# ENTRENAMIENTO Y EVALUACIÓN DE MODELOS DE CLASIFICACIÓN ICR

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

**Resumen:** En el actual proyecto, se presentan 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:** Aprendizaje automático, predicción, Machine Learning, datos, interpretación de resultados, medicina, falsos positivos, métricas

## I. INTRODUCCIÓN

En el contexto del análisis y aplicación de técnicas de aprendizaje automático, el presente proyecto se centra en desarrollar modelos predictivos capaces de clasificar sujetos en base a sus características de salud y comprobar si son pertenecientes o no a una clase que cumple con ciertas condiciones médicas. Para ello, se utilizarán diversas técnicas y algoritmos de aprendizaje automático, incluyendo Random Forest, Gradient Boosted Trees y Decision Trees Classifier.

En está ocasión, se trabajará con un problema de clasificación, el cual tiene cómo título: “ICR - Identifying Age-Related Conditions”, el cual viene de una competencia en la plataforma Kaggle [1]. Se estructurará 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

Los datos suministrados por la competencia comprenden más de cincuenta características diferentes de salud anónimas vinculadas a tres condiciones médicas principales, relacionadas con la edad. El objetivo es predecir si un sujeto ha sido diagnosticado(1) o no(0) con una de estas tres condiciones que se encuentran agrupadas en una sola característica ubicada en la base de datos como *Class*. (Ver Figura 1).


**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.

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

### B. Métricas de Evaluación

Para evaluar el desempeño de los modelos de predicción

en el problema de clasificación médica se utilizaran dos medidas clave: precisión (accuracy) y sensibilidad (recall) [2].

- **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, se debe escoger con qué modelos se va a experimentar. Para está selección, se utilizará 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 se realizó 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 |
|---|---|---|---|---|---|---|---|---|
| et | Extra Trees Classifier | 0.9657 | 0.9860 | 0.9517 | 0.9826 | 0.9669 | 0.9314 | 0.9324 |
| rf | Random Forest Classifier | 0.9596 | 0.9913 | 0.9759 | 0.9485 | 0.9609 | 0.9192 | 0.9218 |
| xgboost | Extreme Gradient Boosting | 0.9508 | 0.9868 | 0.9690 | 0.9381 | 0.9523 | 0.9016 | 0.9040 |
| lightgbm | Light Gradient Boosting Machine | 0.9508 | 0.9898 | 0.9690 | 0.9391 | 0.9525 | 0.9015 | 0.9047 |
| gbc | Gradient Boosting Classifier | 0.9420 | 0.9839 | 0.9690 | 0.9227 | 0.9442 | 0.8839 | 0.8874 |
| ada | Ada Boost Classifier | 0.9385 | 0.9725 | 0.9655 | 0.9210 | 0.9411 | 0.8769 | 0.8812 |
| dt | Decision Tree Classifier | 0.9086 | 0.9078 | 0.9585 | 0.8754 | 0.9140 | 0.8170 | 0.8234 |
| ridge | Ridge Classifier | 0.8717 | 0.9197 | 0.8443 | 0.8989 | 0.8686 | 0.7436 | 0.7483 |
| lda | Linear Discriminant Analysis | 0.8682 | 0.9235 | 0.8514 | 0.8864 | 0.8665 | 0.7366 | 0.7405 |
| lr | Logistic Regression | 0.8226 | 0.8789 | 0.8134 | 0.8376 | 0.8228 | 0.6452 | 0.6494 |
| qda | Quadratic Discriminant Analysis | 0.7716 | 0.9260 | 0.6026 | 0.9246 | 0.7240 | 0.5460 | 0.5856 |
| knn | K Neighbors Classifier | 0.7083 | 0.7793 | 0.7612 | 0.6943 | 0.7252 | 0.4155 | 0.4195 |
| nb | Naive Bayes | 0.6731 | 0.8519 | 0.4081 | 0.8902 | 0.5511 | 0.3517 | 0.4199 |
| svm | SVM - Linear Kernel | 0.5816 | 0.6421 | 0.4621 | 0.5983 | 0.4568 | 0.1653 | 0.1789 |
| dummy | Dummy Classifier | 0.5079 | 0.5000 | 1.0000 | 0.5079 | 0.6736 | 0.0000 | 0.0000 |

```
best_model=compare_models()
```

