Predicción de resultados académicos universitarios

FIB - APA

2023 - 2024 Q1

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Mario Fernández Simón

Joan Sales de Marcos

ÍNDICE

El problema

El objetivo de esta práctica es utilizar diferentes modelos de aprendizaje automático para resolver problemas reales. Haremos un preprocesado de los datos, usaremos métodos lineales para resolver el problema, junto con métodos no lineales y finalizaremos eligiendo el mejor modelo para nuestro caso concreto, comparándolos entre ellos.

Descripción del conjunto de datos y sus fuentes

El conjunto de datos escogido para la práctica corresponde a las estadísticas de estudiantes inscritos en diversas Universidades de Portugal. Por lo tanto, cada entrada de nuestro dataset corresponderá a un estudiante en concreto.
El objetivo será predecir cómo finalizan los estudiantes el curso al final de la carrera y clasificarlos en "dropout", "enrolled" y "graduate" (Si dejan la carrera, si aún no han acabado después de los años esperados de carrera, o si se gradúan en el tiempo que dura la carrera), es decir, estamos tratando un problema de clasificación.
A priori, nos encontramos con 35 variables y 1 variable objetivo de tipo categórica, la cual transformamos en numérica usando "label encoding" más adelante.

Estudios anteriores del dataset

Tan solo existe un estudio que haya utilizado el dataset de esta práctica. Concretamente, el estudio mencionado se encarga de evaluar el dataset y analizar si es un buen conjunto de datos de uso académico. Además, comprueba si cumple otras características clave, como por ejemplo, FAIR (Findability, Accessibility, Interoperability and Reusability), condiciones de privacidad de los datos, etc.
En resumen, el estudio llega a la conclusión que es un buen dataset para realizar experimentos con machine learning de carácter educativo como se indica en la página 11 del pdf (o en el primer párrafo del punto *4. conclusions*):

"*The dataset is useful for researchers who want to conduct comparative studies on student academic performance and also for training in the machine learning area.*" [2]

Preprocesamiento de los datos

Antes de empezar con los experimentos hemos hecho un preprocesado de los datos. Para este apartado, la mayoría de explicaciones las encontramos en el código. Por tanto, para evitar sonar repetitivos y que el documento no sea demasiado extenso, ya que en el enunciado pone que máximo 15 páginas, sólo haremos un resumen de ellas muy por encima.

En primer lugar, es necesario hacer una visualización general de los datos para ver a qué tipo de problema nos enfrentamos. A continuación, realizaremos comprobaciones básicas del dataset para asegurar nos de que todo funcione correctamente y eliminaremos todas aquellas variables que consideremos inservibles.

La primera comprobación realizada será ver valores únicos de las variables o valores perdidos. Una vez realizado este paso, hemos eliminado una serie de variables, las cuales no estaban bien explicadas en el dataset y, por tanto, no seríamos capaces de explicar claramente los resultados o aquellas que contienen muchos valores únicos que tampoco están explicados.

Más tarde, miramos las distribuciones de las variables numéricas y sus correlaciones. Descubrimos que algunas variables son casi siempre ceros y otras no son Gaussianas y siguen una distribución pseudo-aleatoria. Por esos motivos, prescindimos de ellas.

También, hemos analizado las correlaciones entre variables y con la variable objetivo. La observación más relevante que hemos obtenido entre variables es que las variables de créditos matriculados y aprobados están fuertemente relacionadas. Estas últimas también están fuertemente relacionadas con la variable objetivo (Figura 1.1).

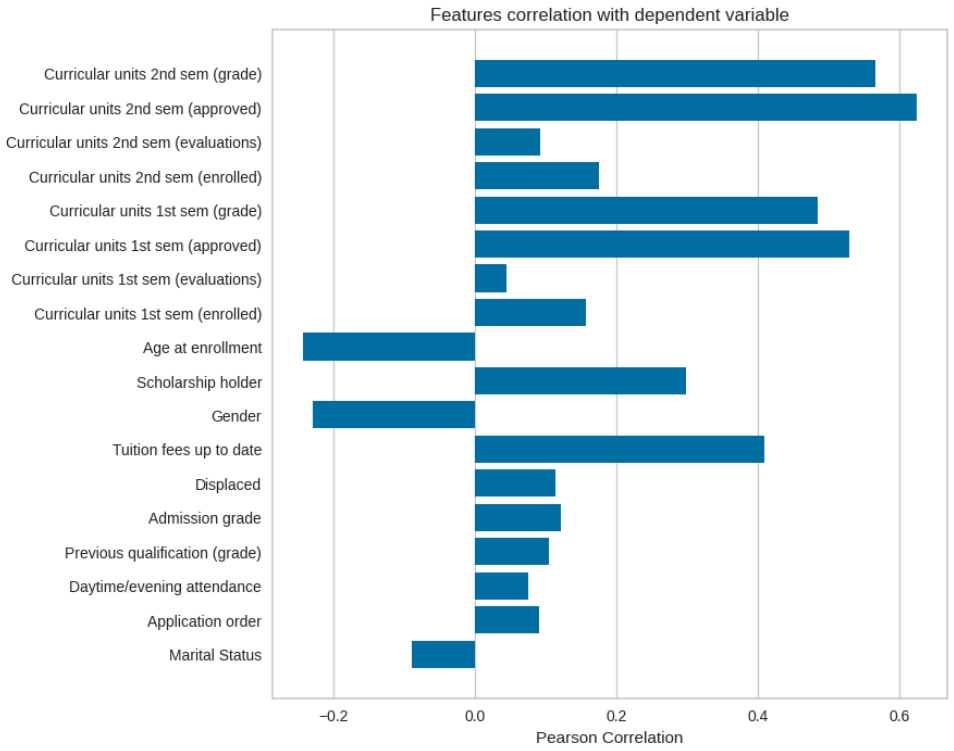

Figura 1.1. Correlaciones con la variable objetivo

Por último, realizaremos aplicaremos un PCA a los datos (Figura 1.2). De esta técnica, podemos ver cómo afecta el ir añadiendo componentes. Se observa que sin contar el primer componente no hay componentes que acumulen por si solo mucha varianza. Llegamos al 80% de varianza acumulada con 8 componentes y necesitamos unos 15 para llegar al 100%.

