Modelo para la Predicción de Padecimiento de Diabetes

Ernesto Acosta Ruiz

A01364982

ABSTRACT – El principal objetivo de este documento es presentar la implementación de un modelo binario para predecir si un paciente padece diabetes dependiendo de varios datos del histórico de salud y niveles actuales de glucosa, peso y hemoglobina.

1. INTRODUCCIÓN

La diabetes es una enfermedad actualmente incurable. Es por eso por lo que es indispensable una detección temprana a partir de factores clínicos para priorizar estudios y dar tratamientos preventivos. La regresión logística es adecuada para este objetivo ya que esta nos entrega una respuesta binaria (tiene o no tiene diabetes).

2. DATASET

Este dataset incluye 8 “features” que su relevancia es alta a la hora de detectar si un paciente pudiera llegar a padecer diabetes en caso de que estas salieran de sus rangos normales.

- **Age:** La edad es un factor importante ya que, con el tiempo, las personas reducen la cantidad de actividad física, padecen de cambios hormonales y la alta posibilidad de desarrollar otros padecimientos que contribuyen a la diabetes. Se mide como un número entero.

- **Gender:** Los efectos de la diabetes pueden variar dependiendo del género. Algunos estudios han encontrado que los hombres tienen un riego mayor a padecer diabetes sobre las mujeres. Puede ser “Male”, “Female” o “Other”.

- **Body Mass Index (BMI):** El índice de masa corporal mide la cantidad de grasa en una persona a través de su peso y su altura. Las personas con un BMI alto tienen mayores riesgos de desarrollar diabetes tipo 2 ya que el exceso de grasa en la cintura genera resistencia a la insulina. Se mide como un número con decimales

- **Hypertension:** La hipertensión es una condición que normalmente coexiste con la diabetes. Ambas condiciones comparten factores de riesgo y contribuyen al desarrollo de amabas. Se toma en consideración si lo padece o no con un 0 o un 1.

- **Heart Disease:** De igual manera que la hipertensión, las enfermedades del corazón comparten factores de riesgo con la diabetes y contribuyen al desarrollo de ambos padecimientos. Se toma en consideración si lo padece o no con un 0 o un 1.

- **Smoking History:** Estudios han encontrado que fumar incrementa el riesgo de desarrollar diabetes de tipo 2 al contribuir resistencia a la insulina y afectar al metabolismo de la glucosa. Puede ser “never”, “not current”, “current”, “former”, “ever” o “No info”.

- **HbA1c Level:** La hemoglobina glicosilada es un examen de sangre que mide el nivel promedio de azúcar en la sangre en los últimos dos o tres meses. Altos niveles de azúcar en la sangre son asociados con un alto riesgo de desarrollar diabetes. Número decimal que varía entre 3.5 y 9

- **Blood Glucose Level:** El nivel de glucosa en la sangre se mide en el momento de la

muestra. Niveles elevados de glucosa en la sangre cuando se está en ayunas o después de ingerir carbohidratos, pueden indicar una mala regulación de la glucosa haciendo que el nivel de riesgo de diabetes incremente. Número entero que varía entre 80 y 300.

3. LIMPIEZA DE DATOS

La limpieza de los datos se enfocó principalmente en tratar las features categóricas como lo son “gender” y “smoking_history”. Los que se hizo para gender fue primero eliminar la categoría “Other” ya que solo había 18 entradas de 100,000. Lo segundo que se hizo fue reducir las 6 posibles categorías de “smoking_history” a 3. Se agrupó “never” y “No info” como “non-smoker”. Se agrupó “ever”, “former” y “not current” como “past_smoker”. Por último, “current” se dejó por si solo.

4. DIVISIÓN DE LOS DATOS

La división de los datos fue la siguiente:

- Training = 80% del dataset

Maximiza los datos disponibles para aprender patrones y reducir el underfitting (sesgo)

- Validation = 10% del dataset

Se usa para la selección de hiperparámetros y ajuste de métricas. Al mantener este conjunto separado evitamos el overfitting por prueba y error.

