DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

TP Nº1: Aprendizaje Supervisado

Aprendizaje Automático
Segundo Cuatrimestre - 2024

Integrante	LU	Correo electrónico
Jerónimo Barragán	1472/21	barragan.jeronimo123@gmail.com
Manuel Horn	321/21	manuhorn1910@gmail.com
Jeremías Laria Guaza	1329/21	jeremiaslaria7@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1. Ejercicio 1

Se decidió reservar un 90 % de las observaciones totales para desarrollo, y el 10 % restante para evaluación. Esta separación se realizó de manera pseudo-aleatoria según una semilla fija. Además, para que en los datos de evaluación se replique lo mejor posible la distribución original de los datos, chequeamos que la proporción de casos positivos sea la misma en ambos conjuntos: para la semilla utilizada dio aproximadamente 25 % en cada uno, así que nos quedamos con ese criterio de separación.

2. Ejercicio 2

Permutación	Accuracy (training)	Accuracy (validation)	AUPRC (training)	AUPRC (validation)	AUCROC (training)	AUCROC (validation)
1	0.827778	0.711111	0.710734	0.485132	0.797378	0.682885
2	0.847222	0.733333	0.705731	0.451482	0.770786	0.649937
3	0.855556	0.755556	0.758134	0.331357	0.815895	0.600260
4	0.866667	0.733333	0.744012	0.439658	0.850473	0.612380
5	0.852778	0.766667	0.735426	0.446235	0.826428	0.671791
Promedios	0.850000	0.740000	0.730807	0.430773	0.812192	0.643450
Global	(NO)	0.740000	(NO)	0.421786	(NO)	0.645790

Cuadro 1: Resultados de las permutaciones y métricas en entrenamiento y validación.

Las métricas calculadas sobre los conjuntos de entrenamiento son en todos los casos mayores, como es de esperarse, mientras que no ocurre lo mismo sobre los conjuntos de validación.

En validación, la métrica de accuracy es la que mejores resultados arroja. Por el contrario, los peores resultados se obtuvieron por AUPRC. En el caso intermedio, AUCROC arroja valores más altos que AUPRC.

En este problema, estamos interesados en minimizar la cantidad de falsos negativos, lo que se traduce en maximizar el recall. Sin embargo, un AUPRC bajo puede deberse a una precision baja, y a un recall considerablemente bueno. Por lo tanto, no creemos que sea una métrica indicadora de un mal clasificador en este caso.

Por otro lado, el AUCROC da bastante por encima de 0.5, con lo cual tenemos bastante seguridad de que dadas una instancia positiva y una negativa, la positiva tenga más probabilidad de ser clasificada como tal (que es lo que queremos, que no se nos escapen instancias positivas clasificándolas como negativas).

Mirando el Cuadro 2, si fijamos una altura y variamos el criterio de corte, la diferencia obtenida en accuracy calculada en training y en validación tiende a ser mayor cuanto más grande sea el valor de altura fijado. Esto tiene sentido porque al complejizar el modelo, más puede adaptarse al conjunto de entrenamiento y provocarse sobreajuste.

Altura máxima	Criterio de corte	Accuracy (training)	Accuracy (validación)
3	gini	0.850000	0.740000
5	gini	0.938889	0.735556
Infinito	gini	1.000000	0.704444
3	entropy	0.820556	0.748889
5	entropy	0.917778	0.682222
Infinito	entropy	1.000000	0.671111

Cuadro 2: Resultados de Accuracy para diferentes criterios de corte y alturas máximas.

3. Ejercicio 3

En cada modelo realizamos 200 iteraciones, eligiendo combinaciones de parámetros al azar de los espacios de búsqueda determinados, y mediante validación cruzada con 5 folds evaluamos el rendimiento. Intentamos elegir hiperparámetros influyentes que no estén tan relacionados entre sí, es decir, queremos que todos los hiperparámetros utilizados restrinjan al modelo y no a otros hiperparámetros, en el sentido de evitar que un hiperparámetro no pueda modificar el modelo, porque su propiedad esté restringida por otro. Así evitamos que RandomSearchCV pruebe combinaciones que no tengan diferencias reales en rendimiento.

