Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Estado de México

Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución

TC3006C. Inteligencia artificial avanzada para la ciencia de datos

Módulo 2. Aprendizaje Máquina

Grupo: 101

A01749075. Ameyalli Contreras Sánchez

Prof. Jorge Adolfo Ramírez Uresti

Fecha de entrega: 10 de septiembre de 2025

Semestre agosto - diciembre 2025

Índice

1. Dataset empleado

De la plataforma de Kaggle se descargó el dataset ***Student Performance (Multiple linear regression)*** en el cual se presenta información sobre los hábitos de estudio de distintos estudiantes durante el periodo escolar, tales como horas de estudio previas al examen, sus notas en el examen previo, si son parte de actividades extracurriculares, horas de sueño, los documentos de práctica que revisaron y como variable objetivo su desempeño final.
Se optó por transformar el problema en uno de clasificación, es decir se transformó la variable objetivo en una variable binaria, donde calificaciones finales por debajo de 70 se considera como la clase *"Failed"* = 0, y arriba de esa nota se considera *"Passed"* = 1.
Se utilizó el mismo dataset que en la actividad *"Implementación de una técnica de aprendizaje máquina sin el uso de un framework"* con el objetivo de poder comparar los resultados obtenidos en ambos entregables.

A continuación se presenta un extracto del dataset final empleado para el modelo y su información obtenida con el comando data.info():

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	1	9	1	1
1	4	82	0	4	2	0
2	8	51	1	7	2	0
3	5	52	1	5	2	0
4	7	75	0	8	5	0
...	...	...	...	...	...	...
9995	1	49	1	4	2	0
9996	7	64	1	8	5	0
9997	6	83	1	8	5	1
9998	9	97	1	7	0	1
9999	7	74	0	8	1	0

10000 rows × 6 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 6 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Hours Studied                   10000 non-null  int64
 1   Previous Scores                 10000 non-null  int64
 2   Extracurricular Activities      10000 non-null  int64
 3   Sleep Hours                     10000 non-null  int64
 4   Sample Question Papers Practiced  10000 non-null  int64
 5   Performance Index               10000 non-null  int64
dtypes: int64(6)
memory usage: 468.9 KB
```