- Testing = 10% del dataset

Esta sección se mantiene intacta hasta el final y se utiliza una vez para estimar el desempeño fuera de la muestra total. No se usa para tomar decisiones del modelo

Además, se aplica una estrategia llamada “stratified split” que nos ayuda a mantener un balance a la hora de aprender de las dos posibles salidas.

5. MODELO BASE

Para el modelo base, se usó un modelo de regresión logística con el cual podemos hacemos predicciones categóricas en vez de recibir un número. Para este modelo solo esperamos en la salida si un paciente padece diabetes o no con base en las “features” previamente mencionadas. La respuesta esperada recibimos un “0” si el paciente no padece diabetes o un “1” si el paciente padece diabetes.

La construcción de este modelo fue hecho sin usar librerías de machine learning, entonces el proceso se hizo “a mano”. Las funciones principales utilizadas son:

- Sigmoid: La función sigmoide toma cualquier número real y lo trasforma a un rango de 0 a 1.

- Hypothesis: La función de hipótesis es la función de predicción del modelo en donde se construye una combinación lineal de las variables X y theta. Luego se le aplica la función sigmoide y retorna una probabilidad de que la observación pertenezca a la clase positiva.

- Log_loss: La función log_loss, también conocida como binary cross-entropy los, es la función de error que mide que tan buena es la predicción. Promedia todas las observaciones de predicción y las promedia. Entre más se acercan las probabilidades predichas a las reales, menor es el resultado de log_loss.

- Gradient_descent_logloss: Es la función de que optimiza el modelo para encontrar los mejores parámetros. Funciona ajustando los parámetros poco a poco en la dirección que reduce el log_loss.

6. RESULTADOS MODELO BASE

```
=== TRAIN (80%) ===
LogLoss: 0.12347306298454205
Accuracy: 0.9596
Recall: 0.576764705882353
F1: 0.7081979053810039
Confusion (TP, FP, FN, TN): (3922, 354, 2878, 72846)

=== VALIDACIÓN (10%) ===
LogLoss: 0.12345291041879132
Accuracy: 0.9597
Recall: 0.5729411764705883
F1: 0.7073347857661584
Confusion (TP, FP, FN, TN): (487, 40, 363, 9110)

=== TEST (10%) ===
LogLoss: 0.12620768413238023
Accuracy: 0.9596
Recall: 0.5658823529411765
F1: 0.7042459736456809
Confusion (TP, FP, FN, TN): (481, 35, 369, 9115)
```

Los resultados, a primera vista, nos muestran un accuracy alto en todas las divisiones del dataset junto con un logloss bajo que nos indica que el modelo realiza predicciones precisas y seguras. Sin embargo, la matriz de confusión muestra un patrón en las tres divisiones y es el de un número relativamente alto de falsos negativos. Esto se debe a que, aunque el dataset tiene 100,000 muestras, solo 8500 de estas nos dan un diagnostico positivo de diabetes. Esto hace que las predicciones tiendan a darnos estos falsos negativos y que la estadística del "recall" (la estadística de verdaderos positivos) sea baja.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Por otro lado, tenemos el F1 score, que nos indica que tan bien el modelo de clasificación responde a un dataset con clases desbalanceadas como lo es en este caso. Para esta métrica necesitamos la precisión, que es la proporción de predicciones positivas correctas.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Combinando la precisión y el recall conseguimos el F1 score.

$$F_1 \text{ Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

7. FRAMEWORK

Para el modelo utilizando un framework de machine learning, se utilizó el framework del algoritmo de "random forest classifier". El algoritmo de random forest utiliza múltiples "árboles" para hacer clasificaciones. La manera en la que funcionan estos árboles es que con un nodo raíz que comienza con todos los datos, busca la variable que mejor separe las clases y crea una división con dos ramas donde la de la izquierda es la condición verdadera y la derecha es la falsa. Este proceso se repite recursivamente en cada rama hasta

que se cumple una condición de parada. Un solo árbol de decisión puede ser inestable debido a que pequeños cambios en los datos pueden cambiar mucho la predicción. Es por eso por lo que se mitiga esto con random forest al crear muchos árboles, entrenarlos en paralelo y combinar sus resultados. La ventaja principal del random forest es que es robusto contra el sobreajuste gracias a la combinación de múltiples árboles.

