

ENTRENAMIENTO Y EVALUACIÓN DE MODELOS

Ricardo Osorio Castro
Wilmer Mario Leiva Esteban
Universidad de Antioquia, Colombia

Resumen: En el presente proyecto, presentamos el diseño, análisis y simulación de un sistema de predicción basado en técnicas de aprendizaje de máquina, describiendo el problema y su contexto en términos del estado del arte, y especificando cada una de las etapas del desarrollo del trabajo, los modelos con sus respectivas restricciones, la metodología de validación, los resultados de las simulaciones y las conclusiones obtenidas.

Índice de términos: IEEE 2937-2022 Estándar IEEE para evaluación comparativa de rendimiento para sistemas de servidores de inteligencia artificial

I. INTRODUCCIÓN

En el contexto del análisis y aplicación de técnicas de aprendizaje automático, el presente proyecto se centra en abordar un problema específico de clasificación. El objetivo principal es desarrollar modelos predictivos capaces de clasificar datos de manera efectiva y precisa en categorías predefinidas. Para ello, se utilizarán diversas técnicas y algoritmos de aprendizaje automático, incluyendo Random Forest, Gradient Boosted Trees y Decision Trees Classifier.

El proyecto se estructura en torno a la experimentación con diferentes modelos de aprendizaje automático y el análisis de los resultados obtenidos. Se explorarán diversas métricas de evaluación para determinar la eficacia y el rendimiento de los modelos

II. METODOLOGÍA

A. Base de Datos

En esta ocasión, estaremos trabajando con un problema de clasificación, el cual tiene como título: "ICR - Identifying Age-Related Conditions", el cual viene de una

competencia en Kaggle a la cual puede acceder desde <https://www.kaggle.com/competitions/icr-identify-age-related-conditions/overview>

Los datos comprenden más de cincuenta características de salud anónimas vinculadas a tres afecciones relacionadas con la edad. El objetivo es predecir si un sujeto ha sido diagnosticado(1) o no(0) con una de estas condiciones.

La base de datos preprocesada puede ser descargada desde el siguiente enlace: <https://drive.google.com/file/d/1iLscCDL4PIE4Kiym-BcWa9LDPPcU96qV/view?usp=sharing>

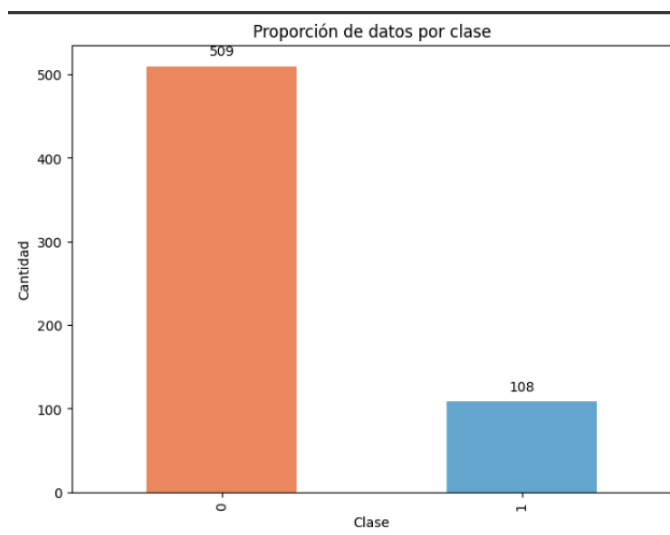


Figura 1. Distribución de clases

Como se puede observar en la figura 1, las clases presentan un claro desbalance. Para solucionar esto, se decidió utilizar la técnica de sobremuestreo. La razón de esto, es que al momento de realizar pruebas con un modelo (Random Forest), se pudo determinar una mayor

precisión con la base de datos balanceada con sobremuestreo que con submuestreo.

Dado esto, cada clase queda con 509 datos.

Esta base de datos balanceada se puede descargar desde el siguiente enlace:

<https://drive.google.com/file/d/1ZfDISHifVL4P42ZBD9uC1BI82YxvFqO4/view?usp=sharing>

Para la validación de cada modelo, se dividirán los datos en prueba y test, con ayuda de la librería de sklearn.model_selection: train_test_split

Puede acceder al repositorio del proyecto a través del siguiente link en Github:

<https://github.com/Dmolight01/ProyectoModelosII.git>

B. Métricas de Evaluación

Para evaluar el desempeño de nuestros modelos de predicción en el problema de clasificación médica, utilizaremos dos medidas clave: precisión (accuracy) y sensibilidad (recall).