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|---|
| nb | Naive Bayes | 0.8495 | 0.7676 | 0.3619 | 0.6679 | 0.4489 | 0.3735 | 0.4072 |
| lr | Logistic Regression | 0.8494 | 0.7440 | 0.4119 | 0.6262 | 0.4762 | 0.3989 | 0.4182 |
| ridge | Ridge Classifier | 0.8434 | 0.7859 | 0.1976 | 0.6167 | 0.2746 | 0.2298 | 0.2879 |
| rf | Random Forest Classifier | 0.8261 | 0.9200 | 0.0167 | 0.1000 | 0.0286 | 0.0248 | 0.0376 |
| dt | Decision Tree Classifier | 0.8232 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| qda | Quadratic Discriminant Analysis | 0.8232 | 0.4851 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| ada | Ada Boost Classifier | 0.8232 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| gbc | Gradient Boosting Classifier | 0.8232 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| lda | Linear Discriminant Analysis | 0.8232 | 0.5000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| et | Extra Trees Classifier | 0.8232 | 0.8910 | 0.0167 | 0.1000 | 0.0286 | 0.0196 | 0.0291 |
| xgboost | Extreme Gradient Boosting | 0.8232 | 0.7422 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| lightgbm | Light Gradient Boosting Machine | 0.8232 | 0.7305 | 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 los experimentos se va 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 | 0.998771 | 0.0 | 0.980392 | 0.0 |
| 1 | 5.0 | 5.0 | 1.000000 | 0.0 | 0.960784 | 0.0 |
| 2 | 5.0 | 10.0 | 0.998771 | 0.0 | 0.970588 | 0.0 |
| 3 | 5.0 | 52.0 | 0.995086 | 0.0 | 0.965686 | 0.0 |
| 4 | 20.0 | 2.0 | 1.000000 | 0.0 | 0.995098 | 0.0 |
| 5 | 20.0 | 5.0 | 1.000000 | 0.0 | 0.985294 | 0.0 |
| 6 | 20.0 | 10.0 | 1.000000 | 0.0 | 0.975490 | 0.0 |
| 7 | 20.0 | 52.0 | 1.000000 | 0.0 | 0.965686 | 0.0 |
| 8 | 50.0 | 2.0 | 1.000000 | 0.0 | 0.995098 | 0.0 |
| 9 | 50.0 | 5.0 | 1.000000 | 0.0 | 0.995098 | 0.0 |
| 10 | 50.0 | 10.0 | 1.000000 | 0.0 | 0.975490 | 0.0 |
| 11 | 50.0 | 52.0 | 1.000000 | 0.0 | 0.955882 | 0.0 |
| 12 | 100.0 | 2.0 | 1.000000 | 0.0 | 0.995098 | 0.0 |
| 13 | 100.0 | 5.0 | 1.000000 | 0.0 | 0.990196 | 0.0 |
| 14 | 100.0 | 10.0 | 1.000000 | 0.0 | 0.975490 | 0.0 |
| 15 | 100.0 | 52.0 | 1.000000 | 0.0 | 0.960784 | 0.0 |
| 16 | 150.0 | 2.0 | 1.000000 | 0.0 | 1.000000 | 0.0 |
| 17 | 150.0 | 5.0 | 1.000000 | 0.0 | 0.985294 | 0.0 |
| 18 | 150.0 | 10.0 | 1.000000 | 0.0 | 0.975490 | 0.0 |
| 19 | 150.0 | 52.0 | 1.000000 | 0.0 | 0.960784 | 0.0 |

**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, se observa 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, se empezará a entrenar el modelo con los mejores hiperparametros obtenidos, los cuales fueron:
-Mejor número de árboles: 20
-Mejor número de variables: 2

Además, se calcularon las métricas de evaluación (accuracy, recall y precision) en el conjunto de pruebas.
Accuracy: 0.995098
Recall: 1.0
Precisión: 0.990566