```
Performance Index
0    7495
1    2505
Name: count, dtype: int64
```

Como se puede observar, la variable objetivo cuenta con un alto nivel de desbalance en sus datos. Sin embargo, se optó por intentar aplicar el modelo de machine learning sin realizar modificaciones en el dataset, es decir sin aplicar ninguna especie de submuestreo o sobremuestreo. Esto con el propósito de experimentar con el modelo elegido (Random forest) para obtener un resultado asociado directamente a los datos originales.
El data split se realizó utilizando el parámetro *stratify = y* para asegurar que la proporción de clases en el conjunto de entrenamiento y en el conjunto de prueba sea la misma que en el dataset original. Esto es importante, porque así se evita que el conjunto de prueba o de entrenamiento quede sin ejemplos de la clase minoritaria, asegurando que ambos sean representativos de la distribución real.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

A continuación se muestra la información correspondiente a los datos de entrenamiento y de prueba obtenidos tras el seccionamiento.

1.1. Datos de entrenamiento

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
8831	2	47	0	9	1	0
6652	8	59	0	8	7	0
8276	5	68	1	7	7	0
46	1	43	1	7	0	0
5422	1	66	1	5	3	0
...	...	...	...	...	...	...
928	9	71	0	5	7	0
7112	6	68	0	5	5	0
6704	3	92	0	4	2	1
6645	6	65	1	5	4	0
188	3	91	0	5	6	1

8000 rows × 6 columns

1.2. Datos de prueba

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
219	3	60	1	4	4	0
1776	7	97	1	5	2	1
8055	6	67	0	5	5	0
7173	2	43	1	6	8	0
529	9	72	0	5	6	1
...	...	...	...	...	...	...
1538	5	81	1	5	3	0
7640	2	74	1	6	8	0
3258	4	98	1	9	8	1
1752	8	47	1	6	8	0
1177	7	72	1	7	3	0

2000 rows × 6 columns

2. Resultados del modelo

En la presente sección se muestran los resultados de la implementación de un Random forest para intentar clasificar el desempeño de los alumnos.
Se desarrollaron varias versiones de Random Forest Classifier, para todos se utilizaron los mismos conjuntos de entrenamiento y prueba, con ligeras modificaciones que se mencionarán más adelante.
Nuevamente se buscó obtener resultados donde los aciertos de cada clase estuvieran lo más balanceados posibles, pero teniendo como objetivo principal, evitar clasificar como Failed datos que pertenecen a la clase Passed.

2.1. Random Forest inicial

A manera de inicio, se generó un random forest de forma "aleatoria", es decir se colocaron valores que al autor le parecieron adecuados para no sobrecargar el modelo y tener un punto de partida para posteriormente seleccionar valores más acertados para la optimización de parámetros que se realizará en la siguiente subsección.
Se definieron los hiperparámetros siguientes:

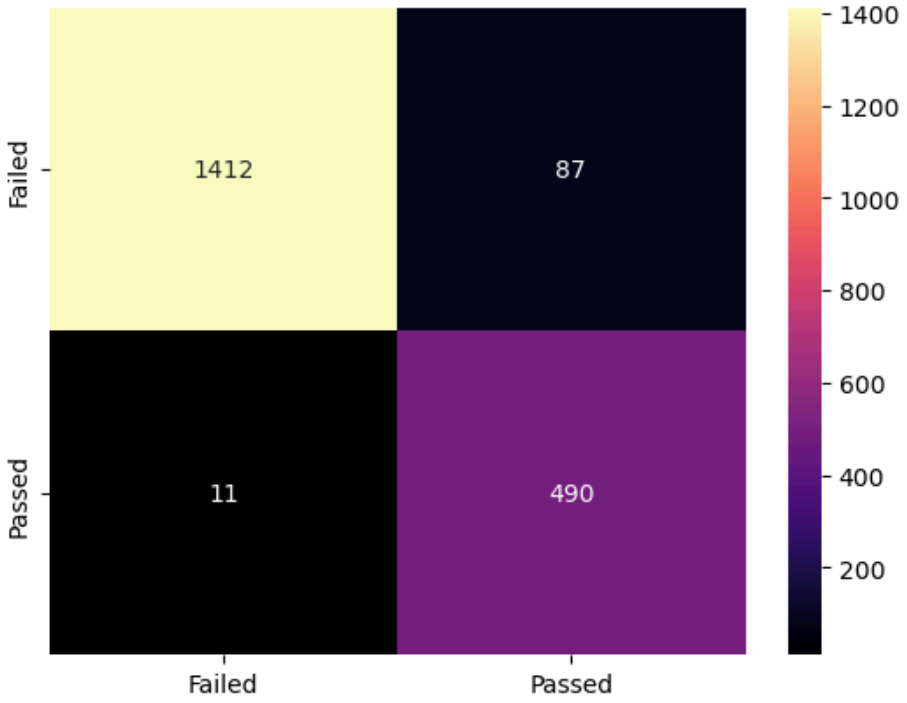
```
rf = RandomForestClassifier(
    n_estimators=20,           # número de árboles
    max_depth=8,               # profundidad máxima
    min_samples_split=200,     # mínimo de muestras para dividir un nodo
    min_samples_leaf=100,      # mínimo de muestras por hoja
    max_features="sqrt",       # n° de features consideradas en cada split
    bootstrap=True,            # muestreo con reemplazo
    class_weight="balanced",   # útil en caso de desbalance
    random_state=42,           # reproducibilidad
    n_jobs=-1                  # uso de todos los núcleos
)
```

y los resultados obtenidos se evaluaron utilizando las métricas de desempeño, prestando especial atención al F1-Score debido a que las clases se mantuvieron desbalanceadas, y se usó la matriz de confusión para visualizar más fácilmente las clasificaciones hechas, como se muestra a continuación.

Classification report

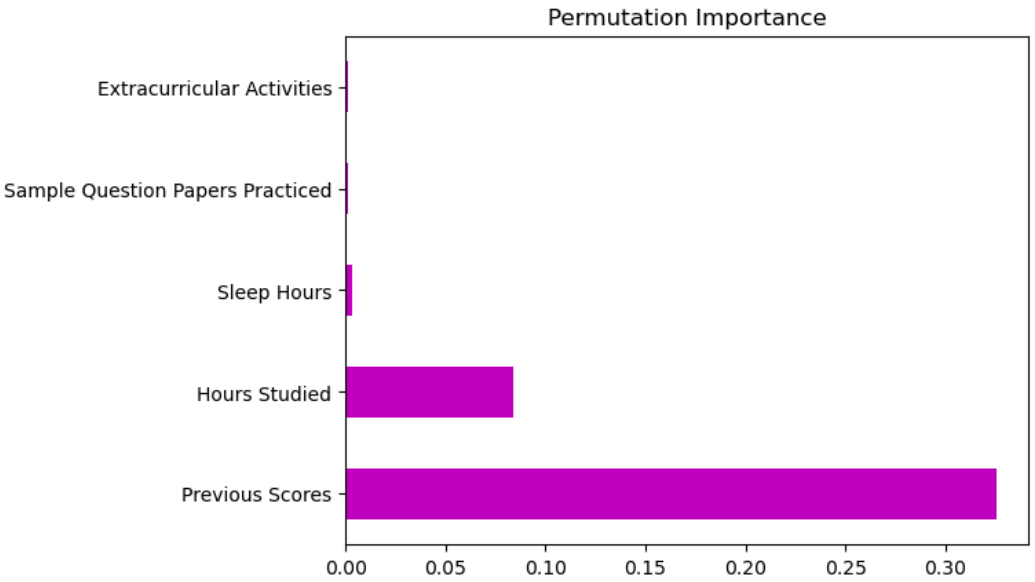
```
>>> REPORTE DE CLASIFICACIÓN: <<<
              precision    recall  f1-score   support