8. TRATADO DE DATOS

Para el framework, se usó el one-hot encoding con la feature de “smoking history” ya que, al tener 3 posibles clasificaciones, el algoritmo no sabe como interpretarlo y el one-hot enconding se encarga de esto al crear una columna binaria por cada categoría.

Smoking history	non-smoker	current	Past_smoker
non-smoker	1	0	0
current	0	1	0
past_smoker	0	0	1

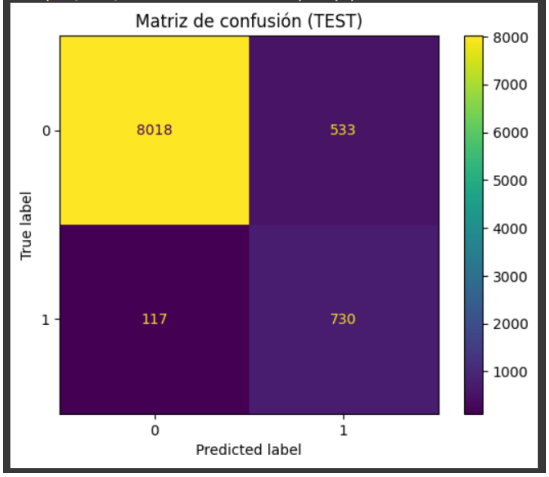
Otra modificación que se realizó a los datos para poder obtener mejores resultados en el framework fue la técnica del re-muestreo a la división de los datos usados para el training. Anteriormente se había denotado que en el dataset solo el 8.5% de las muestras dan como salida un resultado positivo. Para mitigar este sesgo se hizo un re-muestreo 1:4 donde solo se usó 4 veces la cantidad de muestras positivas para las muestras negativas. Las muestras para la validación y el test quedaron intactas.

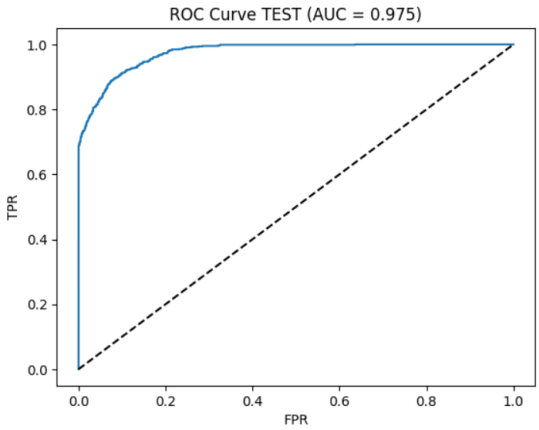
9. RESULTADOS

Los resultados iniciales del framework son los siguientes:

```
== TRAIN ==
Accuracy : 0.9424417367244865
AUC      : 0.9904126593794327
LogLoss  : 0.1293538433776342

Reporte de clasificación (TRAIN):
              precision    recall  f1-score   support

           0       0.99      0.94      0.97     68405
           1       0.62      0.94      0.75      6771

    accuracy                           0.94     75176
   macro avg       0.81      0.94      0.86     75176
weighted avg       0.96      0.94      0.95     75176


== VALIDATION ==
Accuracy : 0.9278493136107269
AUC      : 0.9749710608550063
LogLoss  : 0.14596324146789982

Reporte de clasificación (VAL):
              precision    recall  f1-score   support

           0       0.98      0.94      0.96      8551
           1       0.57      0.85      0.68       846

    accuracy                           0.93      9397
   macro avg       0.78      0.89      0.82      9397
weighted avg       0.95      0.93      0.93      9397


== TEST ==
Accuracy : 0.9308363481591828
AUC      : 0.975195565961133
LogLoss  : 0.14477118798423988

Reporte de clasificación (TEST):
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      8551
           1       0.58      0.86      0.69       847

    accuracy                           0.93      9398
   macro avg       0.78      0.90      0.83      9398
weighted avg       0.95      0.93      0.94      9398
```

```
OOB Score: 0.9277428966691498
Text(0.5, 1.0, 'Matriz de confusión (TEST)')
```