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 se utilizó el método de validación cruzada, para confirmar la generalización del modelo a datos no vistos.
Para el caso se tomó una 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.
La figura 5 muestra el comportamiento del modelo con diferentes número de árboles y diferente número mínimo de muestras para considerar una división de un nodo:

| | número de árboles | min_samples_split | eficiencia de entrenamiento | std(train) | eficiencia de prueba | std(test) |
|---|---|---|---|---|---|---|
| 0 | 5.0 | 3.0 | 0.902948 | 0.0 | 0.852941 | 0.0 |
| 1 | 5.0 | 4.0 | 0.902948 | 0.0 | 0.852941 | 0.0 |
| 2 | 5.0 | 5.0 | 0.902948 | 0.0 | 0.852941 | 0.0 |
| 3 | 5.0 | 6.0 | 0.902948 | 0.0 | 0.852941 | 0.0 |
| 4 | 5.0 | 7.0 | 0.902948 | 0.0 | 0.852941 | 0.0 |
| 5 | 10.0 | 3.0 | 0.926290 | 0.0 | 0.867647 | 0.0 |
| 6 | 10.0 | 4.0 | 0.923833 | 0.0 | 0.867647 | 0.0 |
| 7 | 10.0 | 5.0 | 0.923833 | 0.0 | 0.867647 | 0.0 |
| 8 | 10.0 | 6.0 | 0.923833 | 0.0 | 0.867647 | 0.0 |
| 9 | 10.0 | 7.0 | 0.923833 | 0.0 | 0.867647 | 0.0 |
| 10 | 20.0 | 3.0 | 0.961916 | 0.0 | 0.901961 | 0.0 |
| 11 | 20.0 | 4.0 | 0.952088 | 0.0 | 0.897059 | 0.0 |
| 12 | 20.0 | 5.0 | 0.954545 | 0.0 | 0.897059 | 0.0 |
| 13 | 20.0 | 6.0 | 0.959459 | 0.0 | 0.901961 | 0.0 |
| 14 | 20.0 | 7.0 | 0.958231 | 0.0 | 0.901961 | 0.0 |
| 15 | 50.0 | 3.0 | 0.995086 | 0.0 | 0.950980 | 0.0 |
| 16 | 50.0 | 4.0 | 0.993857 | 0.0 | 0.950980 | 0.0 |
| 17 | 50.0 | 5.0 | 0.995086 | 0.0 | 0.946078 | 0.0 |
| 18 | 50.0 | 6.0 | 0.995086 | 0.0 | 0.946078 | 0.0 |
| 19 | 50.0 | 7.0 | 0.995086 | 0.0 | 0.946078 | 0.0 |
| 20 | 100.0 | 3.0 | 1.000000 | 0.0 | 0.965686 | 0.0 |
| 21 | 100.0 | 4.0 | 1.000000 | 0.0 | 0.960784 | 0.0 |
| 22 | 100.0 | 5.0 | 1.000000 | 0.0 | 0.970588 | 0.0 |
| 23 | 100.0 | 6.0 | 1.000000 | 0.0 | 0.960784 | 0.0 |
| 24 | 100.0 | 7.0 | 1.000000 | 0.0 | 0.960784 | 0.0 |

**Figura 5**. Comportamiento de GBT

En base a la Figura 5 se logra 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, se procede a entrenar el modelo con los mejores hiperparametros obtenidos, los cuales fueron:
-Mejor número de árboles: 50
-Mejor número número mínimo de muestras: 3
Además, se calcularon las métricas de evaluación (accuracy, recall y precision) en el conjunto de prueba.
Accuracy: 0.95098
Recall: 0.98095
Precision:0.92793

Se realiza nuevamente la validación cruzada, para confirmar la generalización del modelo a datos no vistos.
Para este caso también se tomó una validación cruzada de 5 folds

```
Puntajes de validación cruzada: [0.97058824 0.94607843 0.96078431 0.97536946 0.94581281]
Precisión media: 0.9597266492804017
Desviación estándar de la precisión: 0.012195564833029556
```

**Figura 6**. Validación Cruzada para GBT

Los resultados de la validación cruzada muestran que el modelo GBT tiene una precisión media de aproximadamente 0.9597, con una desviación estándar de alrededor de 0.012. 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 los experimentos se realiza la configuración del árbol con la medida de impureza Entropía.
La figura 7 muestra el comportamiento del modelo con diferentes números de profundidades.