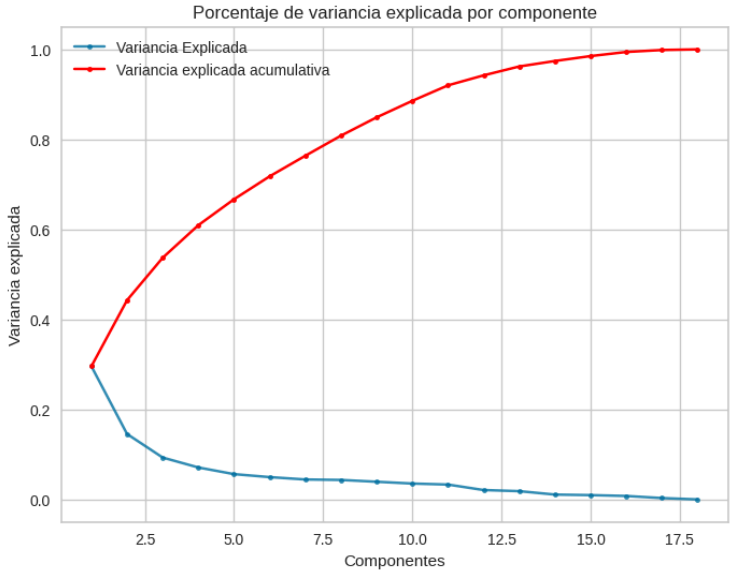

Figura 1.2. Porcentaje de Varianza por Componente

Al finalizar la limpieza de las variables nos hemos quedado con las siguientes:

- **Marital status:** Estado legal matrimonial (entre 9 opciones)
- **Application order:** orden de aplicación (entre 0-primera opción; y 9-última opción).
- **Daytime/evening attendance:** 1 si va de mañanas y 0 si va por la tarde.
- **Previous qualification** (**grade**)**:** estudios cursados anteriormente.
- **Admission grade:** nota de admisión (entre 0 y 200).
- **Displaced:** 1 si se ha mudado para ir a la universidad o 0 por el contrario.
- **Tuition fees up to date:** 1 si lleva las facturas al día o 0 por el contrario.
- **Gender:** 1-macho y 0-hembra.
- **Scholarship holder:** 1 dispone de beca o 0 por el contrario.
- **Age at enrollment:** edad al inscribirse en el curso.
- **Curricular units 1st/2nd sem** (**enrolled**)**:** créditos cursados.
- **Curricular units 1st/2nd sem** (**evaluations**)**:** créditos evaluados.
- **Curricular units 1st/2nd sem** (**approved**)**:** créditos aprobados.
- **Curricular units 1st/2nd sem** (**grade**)**:** nota obtenida.

Métodos Lineales

En esta práctica utilizaremos 3 métodos lineales de clasificación: Naive Bayes, LDA y Regresión Logística.

Naive Bayes

El primer modelo lineal utilizado es Naive Bayes que es el más simple. Empezaremos evaluando el valor de validación cruzada (Figura 1.1) y el *clasification report* (Figura 1.2).

```
Cross Validation: 0.7179886685552409
```

Figura 2.1. Validación Cruzada NB

Este resultado tiene rango entre 0 y 1, siendo 0 que el modelo no es capaz de predecir bien los datos y 1 que los predice perfectamente. Obtenemos un valor aproximado de 0.72, el cual no es mal resultado, pero tiene margen de mejora.

	precision	recall	f1-score	support
dropout	0.68	0.73	0.71	266
enrolled	0.35	0.48	0.41	110
graduate	0.88	0.78	0.83	509
accuracy			0.73	885
macro avg	0.64	0.66	0.65	885
weighted avg	0.76	0.73	0.74	885

Figura 2.2. Clasification report NB

A continuación, miramos el *clasification report* donde tenemos varios parámetros. En primer lugar, encontramos **precision**. Esta métrica mira la cantidad de instancias clasificadas como positivas que realmente lo son.

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

Figura 2.3. Fórmula de precision

En segundo lugar, tenemos **recall** que mira la proporción de de instancias positivas que se clasifican correctamente respecto al número real de ellas.

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

Figura 2.4. Fórmula de recall

En tercer lugar, nos encontramos con **f1-score.** Esta métrica mira un balance entre *precision* y *recall.* Es útil observarla cuando se busca un poco de las dos anteriores.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figura 2.5. Fórmula de f1-score

Por último, en **support** vemos la cantidad de instancias clasificadas en cada clase.
Tras esta breve explicación, pasamos a analizar los resultados.
Rápidamente, se ve que como *graduate* es la clase que mejor clasifica con un acierto considerable, seguida de cerca por dropout y muy alejada de enrolled, la cual obtiene unos resultados pésimos.

Para seguir el análisis, observaremos detenidamente la matriz de confusión (Figura 2.6) y la curva de ROC (Figura 2.7).

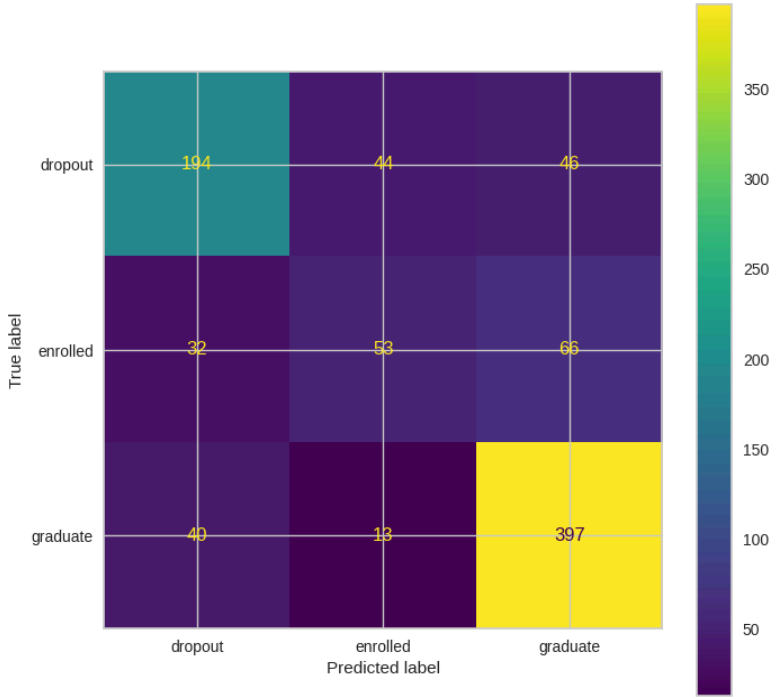

Figura 2.6. Matriz de Confusión NB

El resultado óptimo para esta matriz sería que todos los ejemplos estuvieran en la diagonal, la cual representa los aciertos en las predicciones. Vemos cómo *graduate* obtiene muchos aciertos, aunque hay que tener en cuenta que también existen más instancias de esta clase. Otro aspecto a valorar es el lamentable acierto que tenemos para *enrolled*.