           0       0.99      0.94      0.97      1499
           1       0.85      0.98      0.91       501

    accuracy                           0.95      2000
   macro avg       0.92      0.96      0.94      2000
weighted avg       0.96      0.95      0.95      2000


>>> AUC-ROC: 0.9930465952684358 <<<
```

Matriz de confusión



Como se observa en la matriz, se cumplió el objetivo de clasificar más adecuadamente los datos pertenecientes a la clase Passed, sin embargo, existen posibilidades de mejora respecto a la clase Failed, la cual presenta cerca de 90 valores mal clasificados. Adicionalmente, es importante notar que según las métricas de desempeño, esta primera aproximación de hiperparámetros genera resultados satisfactorios. Además el AUC-ROC score de 0.99 en conjunto con las otras métricas de desempeño, demuestra que el modelo casi siempre es capaz de distinguir entre clases y que a pesar de tratarse de un dataset desbalanceado, el modelo no solo se concentra en la clase mayoritaria. Para intentar mejorarlos, se realizó una optimización de hiperparámetros.

Importancia de variables



El análisis de permutation importance, evalúa la importancia de cada variable dentro del random forest empleado, en este caso es fácil visualizar que las dos variables más importantes resultan ser *Previous Scores* y *Hours Studied*, lo que destaca que son las variables que mejor ayudan a la clasificación del desempeño del estudiante, información que será relevante retomar más adelante.

2.2. Random Forest con optimización de hiperparámetros

Se realizó una optimización de hiperparámetros mediante el uso de Grid Search, probando valores alrededor de los empleados en el primer modelo (tanto más pequeños, como más altos) de forma que se pudiera explorar un abanico de 7200 soluciones que se podían obtener. Al final los hiperparámetros recomendados, que fueron empleados para el entrenamiento de un nuevo árbol, fueron los siguientes:

```
Fitting 5 folds for each of 1440 candidates, totalling 7200 fits

>>> MEJORES PARÁMETROS: <<<
 {'max_depth': 15, 'max_features': 'sqrt', 'min_samples_leaf': 20, 'min_samples_split': 50, 'n_estimators': 25}
```

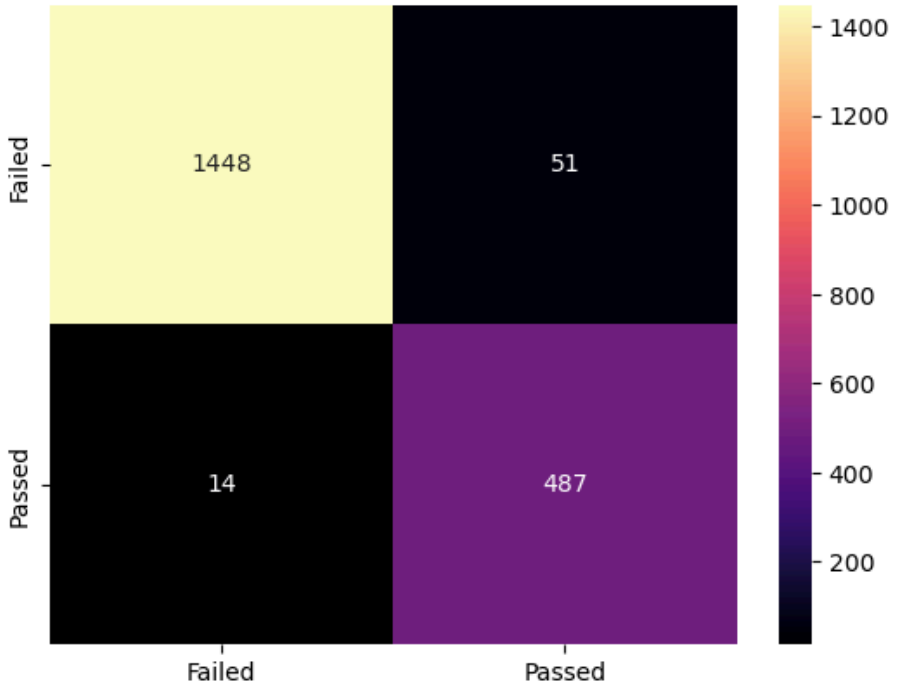
2.2.1. Usando todas las variables

En primera instancia se optó por evaluar el modelo con todas las variables, tal como se hizo en el modelo inicial, de forma que hubiera una comparación más directa de los resultados obtenidos con el primer modelo.

Reporte de clasificación

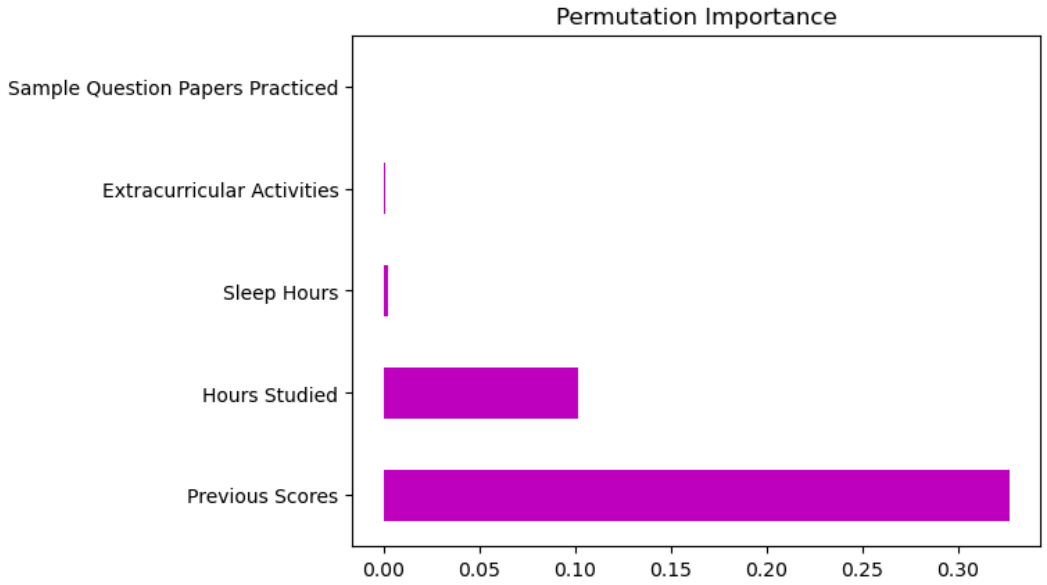
```
>>> REPORTE DE CLASIFICACIÓN: <<<
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1499
           1       0.91      0.97      0.94       501

    accuracy                           0.97      2000
   macro avg       0.95      0.97      0.96      2000
weighted avg       0.97      0.97      0.97      2000