- **Accuracy:** Es la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones realizadas. En el contexto de nuestro problema, la precisión nos dará una idea de la capacidad general del modelo para predecir correctamente tanto los casos positivos como los negativos..
- **Recall:** Es la proporción de casos positivos que fueron correctamente identificados por el modelo. En el contexto médico, la sensibilidad es una métrica crucial, ya que nos indica la capacidad del modelo para detectar correctamente los casos de enfermedad.
- **Precision:** Es la proporción de predicciones positivas correctas (verdaderos positivos) sobre el total de predicciones positivas realizadas.

III. ANÁLISIS Y RESULTADOS

Para comenzar, debemos escoger con qué modelos vamos a experimentar. Para esta selección, utilizamos la librería Pycaret, la cual permite entrenar los datos con

varios modelos y de esta manera seleccionar los que se vean más prometedores, basados en las métricas de evaluación escogidas.

También hicimos la prueba con los datos balanceados y sin balancear. Estos resultados se pueden ver en la figura 2.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.9808	0.9992	0.9703	0.9808	0.9808	0.9816	0.9821	
rf	Random Forest Classifier	0.9734	0.9977	0.9852	0.9623	0.9735	0.9468	0.9473	
xgboost	Extreme Gradient Boosting	0.9675	0.9952	0.9881	0.9491	0.9679	0.9350	0.9364	
gbc	Gradient Boosting Classifier	0.9660	0.9930	0.9818	0.9412	0.9667	0.9320	0.9338	
lightgbm	Light Gradient Boosting Machine	0.9660	0.9974	0.9881	0.9485	0.9666	0.9321	0.9336	
ada	Ada Boost Classifier	0.9349	0.9662	0.9732	0.9048	0.9370	0.8699	0.8740	
dt	Decision Tree Classifier	0.9244	0.9251	0.9672	0.8933	0.9275	0.8491	0.8548	
ridge	Ridge Classifier	0.8920	0.9261	0.8806	0.9027	0.8899	0.7840	0.7870	
lda	Linear Discriminant Analysis	0.8787	0.9328	0.8716	0.8844	0.8771	0.7574	0.7587	
qda	Quadratic Discriminant Analysis	0.8092	0.9508	0.6596	0.9370	0.7711	0.6172	0.6471	
lr	Logistic Regression	0.8078	0.8880	0.7701	0.8313	0.7986	0.6153	0.6182	
knn	K Neighbors Classifier	0.7384	0.8156	0.8127	0.7061	0.7539	0.4773	0.4862	
nb	Naive Bayes	0.6835	0.8619	0.4242	0.8678	0.5644	0.3642	0.4223	
svm	SVM - Linear Kernel	0.6391	0.6946	0.6284	0.6737	0.6065	0.2759	0.3072	
dummy	Dummy Classifier	0.5045	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.8463	0.7724	0.3696	0.6533	0.4863	0.3712	0.4032	
ridge	Ridge Classifier	0.8463	0.8249	0.2286	0.6533	0.3133	0.2612	0.3220	
rf	Random Forest Classifier	0.8317	0.9180	0.0286	0.2000	0.0500	0.0433	0.0697	
knn	K Neighbors Classifier	0.8293	0.6546	0.1839	0.5633	0.2520	0.1931	0.2432	
et	Extra Trees Classifier	0.8293	0.8546	0.0143	0.1000	0.0250	0.0217	0.0348	
nb	Naive Bayes	0.8268	0.7913	0.2661	0.6250	0.3425	0.2596	0.3046	
dt	Decision Tree Classifier	0.8268	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	
qda	Quadratic Discriminant Analysis	0.8268	0.4840	0.0000	0.0000	0.0000	0.0000	0.0000	
ada	Ada Boost Classifier	0.8268	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	
gbc	Gradient Boosting Classifier	0.8268	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	
lda	Linear Discriminant Analysis	0.8268	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	
xgboost	Extreme Gradient Boosting	0.8268	0.6257	0.0000	0.0000	0.0000	0.0000	0.0000	
lightgbm	Light Gradient Boosting Machine	0.8268	0.7245	0.0000	0.0000	0.0000	0.0000	0.0000	
dummy	Dummy Classifier	0.8268	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	

Figura 2. Tabla de métricas de dos archivos de datos.