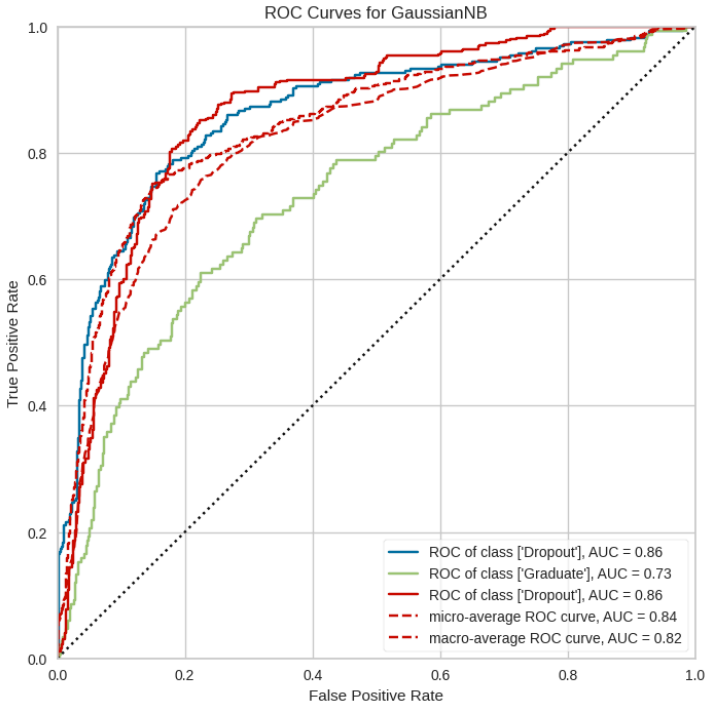

Figura 2.7. Curvas ROC para NB

Como nuestra clase objetivo puede ser clasificada en tres clases, existirán tres curvas ROC. Cada clase se trata de forma individual y se combinan las otras dos para compararlas. Concretamente, en esta gráfica se busca maximizar el área inferior a la curva para cada clase. Como se ha comentado anteriormente, los resultados para *enrolled* son realmente malos, lo cual nos puede hacer sospechar que quizás no sea posible predecir correctamente el valor de esta variable con el dataset actual. Resolveremos esta duda probando otros modelos.

Regresión Logística

El segundo modelo evaluado es Regresión Logística. Como antes, miraremos validación cruzada (Figura 2.8) y el *clasification report (*Figura 2.9).

```
Cross Validation: 0.7584057553496263
```

Figura 2.8. Validación Cruzada LR

Se observa como la validación cruzada un poco y ya supera el 0.75.

	precision	recall	f1-score	support
dropout	0.74	0.80	0.77	263
enrolled	0.38	0.53	0.44	107
graduate	0.93	0.82	0.87	515
accuracy			0.78	885
macro avg	0.68	0.72	0.69	885
weighted avg	0.81	0.78	0.79	885

Figura 2.9. Clasification report LR

Por otro lado, vemos como los valores de *clasification report* son bastante similares a los anteriores con alguna leve mejora. Aun así, *enrolled* sigue siendo extremadamente bajo.

A continuación, exploraremos los hiperparámetros de este modelo y observamos que ninguno obtiene una mejora que haya que destacar.

params	mean_test_score	rank_test_score
{'C': 1000.0, 'penalty': 'none'}	0.758971	1
{'C': 7.943282347242813, 'penalty': 'none'}	0.758971	1
{'C': 3.981071705534969, 'penalty': 'none'}	0.758971	1
{'C': 3.981071705534969, 'penalty': 'l2'}	0.758971	1
{'C': 1.9952623149688788, 'penalty': 'none'}	0.758971	1

Figura 2.10. Hiperparámetros de LR

Además, miraremos el peso que les otorga el modelo a las variables. Lo primero que destaca es que no hay una variable que predomine para *enrolled (1).* Esta puede ser una de las razones de porque predice tan mal esta clase.