Para interpretar estos resultados primero hay que explicar el ROC curve (Reciever Operating Characteristic curve). Es una herramienta muy usada para evaluar el desempeño de un modelo de clasificación binaria. Es una gráfica que compara el True Positive Rate (TPR) que mide que tan bien se detectan los positivos reales y el False Positive Rate (FPR) que indica la proporción de negativos mal clasificados como positivos. En el eje X se coloca el FPR y en el eje Y el TPR

$$TPR = \frac{TP}{TP + FN}$$

True Postive Rate = True Positives / (True Positives + False Negatives)

$$FPR = \frac{FP}{FP + TN}$$

False Positive Rate = False Positives / (False Positives + True Negatives)

El Area Under the Curve (AUC) es el área debajo de la curva y con esta podemos medir que tan bueno es el modelo con estos rangos:

- 0.5 → El modelo es aleatorio (no sirve).
- 0.7–0.8 → Aceptable.
- 0.8–0.9 → Muy bueno.
- >0.9 → Excelente.

El AUC del modelo random forest es de 0.975 usando la división del dataset de test. Este resultado se considera como excelente.

Por otro lado, tenemos los datos de acurracy, AUC y logloss de cada división del dataset

TRAIN

- Accuracy : 0.9424417367244865
- AUC : 0.9904126593794327
- LogLoss : 0.1293538433776342

VALIDATION

- Accuracy : 0.9278493136107269
- AUC : 0.9749710608550063
- LogLoss : 0.14596324146789982

TEST

- Accuracy : 0.9308363481591828
- AUC : 0.975195565961133
- LogLoss : 0.14477118798423988

Todos estos resultados, de igual manera, se interpretan como buenos ya que el modelo parece estar aprendiendo bien y predice bien. Sin embargo, tenemos que poner atención a las otras métricas que previamente fueron explicadas en recall y f1-score:

Reporte de clasificación (TRAIN):

	precision	recall	f1-score	support
0	0.99	0.94	0.97	68405
1	0.62	0.94	0.75	6771

Reporte de clasificación (VAL):

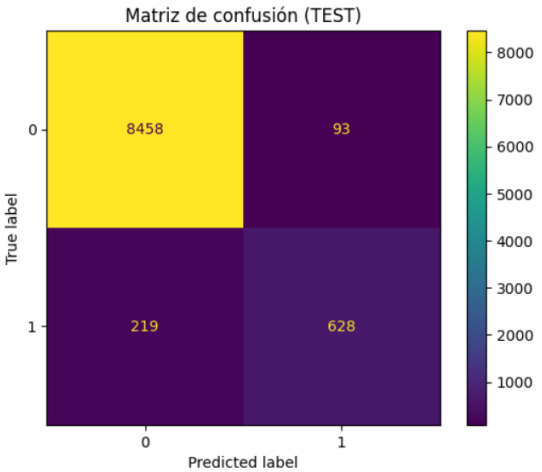
	precision	recall	f1-score	support
0	0.98	0.94	0.96	8551
1	0.57	0.85	0.68	846

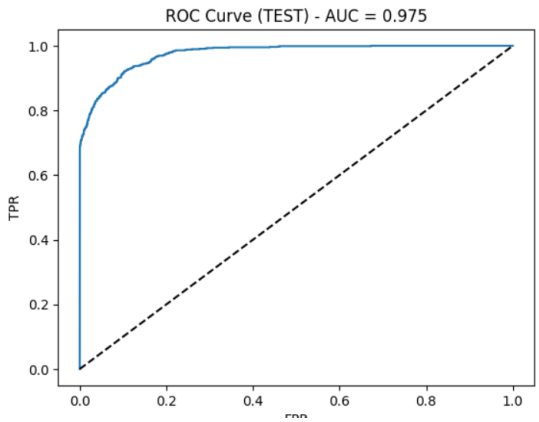
```
Reporte de clasificación (TEST):
              precision    recall  f1-score   support

           0       0.99      0.94      0.96      8551
           1       0.58      0.86      0.69       847
```

Como podemos observar, el f1-score para las clasificaciones que deberían de ser 1 (el paciente padece diabetes) el score es decente en train pero baja en validation y test. Esto se debe a lo previamente mencionado de que el dataset solo tiene el 8.5% de las muestras para entrenar las predicciones positivas entonces tenemos una cantidad alta de falsos positivos.