Cómo se puede ver, los datos balanceados ofrecieron mejores resultados en la mayoría de las métricas que el del dataset sin hacer el Oversampling, por esto, entrenaremos los modelos con los datos balanceados.

A. Random Forest (RF)

El primer modelo a entrenar es Random Forest. Para nuestros experimentos vamos a configurar el RF para que el mínimo de muestras para considerar un nodo sea de 3.

La figura 3 muestra el comportamiento del modelo con diferentes número de árboles.

número de árboles	variables para la selección del mejor umbral	eficiencia de entrenamiento	std(train)	eficiencia de prueba	std(test)
0	5.0	2.0	0.988771	0.0	0.980392
1	5.0	5.0	1.000000	0.0	0.960784
2	5.0	10.0	0.988771	0.0	0.975588
3	5.0	52.0	0.995096	0.0	0.965686
4	20.0	2.0	1.000000	0.0	0.995098
5	20.0	5.0	1.000000	0.0	0.985294
6	20.0	10.0	1.000000	0.0	0.975490
7	20.0	52.0	1.000000	0.0	0.965686
8	50.0	2.0	1.000000	0.0	0.995098
9	50.0	5.0	1.000000	0.0	0.995098
10	50.0	10.0	1.000000	0.0	0.975490
11	50.0	52.0	1.000000	0.0	0.965686
12	100.0	2.0	1.000000	0.0	0.995098
13	100.0	5.0	1.000000	0.0	0.990196
14	100.0	10.0	1.000000	0.0	0.975490
15	100.0	52.0	1.000000	0.0	0.960784
16	150.0	2.0	1.000000	0.0	1.000000
17	150.0	5.0	1.000000	0.0	0.985294
18	150.0	10.0	1.000000	0.0	0.975490
19	150.0	52.0	1.000000	0.0	0.960784

Figura 3. Comportamiento de RF

Los resultados son bastante impresionantes. En general, la eficiencia de entrenamiento es excepcionalmente alta, alcanzando el 100% en muchos casos, lo que sugiere que el modelo Random Forest está aprendiendo perfectamente los datos de entrenamiento.

Además, al mirar la eficiencia de prueba, que es una mejor medida del rendimiento del modelo en datos no vistos, vemos que también es bastante alta, aunque no alcanza el 100% en la mayoría de los casos. Esto es esperado y refleja una buena capacidad de generalización del modelo.

Ahora pasamos a entrenar el modelo con los mejores hiperparámetros obtenidos, los cuales fueron:

-Mejor número de árboles: 150

-Mejor número de variables: 2

Además calculamos las métricas de evaluación (accuracy, recall y precision) en el conjunto de prueba.

Modelo	Accuracy	Recall	Precision
RF	1.0	1.0	1.0

Tabla 1. Métricas de Evaluación para RF

Hay que considerar la posibilidad de sobreajuste del modelo, especialmente cuando se obtiene un rendimiento perfecto en los datos de prueba. Para realizar una evaluación más exhaustiva, utilizamos validación cruzada, para confirmar la generalización del modelo a datos no vistos.

Para este caso hicimos validación cruzada de 5 folds.

```
Puntajes de validación cruzada: [1. 0.98039216 0.98039216 0.99014778 0.98522167]
Precisión media: 0.9872307543707137
Desviación estándar de la precisión: 0.007336646694444087
```

Figura 4. Validación Cruzada para RF

Los resultados de la validación cruzada muestran que el modelo tiene una precisión media del 98.72% con una desviación estándar de aproximadamente 0.73%. Esto indica que el modelo es muy consistente en su rendimiento en diferentes particiones de los datos durante la validación cruzada, lo que sugiere una alta capacidad de generalización.

En resumen, estos resultados respaldan la eficacia del modelo en la tarea de clasificación, ya que logra una alta precisión en la validación cruzada y muestra una baja variabilidad en su rendimiento en diferentes conjuntos de datos.