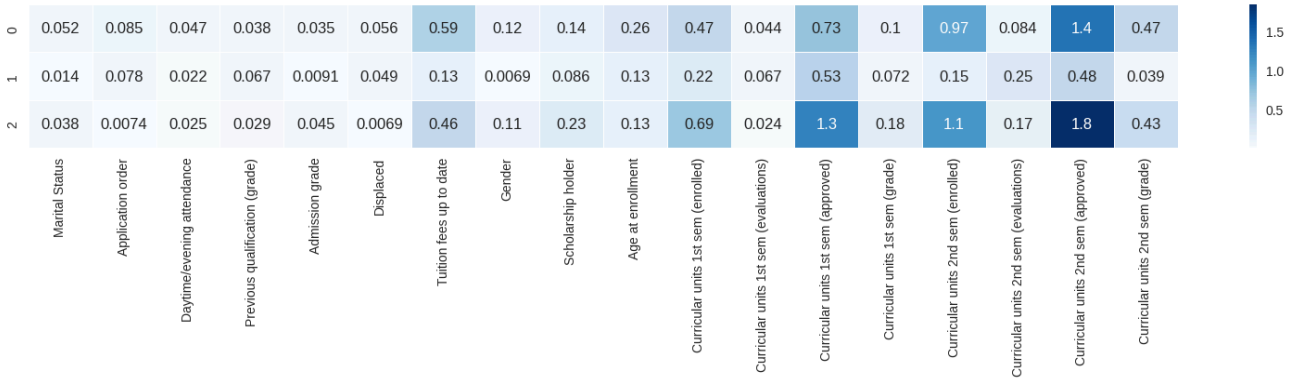

Figura 2.11. Peso de variables LR

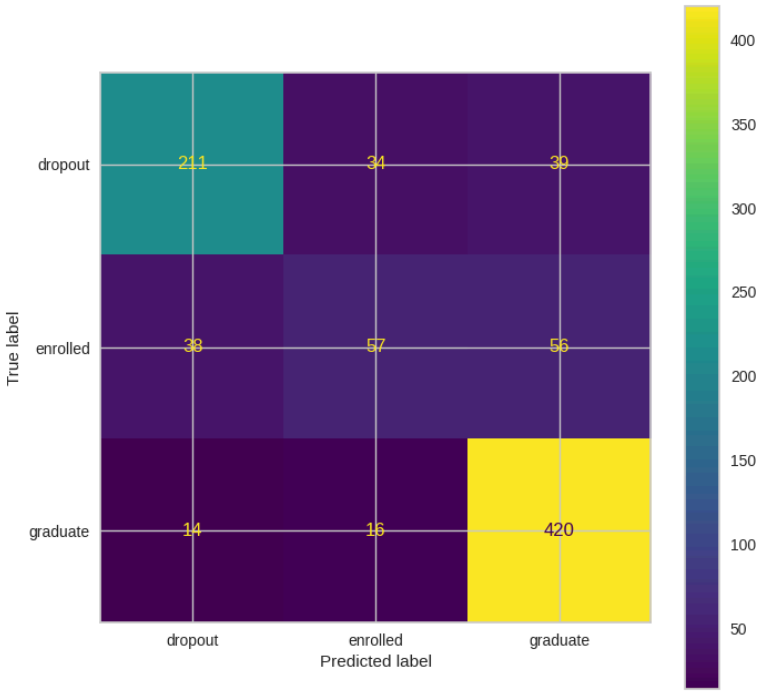

Figura 2.12. Matriz de Confusión LR

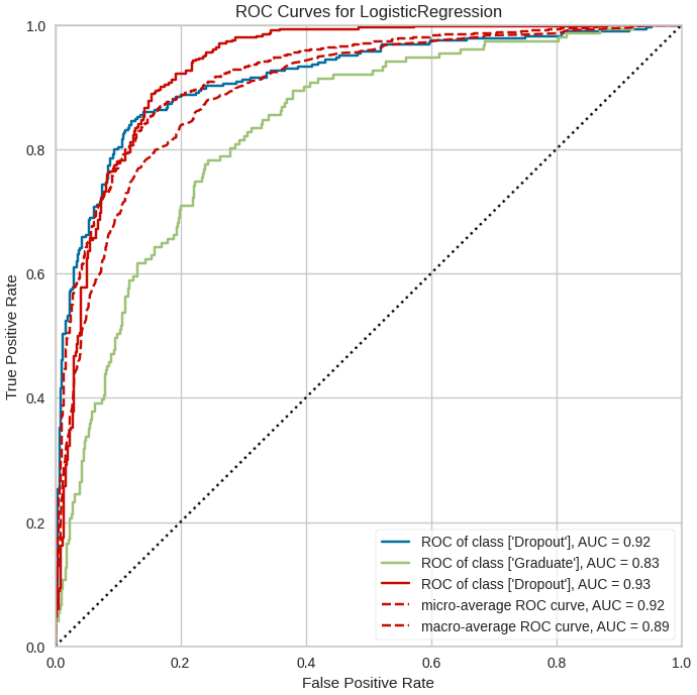

Figura 2.13. Curvas de ROC LR

En la Matriz de Confusión (Figura 2.12) vemos como si ha habido una mejora respecto a NB en *dropout* de 17 aciertos más, en *enrolled* 4 más y en *graduate* 23 más. La curva de ROC (Figura 2.13) también es mejor como es lógico.

LDA

El último modelo lineal que probaremos será LDA. Obtenemos un valor de validación cruzada muy parecido a los modelos anteriores (Figura 2.14).

```
Cross Validation: 0.7499247771322481
```

Figura 2.14. Validación Cruzada para LDA

Como se muestra en la Figura 2.15, los pesos para cada variable son bastante parecidos a los del apartado anterior. Parece que ahora las variables que se repartían el peso se han concentrado en una sola "Curricular unit 2nd sem(Approved)".

Figura 2.15. Pesos de variables LDA

Además, también encontramos algún cambio en el *clasification report* (Figura 2.16), ya que ahora se ha empeorado *dropout* a costa de mejorar *graduate* y *enrolled*, ligeramente.

	precision	recall	f1-score	support
dropout	0.67	0.87	0.75	218
enrolled	0.40	0.50	0.44	120
graduate	0.96	0.79	0.86	547
accuracy			0.77	885
macro avg	0.67	0.72	0.69	885
weighted avg	0.81	0.77	0.78	885

Figura 2.16. Clasification report LDA

Por último, si echamos un vistazo a la Matriz de Confusión (Figura 2.17) y a la curva de ROC (Figura 2.18) para este modelo, vemos como se plasman estos ligeros cambios en la clasificación otorgada por el modelo.

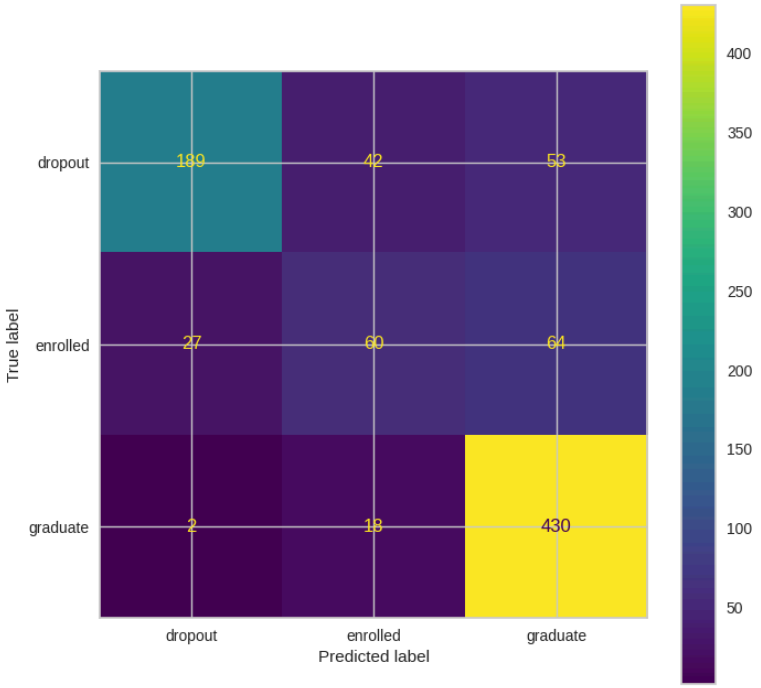

Figura 2.17. Matriz de Confusión LDA

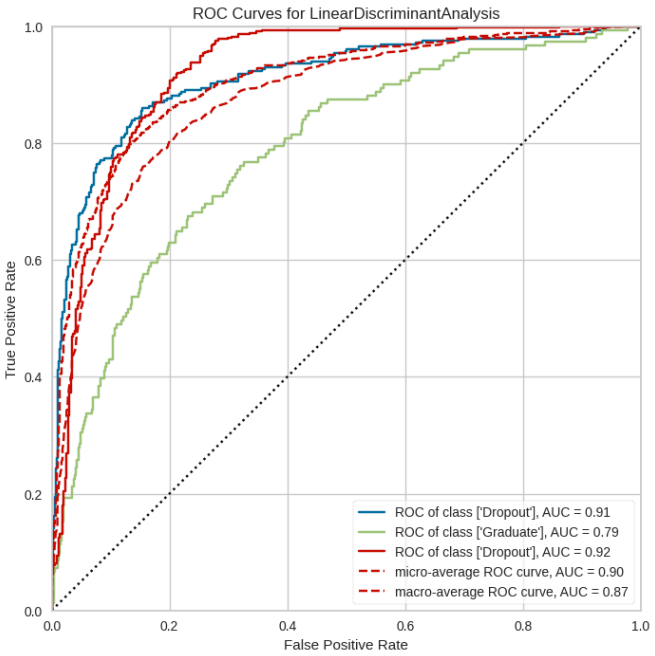

Figura 2.18. Curvas de ROC LDA

Métodos no lineales

Además de los métodos ya descritos en el apartado anterior, usaremos otros 3 que no son lineales. Los escogidos son: Random Forest, SVM con kernel RBF y Gradient Boosting. Primeramente, hemos escogido Random Forest, ya que es perfecto para un problema de clasificación, Gradient Boosting porque sigue la misma idea que Random Forest, por tanto también es adecuado para este problema, y Máquinas de Soporte Vectorial para probar otro modelo que no sea tan orientado específicamente a esta clase de problemas.