>>> AUC-ROC: 0.9963581842319363 <<<
```

Matriz de confusión



Como se observa en el reporte de clasificación, sí se presentó una mejora significativa en cuanto a la clasificación de la clase Failed, al punto en el que ambas clases tienen un recall del 97%. Además al observar la matriz de confusión se observa que efectivamente se logró disminuir la cantidad de errores cometidos al clasificar la clase Failed. Sin embargo, la clase Passed obtuvo 3 errores más que en el modelo inicial, lo que es perjudicial para nuestro objetivo secundario, pero al ser un cambio mínimo y tener un F1-Score del 94% podría no ser tan perjudicial. Nuevamente, el AUC-ROC score en conjunto con el resto de métricas, muestra un muy buen desempeño del modelo al distinguir entre clases. Además, en este modelo con hiperparámetros optimizados obtuvo un valor ligeramente más alto (0.9963 vs 0.9930 del inicial).

Importancia de las variables



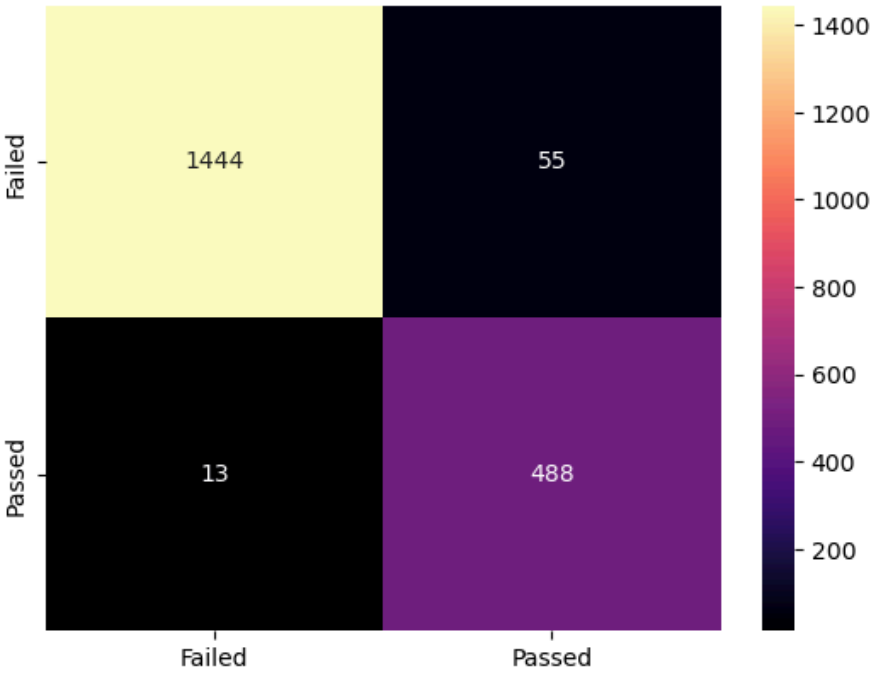
En cuanto al análisis de importancia de las variables, se observa exactamente el mismo patrón de importancia que en el modelo inicial, lo que era de esperarse. Por motivos de experimentación, se optó por probar el random forest con los hiperparámetros pero, reduciendo la cantidad de variables una a una, de la menos importante, hasta quedar únicamente con las dos variables más relevantes. Esto con el objetivo de observar qué tanto se ve afectado el modelo si solo se conservan aquellas variables que tienen mayor importancia. Sus resultados se presentan a continuación.

2.2.2. Usando las 4 variables más importantes

Reporte de clasificación

```
>>> REPORTE DE CLASIFICACIÓN: <<<
              precision    recall  f1-score   support

           0       0.99      0.96      0.98      1499
           1       0.90      0.97      0.93       501

    accuracy                           0.97      2000
   macro avg       0.94      0.97      0.96      2000
weighted avg       0.97      0.97      0.97      2000