B. Gradient Boosted Trees(GBT)

El segundo modelo a entrenar es Gradient Boosted Trees. Para nuestros experimentos configuramos el árbol con un mínimo de tres (3) muestras para considerar una división de un nodo.

La figura 5 muestra el comportamiento del modelo con diferentes número de árboles.

número de árboles	eficiencia de entrenamiento	std(train)	eficiencia de prueba	std(test)
0	5.0	0.902948	0.0	0.852941
1	10.0	0.923833	0.0	0.867647
2	20.0	0.961916	0.0	0.901961
3	50.0	0.996314	0.0	0.950980
4	100.0	1.000000	0.0	0.965686

Figura 5. Comportamiento de GBT

Podemos observar que a medida que aumenta el número de árboles, tanto la eficiencia de entrenamiento como la de prueba tienden a mejorar. Esto sugiere que el modelo se beneficia de una mayor complejidad y generaliza mejor a medida que se agregan más árboles.

Ahora pasamos a entrenar el modelo con los mejores hiperparámetros obtenidos, los cuales fueron:

-Mejor número de árboles: 150

-Mejor número de variables: 2

Además calculamos las métricas de evaluación (accuracy, recall y precision) en el conjunto de prueba.

Modelo	Accuracy	Recall	Precision
GBT	0.95098	1.0	0.90196

Tabla 2. Métricas de Evaluación para GBT

Hacemos validación cruzada, para confirmar la generalización del modelo a datos no vistos. Para este caso hicimos validación cruzada de 5 folds.

```
Puntajes de validación cruzada: [0.98529412 0.95588235 0.97058824 0.97536946 0.95566502]
Precisión media: 0.9685598377281949
Desviación estándar de la precisión: 0.011467486877492699
```

Figura 6. Validación Cruzada para GBT

Los resultados de la validación cruzada muestran que el modelo GBT tiene una precisión media de aproximadamente 0.969, con una desviación estándar de alrededor de 0.011. Esto indica que el modelo tiene un rendimiento generalmente consistente en diferentes conjuntos de datos de validación, ya que la desviación estándar es relativamente baja en comparación con la precisión media.

El modelo parece ser una buena opción para este problema de clasificación, ya que muestra un buen rendimiento en términos de precisión tanto en el conjunto de prueba como en la validación cruzada.

C. Decision Tree Classifier(DTC)

El tercer modelo a entrenar es Decision Tree Classifier. Para nuestros experimentos configuramos el árbol con la medida de impureza Entropía.

La figura 7 muestra el comportamiento del modelo con diferentes números de profundidades.

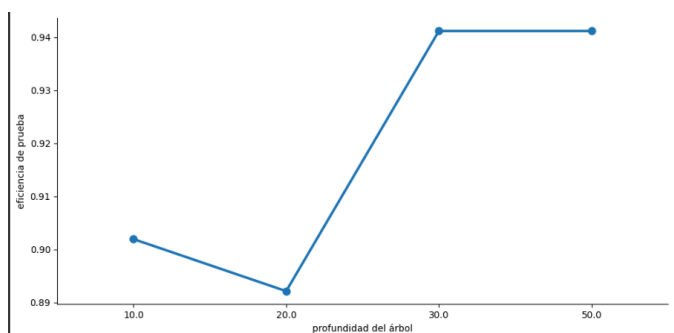


Figura 7. Comportamiento de DTC

Se puede observar que el modelo no presenta un comportamiento claramente definido, ya que mientras aumenta la profundidad del árbol, la eficiencia de prueba puede crecer cómo decrecer.

En este caso seleccionamos cómo mejor profundidad 30, y haciendo un “podado” con un “Cost complexity pruning” o ccp_alpha de 0.2, el resultado se puede ver en la figura 8.