Para los métodos no lineales hay que elegir una serie de hiperparámetros que serán los que determinen la forma en la que el modelo se entrena. En los 3 modelos haremos una búsqueda bayesiana para encontrarlos. Hemos decidido utilizar *BayesSearch* en lugar de *GridSearch* porque esta segunda se demoraba mucho en los modelos más costosos computacionalmente (por ejemplo gradient Boosting y SVM), además de no proporcionar resultados muy distintos a los de la búsqueda Bayesiana.

Los parámetros básicos usados para la búsqueda bayesiana son:

- **40 iteraciones**. No hemos visto una diferencia significativa entre 20 y 50 iteraciones, así que hemos determinado un punto medio coherente para evitar el sobreajuste del modelo.
- **cv de 5**. Al probar el cv de 10 la búsqueda tardaba mucho más (probablemente debido a la gran cantidad de instancias de datos y la complejidad de algunos modelos, especialmente SVM y Gradient Boosting) y no daba resultados muy distintos a la de 5. Aunque pudieran haber sido mejores, una vez más no queremos arriesgarnos a que haya sobreajuste.

Random Forest

Como ya se ha explicado, este modelo es perfecto para el tipo de problema que estamos abordando, así que tenemos mucha fe en él.

Los resultados del *BayesSearch* para el *Random Forest* son los siguientes:

```
OrderedDict([('criterion', 'entropy'), ('max_depth', None),
('min_samples_leaf', 1), ('min_samples_split', 2),
('n_estimators', 130)])
```

Figura 3.1. Mejores hiperparámetros para *Random Forest* según *BayesSearch*

Comentando los resultados más relevantes, podemos ver que el número más óptimo de árboles para hacer el bosque son 130 (de los 10 a 200 introducidos), es un número que podríamos considerar promedio y esperable, no muy grande para el sobreajuste de los datos ni muy pequeño para dudar de los resultados. La decisión de no tener una profundidad limitada de los árboles puede hacer que sean más diversos y por tanto que "piensen mejor" en conjunto de mente colmena. Y por último, la decisión del criterio de entropía tiene sentido ya que (sin tener en cuenta el coste computacional de esta elección) las clases objetivo están desequilibradas.

Después del costoso entrenamiento (con un tiempo aproximado de 30 minutos), la precisión del modelo es de **0.7954802259887006**. Podemos ver más detalles en el *clasification report* (Figura 3.2):

	precision	recall	f1-score	support
dropout	0.83	0.76	0.79	284
enrolled	0.58	0.44	0.50	151
graduate	0.83	0.94	0.88	450
accuracy			0.80	885
macro avg	0.75	0.71	0.72	885
weighted avg	0.79	0.80	0.79	885

Figura 3.2. Clasification Report del Random Forest

Se observa una mejoría notable en la precisión de "enrolled" respecto a los métodos lineales (de ~0.3 a ~0.6) sin un empeoramiento grave en las demás clases. Esto es una muy buena señal ya que hemos encontrado un modelo capaz de hacer un trabajo

decente en la clase que más nos costaba predecir. Para visualizar mejor estos datos podemos ver la matriz de confusión y la curva de ROC:

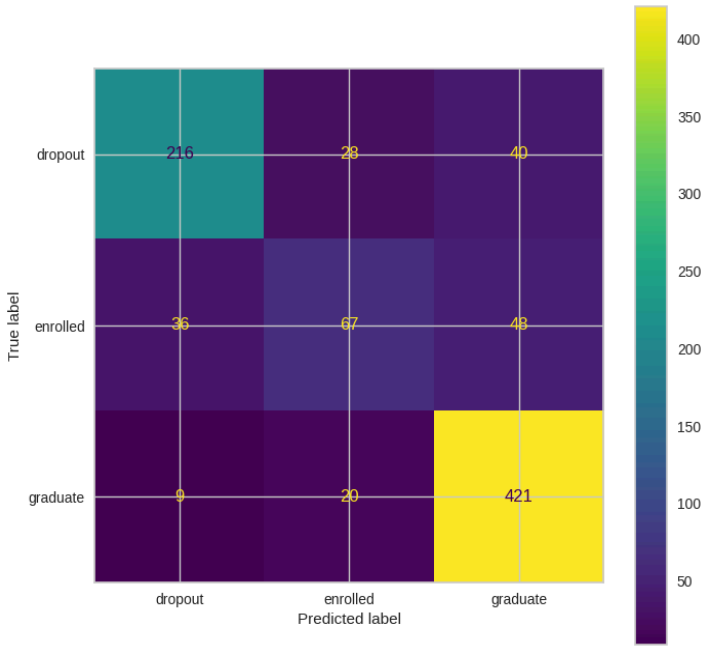

Figura 3.3. Matriz de Confusión Random Forest

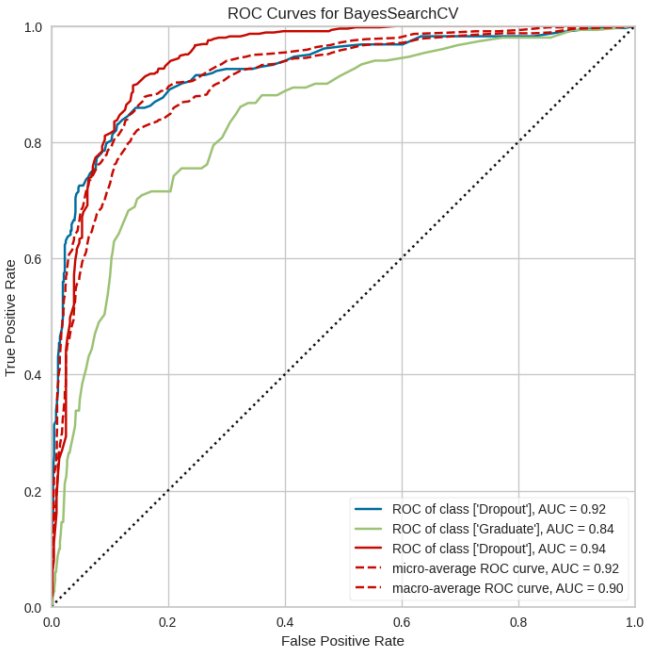

Figura 3.4. Curvas de ROC Random Forest

Observando más detalladamente la matriz de confusión (Figura 3.3) podemos ver que la mejora de precisión de la clase “enrolled” respecto a los demás modelos viene de que éste ha clasificado a menos instancias en esta clase, por eso tenemos menos precisión en el resto de clases y más en ésta, así que no hemos solucionado del todo la complicación de predecir esta clase.