>>> AUC-ROC: 0.9968235643456249 <<<
```

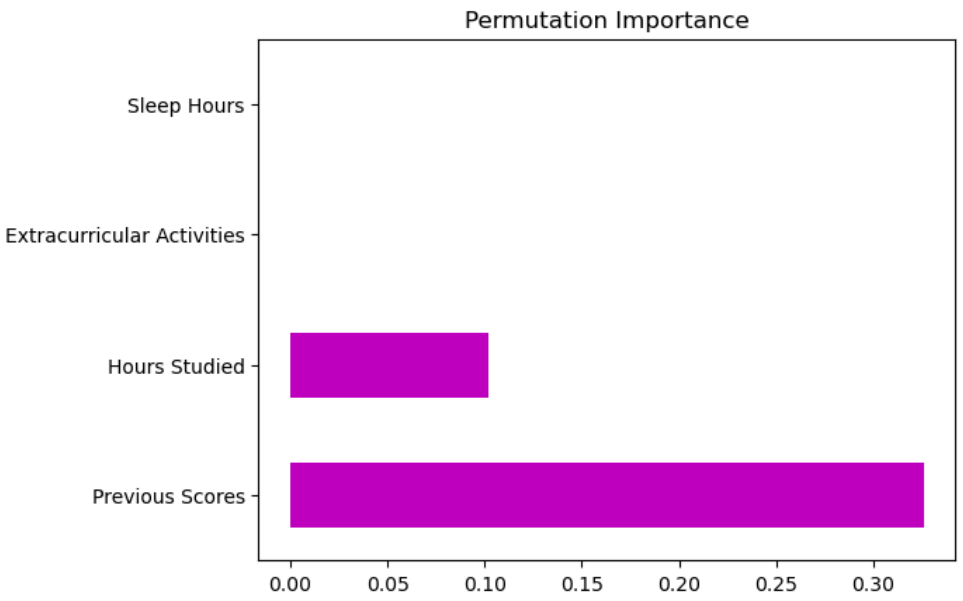
Matriz de confusión



En este modelo se puede apreciar rápidamente como el eliminar una sola variable (por menos importante que parezca) sí afecta el desempeño del modelo al final, ya que el recall de la clase Failed cayó un 1%, y el F1-Score y precision de la clase Passed también decayeron un 1%. Sin embargo, al observar la matriz de confusión, se nota que clasificó correctamente un dato más de la clase Passed, mientras que, como lo indicó el recall, la clasificación de la clase Failed cometió 3 errores más que en el modelo anterior. Pese a ello, no se considera que sea un mal modelo, ya que, se cumple el objetivo de que cometa un número balanceado de errores para cada clase, y que aún así sea mucho más preciso al momento de clasificar los datos de la clase Passed.

Importancia de variables

En los modelos subsecuentes, los gráficos de importancia de variables son para visualizar dentro del presente reporte qué variables se están tomando en cuenta para el modelo, por lo que no habrá un análisis de esta sección, ya que los niveles de importancia se mantienen estáticos.

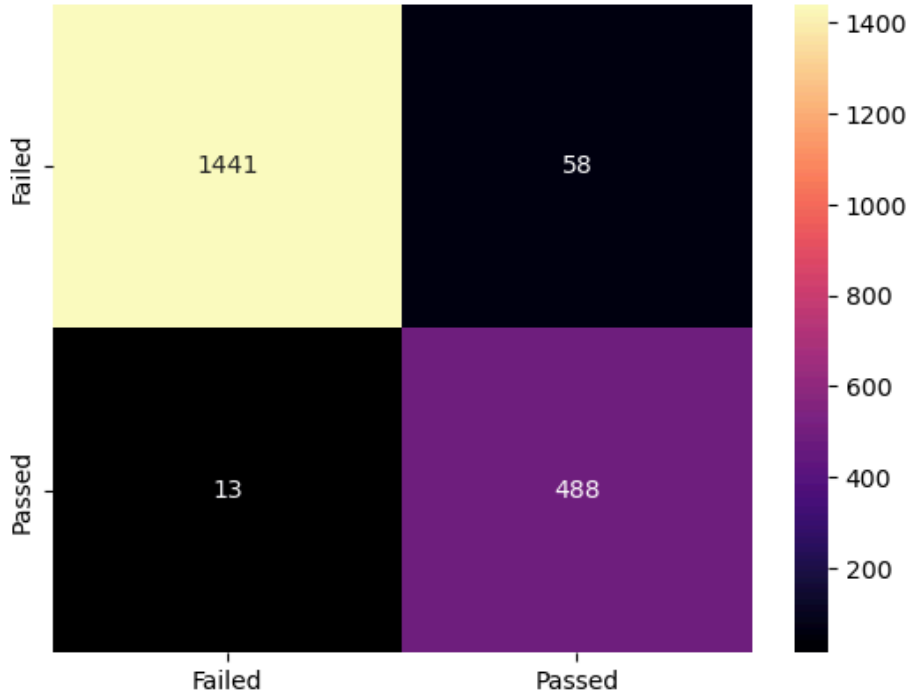


2.2.3. Usando las 3 variables más importantes

Reporte de clasificación

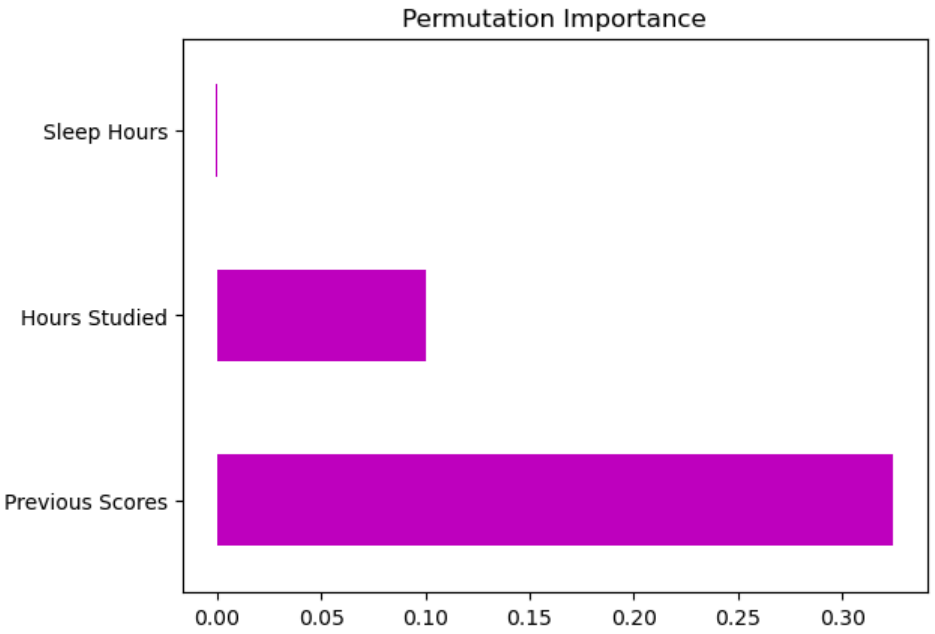
```
>>> REPORTE DE CLASIFICACIÓN: <<<
              precision    recall  f1-score   support