En mean_test_score reportamos, para cada modelo, la performance promedio tras hacer validación cruzada, usando AUCROC como métrica.

3.1. Árboles de Decisión

- **Criterio**: Probamos con todos los posibles criterios de selección de atributos para dividir el árbol en cada nodo.
- **min_samples_split**: El mínimo número de observaciones que tienen que caer en un nodo para que el árbol se pueda volver a dividir. No depende del criterio. Idealmente creemos que este parámetro debería ser medianamente grande, para asegurarnos tener hojas con una cantidad aceptable de observaciones, y no acercarnos al caso extremo de sobreajuste en que cada observación de entrenamiento tiene su propia hoja.
- **max_leaf_nodes**: El máximo número de hojas que puede tener el árbol. No depende del criterio de selección de atributos, pero si tomamos un valor de $min\_samples\_split$ muy grande, entonces el árbol no va a poder crecer tanto en altura, y considerar valores altos de max_leaf_nodes puede no tener sentido (sin embargo en la práctica, obtuvimos combinaciones de hiperparámetros poco repetitivas en rendimiento).
- **min_samples_leaf**: El mínimo número de observaciones que tienen que caer en un nodo para que pueda ser una hoja. La idea es probar con valores medianamente grandes, para evitar sobreajuste (es decir, tener una hoja por cada observación de entrenamiento). No depende del criterio, pero no tiene sentido probar valores de $min\_samples\_split$ y $min\_samples\_leaf$ muy grandes, porque pueden llevar a subajuste (obteniéndose árboles muy poco ramificados). Tampoco tiene sentido probar valores de $min\_samples\_leaf$ y max_leaf_nodes muy altos, por la misma razón. En la práctica igualmente, obtuvimos combinaciones poco repetitivas en cuanto a rendimiento.

El espacio de busqueda utilizado sobre estos hiperparámetros fue:

- **Criterio** $\in \{gini, entropy, log\_loss\}$
- **min_samples_split** $\in \{100, ..., 500\}$
- **max_leaf_nodes** $\in \{2, ..., 16\}$
- **min_samples_leaf** $\in \{10, ..., 100\}$

Criterio	Min Samples Split	Max Leaf Nodes	Min Samples Leaf	mean_test_score	Rank
entropy	146	12	69	0.725847	1
entropy	108	4	75	0.725263	2
entropy	111	5	66	0.723764	3

Cuadro 3: Top 3 modelos de Árbol de Decisión.

3.2. KNN

- **n_neighbors**: La cantidad de vecinos a considerar. Probamos con valores medianamente bajos de vecinos en relación a la cantidad de datos de entrenamiento, para evitar que los vecindarios de cada observación sean los mismos (o sea, prácticamente el conjunto de entrenamiento entero). Si pusiéramos valores muy bajos, estaríamos sobreajustando, y con valores muy altos, subajustando.
- **weights**: Probamos darle mayor peso en la decisión de clasificación a los vecinos que están más cerca, o tomar pesos uniformes independientemente de la distancia (básicamente todas las opciones posibles).
- **algorithm**: Probamos cambiar el algoritmo para computar los vecinos más cercanos, entre los disponibles.
- **metric**: Probamos medir distancias con distintas métricas, siempre y cuando sean compatibles con la estructura de las observaciones.

El espacio de busqueda utilizado sobre estos hiperparámetros fue:

- **n_neighbors** $\in \{1, ..., 4\}$
- **weights** $\in \{uniform, distance\}$
- **algorithm** $\in \{auto, ball\_tree, kd\_tree, brute\}$
- **metric** $\in \{cityblock, euclidean, manhattan\}$

n_neighbors	weights	algorithm	metric	mean_test_score	Rank
27	distance	kd_tree	cityblock	0.861110	1
24	distance	kd_tree	manhattan	0.860409	2
23	distance	kd_tree	manhattan	0.859883	3

Cuadro 4: Top 3 modelos de KNN.

(*) En la posición 2 obtuvimos un empate, podríamos haber puesto también el modelo con parámetros {'weights': 'distance', 'n_neighbors': 24, 'metric': 'cityblock', 'algorithm': 'brute'}, pero decidimos quedarnos con uno sólo de ellos por simplicidad.