También podemos apreciar claramente (en la Figura 3.4) que la curva de ROC de la clase enrolled tiene más área ya que su línea es mucho más vertical al principio y un pelín más horizontal al final que el resto de modelos usados.

Gradient Boosting

Como el Gradient Boosting es una especie de Random forest que se retroalimenta para aprender, creemos que dará mejores resultados que el modelo del apartado anterior.
Los mejores hiperparámetros según *BayesSearch* son los siguientes:

```
OrderedDict([('criterion', 'friedman_mse'),
('learning_rate', 0.01), ('loss', 'log_loss'), ('max_depth',
None), ('min_samples_leaf', 15), ('n_estimators', 191)])
```

Figura 3.5. Mejores hiperparámetros para *Gradient Boosting* según *BayesSearch*

De la misma manera que *Random Forest*, el que no haya una profundidad de árboles máxima provoca que haya más posibilidades de dar con el correcto. La proporción de aprendizaje tan baja hace que el modelo tarde más en converger, por eso tiene un tiempo de ejecución más elevado y a su vez se requerirán más iteraciones para que se ajuste. Un dato bastante diferente al *Random Forest* que nos ha sorprendido es que la búsqueda Bayesiana haya llegado a la conclusión de que lo mejor es hacer hojas de mínimo 15 instancias, y a la vez nos ha hecho pararnos a pensar en por qué en *Random Forest* recomendaba solo 1, pensamos que es porque en ese caso se suelen dar "mejores resultados" cuanto más diversos sean los árboles, mientras que en *gradient boosting*, al ser retroalimentado, probablemente se requieran más ejemplos de muestra para aprender en la siguiente iteración.

Tras entrenar el modelo (tardando 1 hora y 17 minutos), la precisión de las predicciones en general es de un **0.7875706214689265**. No muy distinta a *Random Forest*. Podemos ver más detalles en el *Classification Report* (Figura 3.6):

	precision	recall	f1-score	support
dropout	0.79	0.75	0.77	284
enrolled	0.58	0.39	0.47	151
graduate	0.82	0.94	0.88	450
accuracy			0.79	885
macro avg	0.73	0.70	0.71	885
weighted avg	0.77	0.79	0.77	885

Figura 3.6. Clasification Report del Gradient Boosting

Podemos apreciar que los resultados son ligeramente peores que el modelo anterior pero aún así muy similares, como esperábamos. Habiendo también mejorado en la clase “enrolled” respecto a los métodos lineales.

Finalmente vemos la matriz de confusión y la curva ROC (Figuras 3.7 y 3.8) de este modelo, también muy similares a *Random Forest*:

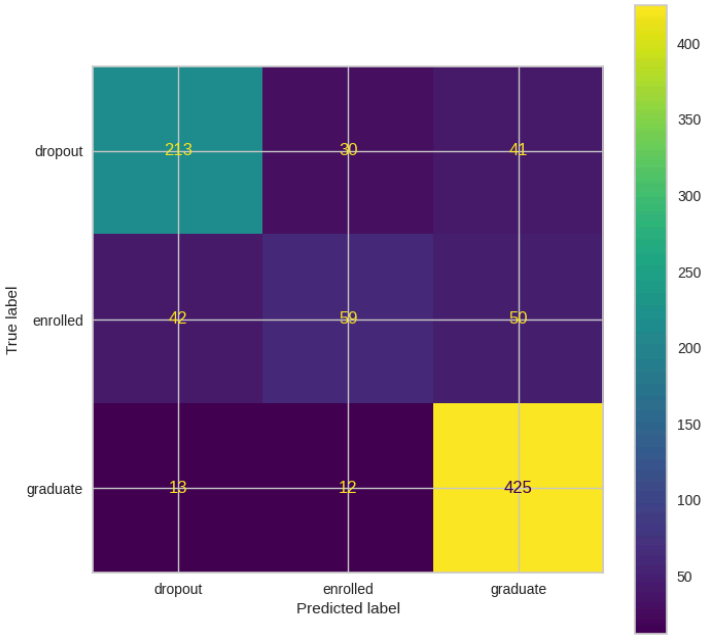

Figura 3.7. Matriz de Confusión Gradient Boosting

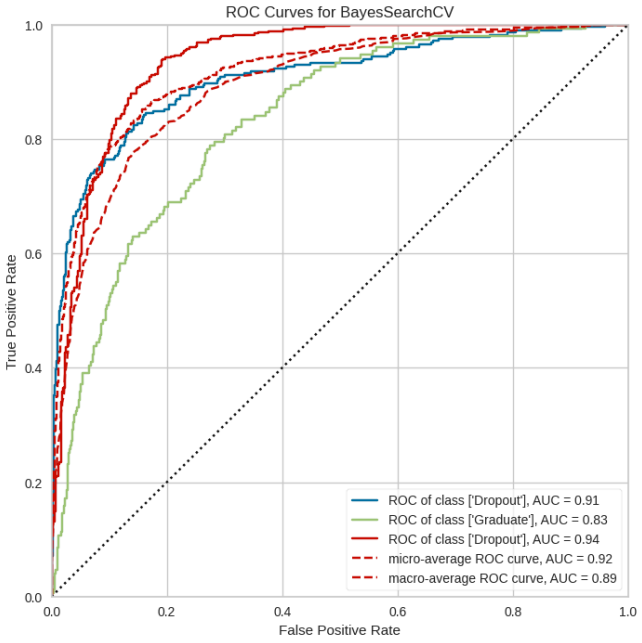

Figura 3.8. Curvas de ROC Gradient Boosting

En definitiva, los resultados de este modelo y *Random Forest* son muy similares y dependen en gran parte del conjunto de datos que se use para resolver este problema.