           0       0.99      0.96      0.98      1499
           1       0.89      0.97      0.93       501

    accuracy                           0.96      2000
   macro avg       0.94      0.97      0.95      2000
weighted avg       0.97      0.96      0.96      2000


>>> AUC-ROC: 0.9963621789110237 <<<
```

Matriz de confusión



Para este modelo, en el que se utilizaron únicamente 3 variables, se puede observar directamente en la matriz de confusión que su desempeño respecto a la clase Failed sigue disminuyendo, pues en esta ocasión cometió más errores. Además, según el reporte de clasificación la precisión de la clase Passed disminuyó otro 1%, mientras que el resto de sus métricas se mantuvieron estáticas. Este decremento constante de precisión indica que las variables eliminadas no eran críticas para detectar correctamente los Passed (por ello se mantiene un recall estable), pero sí ayudaban a reducir errores al clasificar Failed (menos False Positives). Por otra parte el AUC-ROC Score se mantiene dentro del mismo rango 0.9963 lo que indica que el modelo sigue teniendo un buen desempeño al distinguir entre clases.

Importancia de variables

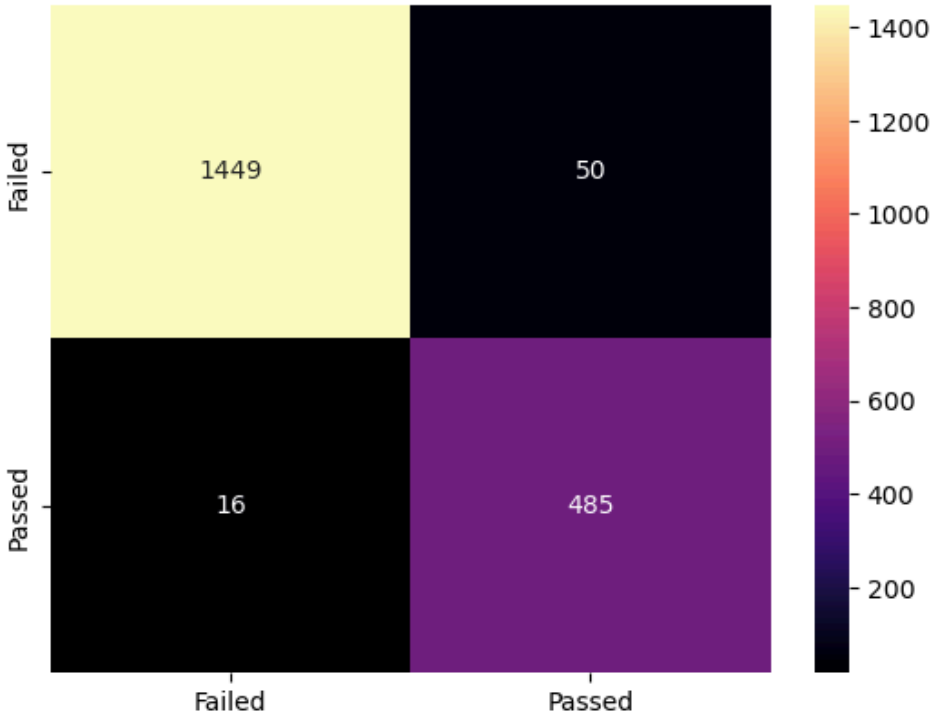


2.2.4. Usando las 2 variables más importantes

Reporte de clasificación

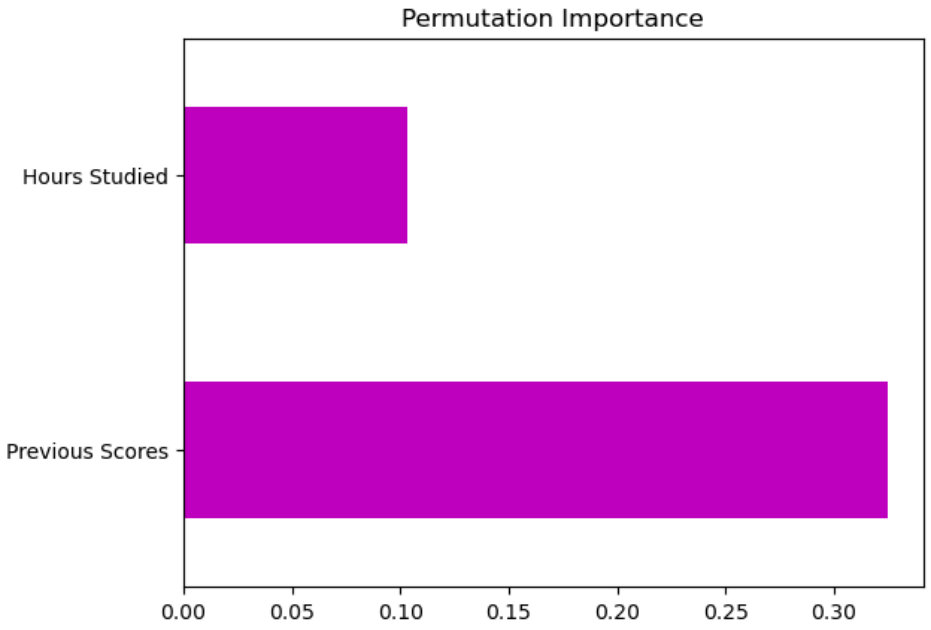
```
>>> REPORTE DE CLASIFICACIÓN: <<<
              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1499
           1       0.91      0.97      0.94       501