Ahora vamos a ver los resultados usando el re-muestreo 1:4.

```
=== TRAIN* (resampled) ===
Accuracy: 0.9561955398020971
ROC AUC: 0.9895732937045212
Log Loss: 0.11073055062890039
OOB Score: 0.9369960124058485
Matriz de confusión (TRAIN*):
 [[27025    59]
 [ 1424  5347]]
Reporte de clasificación (TRAIN*):
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     27084
           1       0.99      0.79      0.88      6771

    accuracy                           0.96     33855
   macro avg       0.97      0.89      0.93     33855
weighted avg       0.96      0.96      0.95     33855


=== VALID (10%) ===
Accuracy: 0.9653080770458657
ROC AUC: 0.9743935220549876
Log Loss: 0.10241516287072061
Matriz de confusión (VALID):
 [[8443  108]
 [ 218  628]]
Reporte de clasificación (VALID):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      8551
           1       0.85      0.74      0.79       846

    accuracy                           0.97      9397
   macro avg       0.91      0.86      0.89      9397
weighted avg       0.96      0.97      0.96      9397


=== TEST (10%) ===
Accuracy: 0.9668014471164077
ROC AUC: 0.9754761934677096
Log Loss: 0.10015146024993556
Matriz de confusión (TEST):
 [[8458   93]
 [ 219  628]]
Reporte de clasificación (TEST):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      8551
           1       0.87      0.74      0.80       847

    accuracy                           0.97      9398
   macro avg       0.92      0.87      0.89      9398
weighted avg       0.97      0.97      0.97      9398
```





TRAIN

- Accuracy: 0.9561955398020971
- ROC AUC: 0.9895732937045212
- Log Loss: 0.11073055062890039

VALIDATION

- Accuracy: 0.9653080770458657
- ROC AUC: 0.9743935220549876
- Log Loss: 0.10241516287072061

TEST

- Accuracy: 0.9668014471164077
- ROC AUC: 0.9754761934677096
- Log Loss: 0.10015146024993556

En general podemos apreciar como el accuracy ha incrementado en todas las divisiones, así como el logloss ha bajado. Pero ahora vamos a evaluar el cambio del f1-score.

Reporte de clasificación (TRAIN*):

	precision	recall	f1-score	support
0	0.95	1.00	0.97	27084
1	0.99	0.79	0.88	6771

Reporte de clasificación (VALID):

	precision	recall	f1-score	support
0	0.97	0.99	0.98	8551
1	0.85	0.74	0.79	846

Reporte de clasificación (TEST):

	precision	recall	f1-score	support
0	0.97	0.99	0.98	8551
1	0.87	0.74	0.80	847

Ahora podemos notar que la precisión y el recall han subido entonces el f1-score ha subido en todas las divisiones gracias a que el dataset pudo aprender de mejor manera sobre las muestras positivas. En la matriz de confusión podemos notar que ahora falsos positivos han bajado significativamente.

10. MEJORA

Para poder obtener una mejora con el framework de random forest, se hizo un ajuste a los parámetros de este. Los parámetros de la función de random forest son los siguientes:

- **n_estimators:** número de árboles usados. Entre más árboles, predicción más estable y precisa (menos varianza), pero también aumenta el tiempo de entrenamiento.

- **max_depth:** Profundidad máxima de los niveles de división de cada árbol.

- **min_samples_leaf:** Número mínimo de muestras que debe haber en una hoja final. Evita que el árbol cree hojas pequeñas y que cree sobreajuste.

- **n_jobs:** Número de núcleos de CPU usados en paralelo.

- **random_state:** Semilla de generación de números aleatorios. Garantiza reproducibilidad para obtener los mismos resultados y hacer ajustes sobre ellos.

- **oob_score**: Out-of-Bag score, estima internamente la precision sin usar un conjunto de validación. Usa el 63% de los datos como entrenamiento y el 37% para evaluar. Se considera una validación cruzada integrada en el modelo.

Los principales parámetros que fueron ajustados para encontrar una mejora fueron “max_depth” y “min_samples_leaf”.