SVM con kernel RBF

Por último haremos una prueba con el modelo de soporte de máquinas vectoriales, que en líneas generales no está orientado a resolver únicamente problemas de clasificación pero también se puede usar. Hemos decidido hacer este experimento

para ver si hay diferencias notables con los modelos basados en árboles de clasificación. Suponemos que no será tan bueno como éstos últimos pero aún así mejor que los métodos lineales.
Los hiperparámetros que nos ha dado el *BayesSearch* son los siguientes:

```
OrderedDict([('C', 1.7378008287493745), ('gamma', 'scale')])
```

Figura 3.9. Mejores hiperparámetros para *SVM* según *BayesSearch*

El valor del hiperparámetro C otorgado por *BayesSearch* no es ni muy alto ni muy bajo, está por encima de 1, por lo que puede que haya un ligero sobreajuste del modelo a los datos de entrenamiento. Y en cuanto a la elección de gamma "*scale*", probablemente sea debido a que hay algunas variables con más importancia en el entrenamiento que otras, como hemos visto en el análisis previo de los datos.

Tras el entrenamiento del modelo, que resultó ser más corto que los otros dos métodos no lineales (22 minutos), la precisión de las predicciones es de **0.7728813559322034**. Un poco peor que los otros dos métodos no lineales pero aún así no muy descabellado. Para explorar mejor estos resultados podemos ver el *Clasification Report* (Figura 3.10):

	precision	recall	f1-score	support
dropout	0.84	0.71	0.77	284
enrolled	0.50	0.41	0.45	151
graduate	0.81	0.93	0.87	450
accuracy			0.77	885
macro avg	0.71	0.69	0.70	885
weighted avg	0.76	0.77	0.76	885

Figura 3.10. *Clasification report* de SVM con kernel RBF

Se ha mejorado la precisión de la clase dropout a coste de empeorar ligeramente las otras dos. Aún así, la precisión global no se ha visto muy afectada. Se puden ver más detalles en la confusion matrix y las curvas ROC (Figuras 3.11 y 3.12):

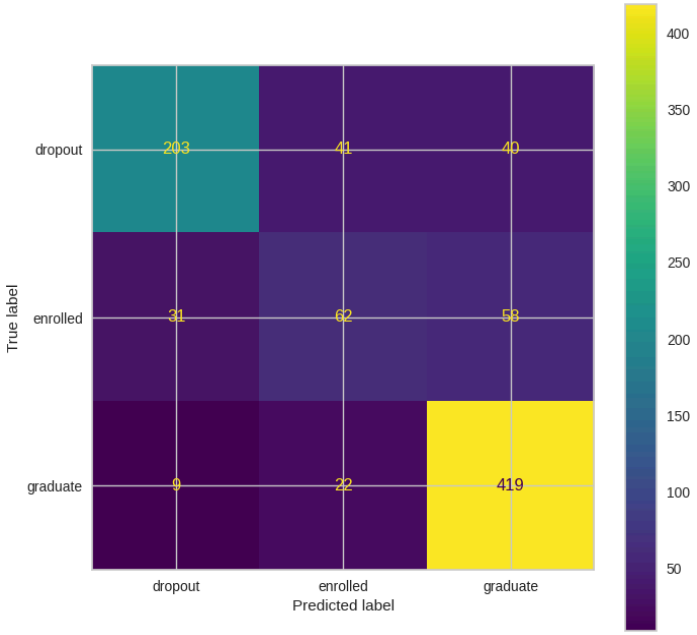

Figura 3.11. Matriz de Confusión SVM con kernel RBF

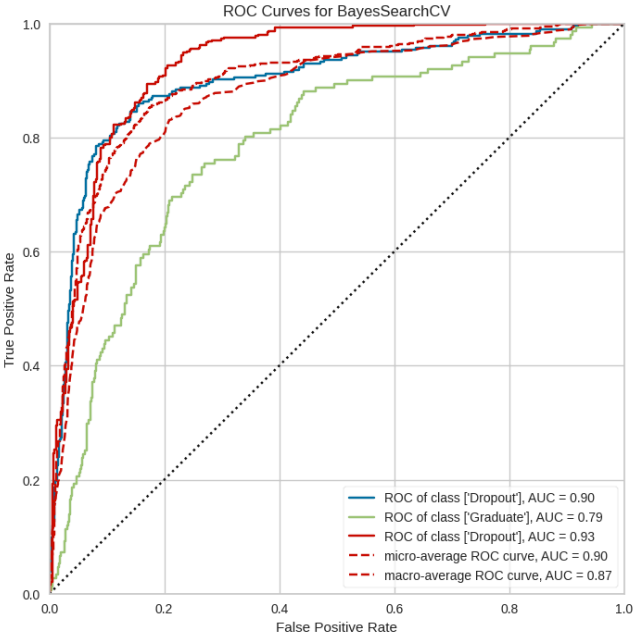

Figura 3.12. Curvas de ROC SVM con kernel RBF

Como era de esperar, no hay muchas diferencias más que las ya comentadas: ligeros cambios en los gráficos y algunos números de más y menos ya comentados en la *Confusion Matrix*.

Elección del mejor modelo

Una vez vistos los resultados de todos los modelos, es hora de elegir el mejor para este problema. Para ayudarnos hemos hecho una tabla con todos los resultados obtenidos (Figura 4.1):

	train acc	test acc	precision score (W)	recall score (W)	f1 score (W)
Random Forest	0.757269	0.795480	0.785547	0.795480	0.786760
Gradient Boosting	0.755295	0.787571	0.773519	0.787571	0.774964
Logistic Regression	0.758406	0.775141	0.759424	0.775141	0.762077
SVM con RBF	0.747957	0.772881	0.764799	0.772881	0.764523
LDA	0.749925	0.767232	0.763241	0.767232	0.755793
Naive Bayes	0.717989	0.727684	0.712841	0.727684	0.716667

Figura 4.1. Resultados totales de todos los modelos

Claramente el peor modelo de todos es el Naive Bayes, con una precisión de aproximadamente 0.72, y los mejores son: *Random Forest*, *Gradient Boosting* y *Logistic Regression*. De entre estos tres, la diferencia de precisión no es grande, así que cualquiera de los tres serviría para resolver el problema en una situación real. Aún así, consideramos que el más útil en este caso es el *Random Forest*, ya que ha demostrado ser el más viable para el problema de clasificación que tenemos delante.

Cabe destacar que, en cuanto a la proporción rendimiento / tiempo, la regresión logística es la más alta (con un tiempo de ejecución de 35 segundos, contra los 33 minutos de *Random Forest* y los 91 minutos de *Gradient Boosting*), así que en un entorno donde no se disponga de hardware muy avanzado o que se tenga tiempo limitado es más recomendable usar un método lineal (preferiblemente el mencionado).

En general, los métodos no lineales dan resultados más buenos con el punto negativo de ser más computacionalmente complejos. De todas formas, seguimos insistiendo en la recomendación de usar Random Forest para este conjunto de datos y objetivo.