3.3. SVM

- Variamos la constante **C** para regular la cantidad de observaciones que pueden caer entre el hiperplano separador y el margen. Como a priori obtuvimos un buen rendimiento por AUCROC, dejamos el rango que encontramos.
- También probamos separar los datos con distintos **kernel**s, resultando que los mejores resultados se encontraban con el rbf. Por lo tanto, decidimos probar con distintos valores del parámetro **gamma**.

El espacio de busqueda utilizado sobre estos hiperparametros fue:

- **C** $\in \{1, ..., 20\}$
- **kernel** $\in \{linear, poly, rbf, sigmoid\}$
- **gamma** $\in \{1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1\}$

C	kernel	gamma	mean_test_score	Rank
14	rbf	0.0001	0.910543	1
13	rbf	0.0001	0.910403	2
18	rbf	0.0001	0.910038	3

Cuadro 5: Top 3 modelos de SVM.

3.4. LDA

LDA asume normalidad e iguales matrices de covarianza por clase, por lo que está haciendo supuestos muy fuertes de los cuales no tenemos garantía que sean ciertos. En caso de que conozcamos una buena aproximación de las priors, se lo podríamos pasar al modelo, pero en este caso no contamos con esa información.

Para los hiperparametros por defecto que utiliza el algoritmo de SK-Learn, obtenemos para los 5 folds un score promedio de: 0.6301

3.5. Gaussian Naive Bayes

Por otro lado, Gaussian Naive Bayes asume normalidad e independencia entre las covariables, y en este problema probablemente no lo sean. Al igual que en LDA, podríamos pasarle al modelo las priors de cada clase si tuviéramos información sobre ellas. Es razonable entonces que estos modelos sean los que peor hayan dado, ya que tienen un alto sesgo inductivo.

Para los hiperparametros por defecto que utiliza el algoritmo de SK-Learn, obtenemos para los 5 folds un score promedio de: 0.6807

3.6. Mejor modelo

Al tener 200 covariables y 450 observaciones de entrenamiento, no sólo tenemos pocos datos, sino que además Árboles de Decisión, que ya de por sí es un modelo con alta varianza, al tener tantas características y tan pocas observaciones, es de esperar que no generalice bien porque se sobreajusta más que otros modelos. Por otro lado, KNN y SVM tienen ambos menor varianza, lo cual los beneficia en el contexto de este problema, y terminan dando resultados similares en rendimiento.

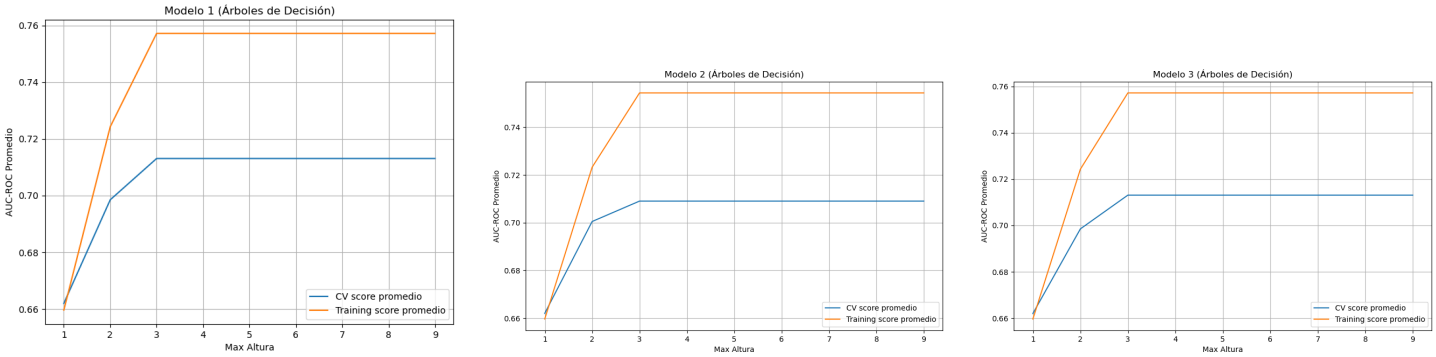
A KNN puede afectarle más las correlaciones espurias entre algunas de las covariables, ya que podría identificar como "cercanas" a instancias que pertenecen a clases diferentes, pero que son similares en covariables que no son explicativas para nuestro modelo. Para realizar un análisis exhaustivo de esto, podríamos evaluar la importancia de nuestras covariables mediante el método de permutación de características, que consiste en permutar los valores de una covariable y volver a evaluar el modelo para determinar si esa covariable es realmente explicativa.