Para la comparación de parámetros vamos a mostrar solo las métricas de precisión, recall y f1-score ya que pudimos apreciar que accuraccy es un poco engañoso.

max_depth = 10

min_samples_leaf = 3

Reporte de clasificación (TRAIN*):

	precision	recall	f1-score	support
0	0.93	1.00	0.96	27084
1	0.99	0.71	0.83	6771

Reporte de clasificación (VALID):

	precision	recall	f1-score	support
0	0.97	1.00	0.98	8551
1	0.94	0.72	0.82	846

```
Reporte de clasificación (TEST):
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      8551
           1       0.94      0.72      0.81       847
```

max_depth = 15

min_samples_leaf = 5

```
Reporte de clasificación (TRAIN*):
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     27084
           1       0.98      0.77      0.86      6771
```

```
Reporte de clasificación (VALID):
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      8551
           1       0.88      0.74      0.80       846
```

```
Reporte de clasificación (TEST):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      8551
           1       0.88      0.74      0.80       847
```

max_depth = 25

min_samples_leaf = 5

```
Reporte de clasificación (TRAIN*):
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     27084
           1       0.98      0.79      0.88      6771
```

```
Reporte de clasificación (VALID):
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      8551
           1       0.86      0.75      0.80       846
```

```
Reporte de clasificación (TEST):
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      8551
           1       0.86      0.74      0.79       847
```

La conclusión es que entre más subimos los parámetros, estos van generando más varianza y esto muestra evidencia de overfitting. Con solo 10 niveles de profundidad y 3 hojas podemos crear un modelo que generaliza bien y tiene bajo sesgo (el sesgo se nota en la clase positiva al detectar menos positivos), en otras palabras, esta “fit”. Esto lo podemos apreciar en como los f1-scores de las tres divisiones son muy similares.

11. CONCLUSIONES

A la hora de realizar modelos para aplicarlos en la vida real, es muy importante tener en consciencia la verdadera efectividad del modelo tomando en cuenta el contexto de la situación de la que se basa el dataset. Para predicciones médicas, como lo es el diagnóstico de la diabetes, es importante estar consciente de la dinámica de los falsos positivos y los falsos negativos ya que un diagnóstico erróneo puede cambiar la trayectoria de la vida de una persona de manera drástica. Es por eso por lo que mi enfoque en el modelo de random forest fue en las estadísticas de la precisión, recall y f1-score en conjunto con la matriz de

confusión. En todas mis pruebas, siempre obtenía resultados con accuraccy altos y logloss bajos, dándome la primera impresión de que estos modelos eran buenos. Pero, al notar la alta cantidad de falsos positivos y dándome cuenta de que en mi dataset de 100,000 muestras solo había 8500 muestras que indicaban que el paciente padecía diabetes, cambió mi enfoque a realmente tomar esa situación y buscar una solución para obtener un modelo que pueda ser usado en la vida real.

12. REFERENCIAS

- Diabetes. (2025, 11 septiembre). Cleveland Clinic. https://my.clevelandclinic.org/health/diseases/7104-diabetes

- GeeksforGeeks. (2025, 11 julio). One Hot Encoding in Machine Learning. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/ml-one-hot-encoding/

- Hemoglobin A1C (HBA1C) test. (s. f.). https://medlineplus.gov/lab-tests/hemoglobin-a1c-hba1c-test/

- StratifiedShuffleSplit. (s. f.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

- Khan Academy. (s. f.). https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent#:~:text=Gradient%20descent%20is%20an%20algorithm,like%20we've%20seen%20before.

- KoshurAI. (2024, 9 enero). Understanding Log Loss: A Comprehensive Guide with Code Examples. Medium. https://koshurai.medium.com/understanding-log-loss-a-comprehensive-guide-with-code-examples-c79cf5411426

- GeeksforGeeks. (2025b, julio 23). F1 Score in Machine Learning. GeeksforGeeks. https://www.geeksforgeeks.org/machine-learning/f1-score-in-machine-learning/

- RandomForestClassifier. (s. f.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

- Clasificación: ROC y AUC. (s. f.). Google For Developers. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419

- Radhika. (2024, 26 febrero). *Understanding Out-of-Bag (OOB) score: Random Forest Algorithm evaluation*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/12/out-of-bag-oob-score-in-the-random-forest-algorithm/