilizado.

Model	Accuracy	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	AUC-ROC
Logistic Regression	0.75	0.708333	0.85	0.772727	0.8125	0.65	0.722222	0.825
Decision Tree	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Random Forest	0.825	0.809524	0.85	0.829268	0.842105	0.8	0.820513	0.91375
Gradient Boosting	0.825	0.842105	0.8	0.820513	0.809524	0.85	0.829268	0.9025
Stacking Classifier	0.825	0.809524	0.85	0.829268	0.842105	0.8	0.820513	0.89875

Imagen 7.

Valores de AUC-ROC para cada modelo utilizado.



Cuarta Iteración

Se realizó una investigación sobre el modelo Gradient Boosting y se encontró un modelo llamado “Stochastic Gradient Boosting”, la principal diferencia entre Gradient Boosting y su versión estocástica, es que esta última utiliza muestreo aleatorio de observaciones de entrenamiento. “en cada iteración del algoritmo, el weak learner se ajusta empleando únicamente una fracción del set de entrenamiento, extraída de forma aleatoria y sin reemplazo (no con bootstrapping). Al resultado de esta modificación se le conoce como Stochastic Gradient Boosting” (Rodrigo, 2017). Se utilizó Stacking Classifier de base y se combinó con SGB; Este nuevo modelo ofreció un Accuracy de 0.925.

Quinta Iteración

Con la finalidad de aumentar la precisión del modelo Stacking Classifier, se realizó un Hyperparameter Tuning, utilizando “Grid parameters”, “El ajuste de hiperparámetros es el proceso de seleccionar los valores óptimos para los hiperparámetros de un modelo de aprendizaje automático. Los hiperparámetros son configuraciones que controlan el proceso de aprendizaje del modelo, como la tasa de aprendizaje, la cantidad de

neuronas en una red neuronal o el tamaño del kernel en una máquina de vectores de soporte. El objetivo del ajuste de hiperparámetros es encontrar los valores que conducen al mejor rendimiento en una tarea determinada.” (GeekForGeeks, 2023)

El modelo obtuvo valores “perfectos”, lo que indica que hubo un sobre-entrenamiento del mismo.

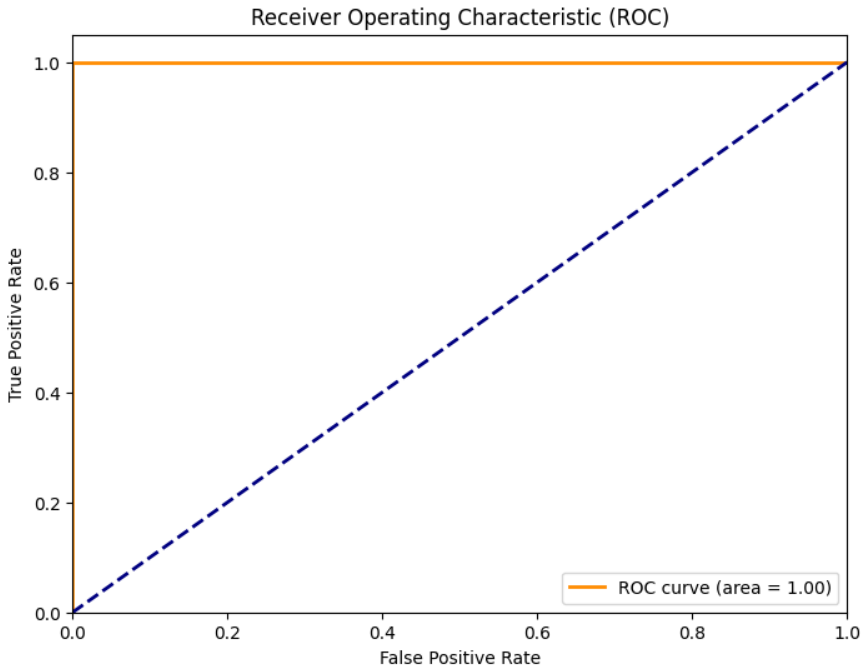
Imagen 8.

Valores de las métricas para Stacking Classifier haciendo Hyperparameter Tuning.

Metric	Value
Accuracy	1
Precision (Class 0)	1
Recall (Class 0)	1
F1-Score (Class 0)	1
Precision (Class 1)	1
Recall (Class 1)	1
F1-Score (Class 1)	1
AUC-ROC	1

Imagen 9.

Valores de AUC-ROC para Stacking Classifier haciendo Hyperparameter Tuning.



Sexta Iteración

Se le agrego al modelo de Stacking Classifier regularización, “La regularización es una técnica utilizada en machine learning para evitar el sobreajuste (overfitting) de los modelos. El sobreajuste ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento y pierde la capacidad de generalizar para nuevos datos. Regularizar los modelos nos ayuda a reducir la complejidad del modelo y a evitar el sobreajuste.” (Alvaro, s.f.)

Imagen 10.

Valores de métricas para el modelo Stacking Classifier haciendo uso de regularización.

Metric	Value
Accuracy	0.975
Precision (Class 0)	0.9545
Recall (Class 0)	1
F1-Score (Class 0)	0.9767
Precision (Class 1)	1
Recall (Class 1)	0.9474
F1-Score (Class 1)	0.973
AUC-ROC	1

Séptima Iteración

Dado que en la sexta iteración, aun aplicado regularizacion, los resultados de las métricas eran extremadamente altos (lo que indica un sobre-entrenamiento) se decidió retirar del Stacking el modelo SVM (support vector machine), dado que este modelo tiende a sobre-ajustarse en Dataset pequeños, como el utilizado para este trabajo, de únicamente 200 registros. Además se agregaron penalidades, con el mismo objetivo de evitar el sobre entrenamiento, como también se agregó una validación cruzada con el mismo fin.