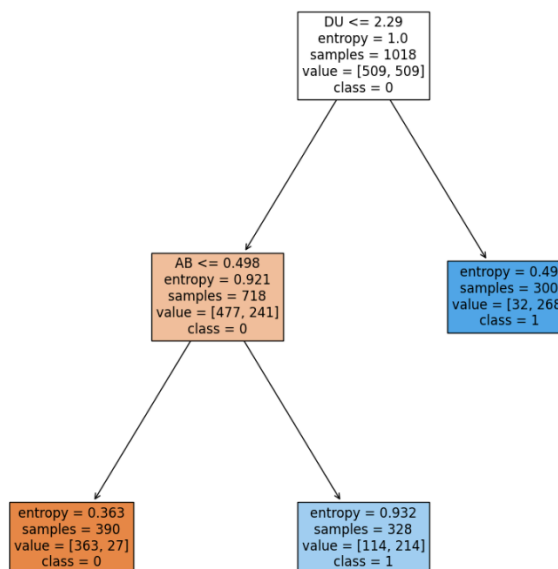


Figura 8. Árbol “podado” de DTC

Los resultados del entrenamiento son:

Modelo	Accuracy	Recall	Precision
DTC	0.95098	1.0	0.90196

Tabla 3. Métricas de Evaluación para DTC

Hacemos validación cruzada, para confirmar la generalización del modelo a datos no vistos.

Para este caso hicimos validación cruzada de 5 folds.

```
Puntajes de validación cruzada: [0.94607843 0.90196078 0.96078431 0.95073892 0.9408867 ]
Precisión media: 0.9400898290350623
Desviación estándar de la precisión: 0.020159301412066684
```

Figura 9. Validación Cruzada para DTC

Los resultados de la validación cruzada muestran que el modelo de DTC tiene una precisión media de aproximadamente 0.94 con una desviación estándar de alrededor de 0.02. Esto indica que el rendimiento del modelo es bastante consistente en diferentes divisiones de los datos. Sin embargo, la precisión puede variar ligeramente entre las diferentes divisiones.

En general, el modelo puede ser bastante efectivo para este problema,

IV. CONCLUSIONES

-Después de entrenar y evaluar tres modelos diferentes (Random Forest, Gradient Boosted Trees y Decision Trees Classifier), se observó que todos lograron un alto rendimiento en términos de precisión y otras métricas de evaluación. Sin embargo, Random Forest mostró un desempeño fenomenal, superando a sus dos contrincantes. Es muy probable que en el futuro próximo, este modelo nos sea de gran utilidad.

-Se puede observar la importancia de los hiperparámetros, ya que afectaron significativamente el rendimiento de los modelos. Si bien, en los diferentes modelos entrenamos con el que para nosotros era el mejor, creemos que este apartado tiene oportunidad de mejora, ya que aun queda por explorarlos más a fondo, para obtener el que de verdad sea el mejor para cada modelo.

-El caso del entrenamiento del modelo Decision Trees Classifier fue en particular el más desafiante en nuestra perspectiva, ya que, a diferencia de los otros modelos, donde se utilizan múltiples árboles y se promedian los resultados, en este se construye un solo árbol basado en los datos de entrenamiento. La aleatoriedad en la construcción del árbol, como la selección de características o el orden de las muestras, lleva a resultados ligeramente diferentes en cada ejecución.

-Si bien no es una conclusión del proyecto en sí, sino más bien del proceso de construcción del proyecto, pudimos observar que hay muchas herramientas que

facilitan el proceso de entrenamiento de IA, cómo lo fue en este caso la librería Pycaret. Pero para sacarle todo el potencial a estas herramientas hay que tener claro lo que se busca y cómo interpretar los hallazgos que nos ofrecen.

RECONOCIMIENTOS

Se hace un agradecimiento a la profesora María Bernarda Salazar Sanchez, ya que nos guió para el desarrollo correcto del presente trabajo y siempre estuvo dispuesta a solucionar dudas cuando lo requerimos.

CIBERGRAFÍA

-Salazar, M. (2024). Modelos y Simulación de Sistemas II [Classroom]. Universidad de Antioquia. URL: <https://classroom.google.com/c/NjYxMTA3NzEyMzYx>

-Emilio. (2022). PyCaret: Paso a paso. TodoBI. Recuperado de <https://todobi.com/pycaret-paso-a-paso/>

-Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

-"Acerca de los estándares de inteligencia artificial" (s.f.). Recuperado de <https://www.antpedia.com/standard/sp/es/1023548.html>

--"Formato IEEE para presentar artículos" (s.f.). Recuperado de https://revistas.pucp.edu.pe/imagenes/electro/ee_formato_ieee.pdf