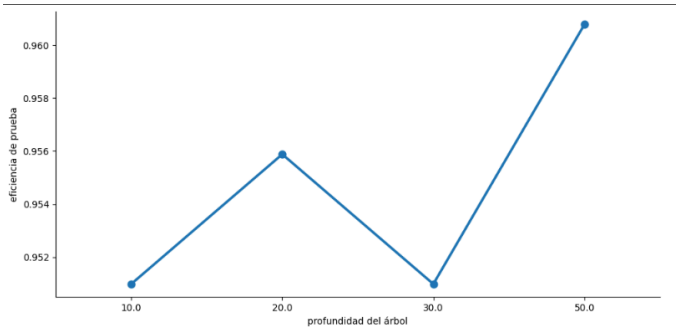


**Figura 7**. Comportamiento de DTC

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 queda considerada 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.

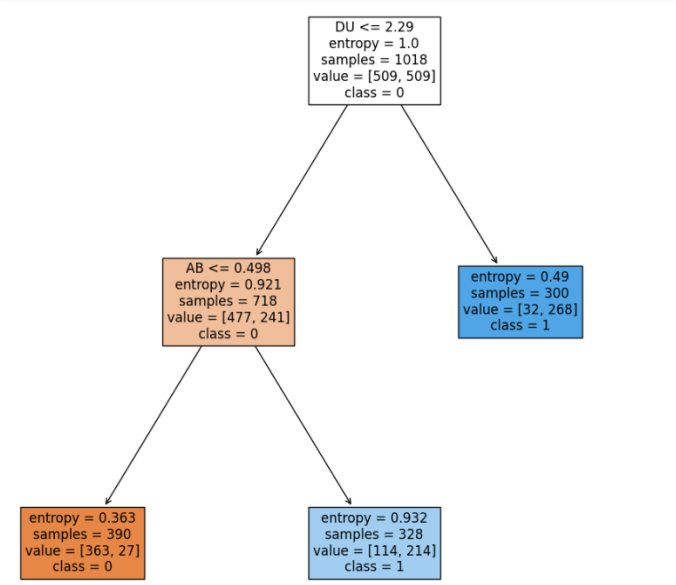


**Figura 8**. Árbol "podado" de DTC

Los resultados del entrenamiento son:
Accuracy: 0.94608
Recall: 1.0
Precision: 0.9052

Se continua con la validación cruzada, para confirmar la generalización del modelo a datos no vistos.
Y con el caso se toma una validación cruzada de 5 folds.

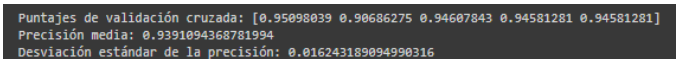


**Figura 9**. Validación Cruzada para DTC

Los resultados de la validación cruzada muestran que el modelo de DTC tiene una precisión media de aproximadamente 0.939 con una desviación estándar de alrededor de 0.016.. 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.

### Resultados

En la tabla 1 se puede ver el resumen de las métricas de evaluación en los modelos que fueron trabajados.

| Modelo | Accuracy | Recall | Precision |
|---|---|---|---|
| RF | 0.995098 | 1.0 | 0.990566 |
| GBT | 0.95098 | 0.98095 | 0.92793 |
| DTC | 0.94608 | 1.0 | 0.9052 |

**Tabla 1**. Métricas de Evaluación para los modelos

## *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 hiperparametros, 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

[1] Competencia de Kaggle(2024). ICR - Identifying Age-Related Conditions. URL: https://www.kaggle.com/competitions/icr-identify-age-related-conditions/overview

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

[2] Emilio. (2022). PyCaret: Paso a paso. TodoBI. Recuperado de https://todobi.com/pycaret-paso-a-paso/

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

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

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

- Se puede acceder al repositorio del proyecto a través del siguiente link en Github: https://github.com/Dmolight01/ProyectoModelosII.git