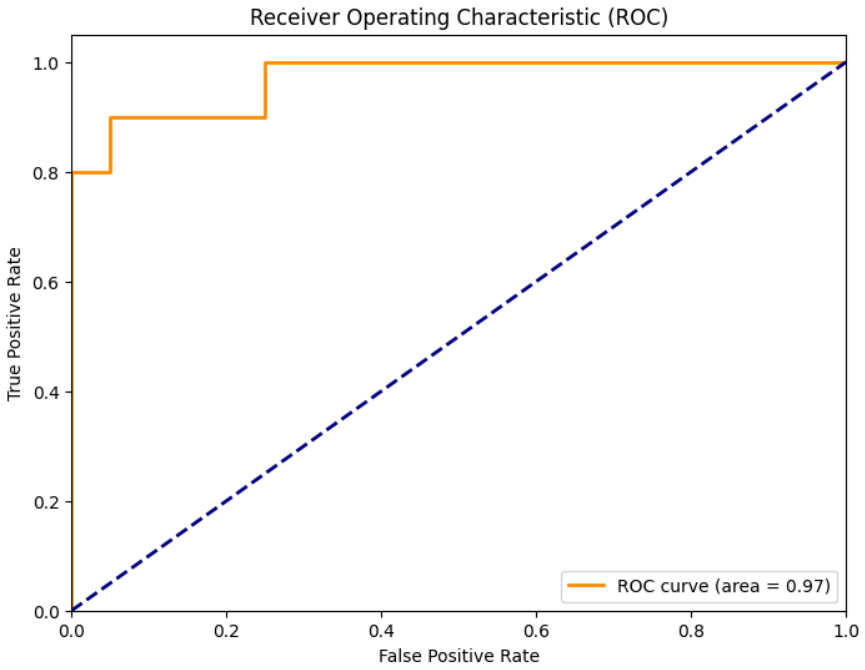
Imagen 8.

Valores de métricas para el modelo Stacking Classifier sin SVM, haciendo uso de regularización, aplicando penalidades y realizando validaciones cruzadas

Metric	Value
Mean Roc_auc	0.9819
Mean Accuracy	0.9542
Mean Precision	0.9478
Mean Recall	0.9667
Mean F1	0.9553
Final Test Accuracy	0.875
Final Test AUC-ROC	0.97
Final Test Precision (Class 0)	0.89
Final Test Recall (Class 0)	0.85
Final Test F1-Score (Class 0)	0.87
Final Test Precision (Class 1)	0.86
Final Test Recall (Class 1)	0.9
Final Test F1-Score (Class 1)	0.88

Imagen 9.

Gráfico de la AUC-ROC para el modelo Stacking Classifier sin SVM, haciendo uso de regularización, aplicando penalidades y realizando validaciones cruzadas.



Para realizar el despliegue se debe preparar un servidor (configurar), crear una API que utilice el modelo y permita realizar peticiones al mismo con los datos médicos, gestionar el modelo, es decir, optimizarlo, monitorear su rendimiento a lo largo de los usos y facilitar actualizaciones. También documentar y capacitar al personal que hará uso del modelo desarrollado.

Conclusiones Finales

Se ha realizado una aplicación de CRIPS-DM, utilizando los datos provenientes del data set enviado por el instructor, se realizaron siete iteraciones, con las cuales se pudo contrastar mejoras en algunos aspectos y algunas desventajas (como el tiempo de entrenamiento). Se han analizado los resultados obtenidos, basándose en las medidas: Accuracy, Recall, F1 score, AUC-ROC, como también el tiempo que el modelo tarda en

ejecutarse. Esto permitió poder llegar a la conclusión de que el mejor modelo para los datos es Stacking Classifier, con los modelos base: Regresión logística, Decision Tree y Random Forest y como estimador final Gradient Boosting basado en Historigramas, a su vez, el modelo utiliza regularización, validaciones cruzadas y penalidades para evitar el sobre entrenamiento. Este modelo obtuvo un Accuracy de 0.875 y un AUC-ROC de 0.97.

Como posible mejora a futuro se podría aumentar la cantidad de registros utilizados para el entrenamiento del modelo, dado que el Dataset es “pequeño” (200 registros), el sobre-entrenamiento es difícil de evitar.

Referencias

Alvaro. (s.f.). *machinelearningparatodos*. Obtenido de https://machinelearningparatodos.com/que-es-la-regularizacion-en-machine-learning-ejemplo-con-python/

BANTU, A. (2020). *Kaggle*. Obtenido de https://www.kaggle.com/code/anuragbantu/stacking-ensemble-learning-beginner-s-guide

España, Gob. (26 de 01 de 2021). *datos.gob.es*. Obtenido de https://datos.gob.es/es/blog/como-se-si-mi-modelo-de-prediccion-es-realmente-bueno#:~:text=Exactitud%20o%20accuracy%3A%20la%20fracci%C3%B3n,valor%20entre%200%20y%201.

GeekForGeeks. (07 de 12 de 2023). *geekforgeeks*. Obtenido de https://www.geeksforgeeks.org/hyperparameter-tuning/

issa. (4 de 6 de 2022). *issa*. Obtenido de https://www.issa.int/es/analysis/detecting-fraud-health-care-through-emerging-technologies

Mordor intelligence. (2024). *mordorintelligence*. Obtenido de https://www.mordorintelligence.com/es/industry-reports/healthcare-fraud-detection-market

Rodrigo, J. A. (Febrero de 2017). *rpubs*. Obtenido de https://rpubs.com/Joaquin_AR/255596