Análisis de interpretabilidad

Importancia de los atributos

Para los modelos lineales vemos en las Figuras 2.11 y 2.15 vemos como las variables con más peso son las mismas prácticamente. Entre los 2 modelos cambian muy poco sus valores. Concretamente, las variables que tienen más peso son “Curricular units 1st/2nd sem (approved)”.

Para los modelos no lineales, hemos hecho una serie de experimentos de la importancia de las variables en cada modelo, se pueden observar en las Figuras 5.1 y 5.2:

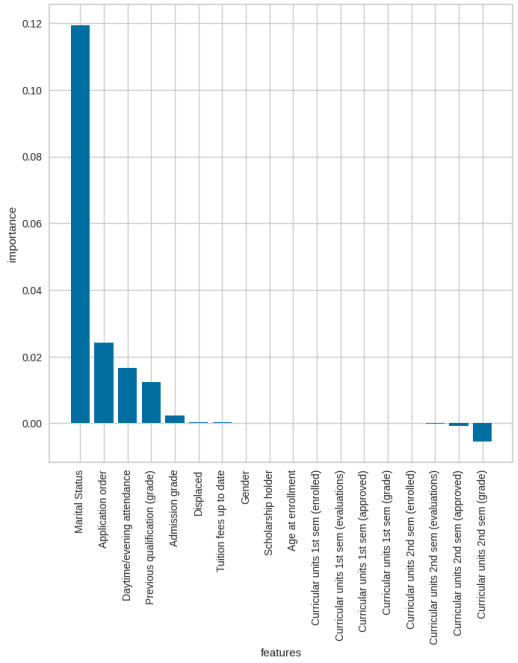

Figura 5.1. Importancia de las variables en Random Forest

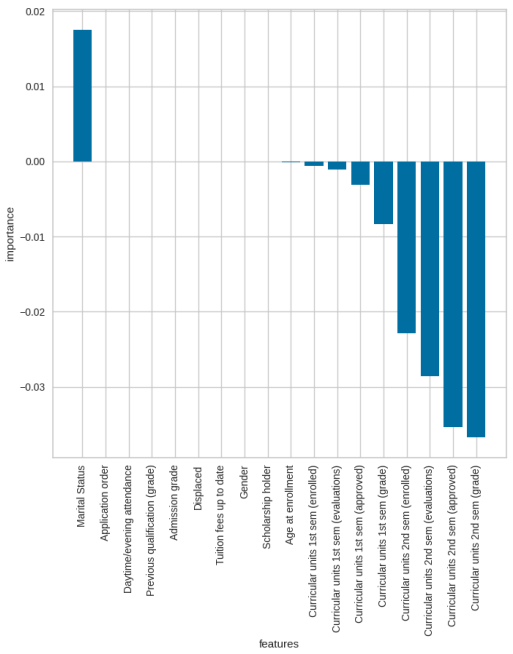

Figura 5.2. Importancia de las variables en Gradient Boosting

Se aprecia que el modelo de gradient Boosting le da más importancia a las variables de unidades curriculares obtenidas en los cursos, más que el Marital Status, completamente inverso de *Random Forest* y los modelos lineales como regresión logística o Naive Bayes. Esto puede deberse a que con la bajísima learning rate el modelo se ha sobreajustado a los datos de ejemplo.

Ejemplos entre modelo lineal y no lineal

Para comparar los ejemplos hemos escogido Regresión Logística y Random Forest. Analizaremos casos extremos de las variables que hemos encontrado que tienen más relevancia para los modelos (Martial Status y Curricular units 1st/2nd sem (approved)).

En primer lugar, para Martial Status obtenemos [2 2 2 2 2] (2 = graduate). Para la máxima "Curricular units 2nd sem" vemos cómo obtenemos "Graduate" lo cual tiene sentido, ya que al tener tanta importancia para el modelo al clasificar para *graduate* es lógico que al ser un extremo estuviera clasificado en 2.

Para los modelos no lineales, vemos algo parecido. Exploramos valores extremos, los cuales deberían dar resultados parecidos por la importancia que les otorga el modelo. En el primer caso, tanto como para Min Martial Status como para el Max Curricular vemos que se clasifica con la clase con más importancia que es "graduate", excepto por un caso.

Complicaciones durante la práctica

Durante la realización de la práctica nos hemos encontrado algunos problemas a afrontar. Lo primero que encontramos fue la cantidad exagerada de variables del dataset. Poco a poco, fuimos reduciendo este número en el preprocesado hasta llegar a tener algo más manejable.

Por otro lado, nos hemos encontrado que los modelos no lineales tardan mucho en ejecutarse. Esto nos ha sorprendido, ya que en las listas de problemas que hemos ido realizando no sucedía tan exageradamente. Cabe decir, que entendemos que al ser un dataset muy grande es lógico que pase y que en el mundo laboral debe ser el pan de cada día. De ahí la importancia de saber preprocesar correctamente los datos y escoger un modelo adecuado.

Por último, un pequeño detalle a comentar es que en algunas curvas de ROC, en vez de aparecer “dropout enrolled graduate” en la leyenda se etiqueta como “dropout graduate dropout” y no corresponde a sus verdaderos valores.

Conclusiones

En esta práctica hemos conseguido predecir con una precisión del 80% los resultados académicos de estudiantes universitarios, consideramos que está bastante bien este resultado ya que es un número muy superior a 33% (el equivalente a elegir una clase objetivo al azar). Hemos aprendido a hacer experimentos de aprendizaje automático con una base científica y analizar los resultados de cada prueba de manera coherente, además de pensar si éstos tienen sentido y se ajustan a lo que queríamos desde un principio.

Al probar con los modelos no tan orientados a los problemas de clasificación y ver que no daban resultados nefastos, nos hemos quedado con la duda de si se podrían haber usado para este problema, es algo que nos gustaría haber probado, pero por falta de tiempo y las limitaciones de la práctica no se ha podido. De todas formas, también hemos aprendido que el modelo a elegir y los métodos y pasos a seguir dependerán del contexto, problema, dataset y entorno concreto en el que se use.

Bibliografía / Webgrafía

1. **Documento del dataset:** M.V.Martins, D. Tolledo, J. Machado, L. M.T. Baptista, V.Realinho. (2021) "Early prediction of student's performance in higher education: a case study" Trends and Applications in Information Systems and Technologies, vol.1, in Advances in Intelligent Systems and Computing series. Springer. DOI: 10.1007/978-3-030-72657-7_16
2. **Estudio previo del dataset:** https://www.mdpi.com/2306-5729/7/11/146
3. **Página de Aprendizaje automático:** https://sites.google.com/upc.edu/aprenentatge-automatic?pli=1&authuser=1