    accuracy                           0.97      2000
   macro avg       0.95      0.97      0.96      2000
weighted avg       0.97      0.97      0.97      2000


>>> AUC-ROC: 0.9970232982999977 <<<
```

Matriz de confusión



Para este último modelo, se observa un cambio interesante respecto a los dos modelos anteriores que parecían ir disminuyendo su capacidad de clasificar correctamente la clase Failed, pues en este caso se obtuvo el número más bajo de errores al clasificar esta clase. Si se observa cuidadosamente el reporte de clasificación, se puede notar que sus valores de las métricas de desempeño son prácticamente idénticos al modelo con hiperparámetros optimizados que utiliza las 5 variables independientes, con la diferencia de que su AUC-ROC Score es ligeramente más alto (0.9970), lo que indica que distingue ligeramente mejor las clases que el modelo con las 5 variables (0.9963). Además, la cantidad de errores cometidos al clasificar cada una de las clases, según lo observado en la matriz de confusión, varía muy poco respecto del primero modelo con hiperparámetros optimizados, por lo que, a nivel interpretabilidad y simplicidad este modelo podría ser una muy buena opción para solucionar este problema.

Importancia de variables



3. Análisis y conclusiones

Tras analizar individualmente cada uno de los modelos, se observó que, aquellos que presentaron tener mejores resultados, además de cumplir con los objetivos planteados desde el entregable anterior, fueron el modelo optimizado que utiliza todas las variables independientes y el modelo optimizado que utiliza solo las 2 variables más importantes. Ambos cumplen satisfactoriamente con los objetivos planteados en el entregable anterior y muestran un reporte de clasificación prácticamente idéntico, diferenciándose únicamente en el AUC-ROC Score, donde el segundo modelo obtuvo un rendimiento ligeramente superior.

La elección entre ambos modelos depende del criterio de aplicación: si se prioriza la simplicidad y facilidad de interpretación, resulta recomendable emplear el modelo con solo dos variables. En cambio, si se busca maximizar el aprovechamiento de toda la información disponible, incluso de las variables con aportes marginales, el modelo que integra todas las variables es la opción más adecuada.

A diferencia del entregable anterior, en el cual la implementación manual del modelo de perceptrón con backpropagation y gradient descent alcanzó métricas cercanas a 0.76 en F1, accuracy y demás indicadores, los modelos desarrollados en el presente reporte lograron desempeños muy cercanos al 100%. Esta mejora significativa se debe principalmente a la mayor complejidad y capacidad predictiva de los Random Forests, que al ser un algoritmo de ensamblado permiten capturar relaciones no lineales y reducir el riesgo de sobreajuste mediante el consenso de múltiples árboles de decisión. En contraste, el modelo manual utilizado previamente presentaba limitaciones tanto en su poder de representación como en la eficiencia del entrenamiento, lo cual explica la gran diferencia observada en los resultados.

En conclusión, al comparar el modelo del presente reporte con el entregable anterior, donde el perceptrón implementado manualmente mostró un desempeño limitado, los modelos de Random Forest desarrollados demostraron un rendimiento sobresaliente y consistente en todas las métricas de evaluación. Esto confirma que, al aprovechar algoritmos de ensamblado más robustos, es posible alcanzar resultados mucho más precisos y confiables, ofreciendo una base sólida para futuras aplicaciones y análisis.

NOTA: Para visualizar de manera más completa los resultados mostrados en el presente reporte abrir el archivo *MLconFramework.ipynb* dentro del repositorio de GitHub donde también se encontrará la versión del mismo código como archivo de python *Act2_MLconFramework.py* .

Link al repositorio de GitHub: ***A01749075-PortafolioImplementacion/Modulo 2_ML/Act 2***