Por otro lado, SVM es menos sensible a las correlaciones espurias entre las variables, ya que al proyectar los datos en dimensiones superiores, no es tan evidente cómo estas relaciones pueden influir en la clasificación. En cuanto a la configuración óptima, el hiperparámetro degree no tiene relevancia porque el kernel utilizado es RBF. Además, el valor de C es el más alto entre los tres mejores resultados, aunque en los diez mejores resultados se mantuvo entre 4 y 6, a pesar de que el espacio de búsqueda abarcó valores de C entre 1 y 20.

4. Ejercicio 4

4.1. Curvas de complejidad

Árboles de Decisión



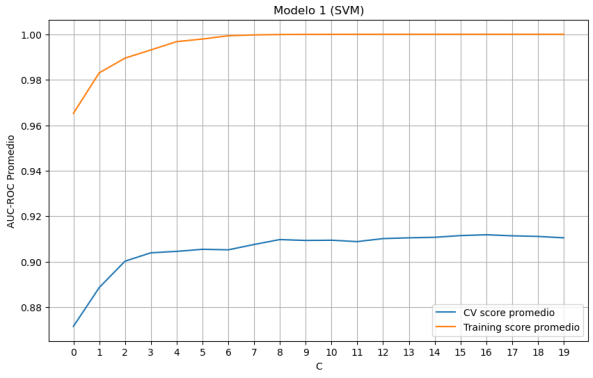
(a) Modelo 1 de ADD (b) Modelo 2 de ADD (c) Modelo 3 de ADD

Figura 1: Curvas de complejidad variando *max_depth* para los tres mejores modelos de Árboles de Decisión (ADD).

Al variar la altura máxima, vemos que la varianza del modelo aumenta dado que las curvas de AUCROC en training y en validación (promedios) se alejan. Sin embargo, no se alejan tanto: se ve que la altura máxima óptima es 3, y no hay cambios a partir de ese valor. Esto puede deberse a la poca cantidad de observaciones que tenemos para la cantidad de features en nuestros datos.

En cuanto al sesgo, diríamos que es bajo pero no tanto, ya que los valores obtenidos de AUCROC no son malos, pero ni tan buenos, en ambas curvas.

SVM



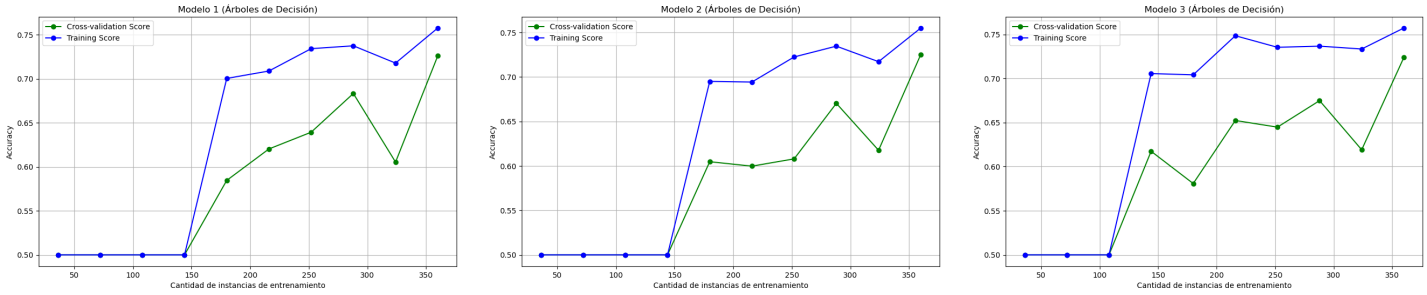
(a) Modelos 1, 2 y 3 de SVM

Figura 2: Curvas de complejidad variando C para los tres mejores modelos de SVM (resulta la misma para los 3 modelos).

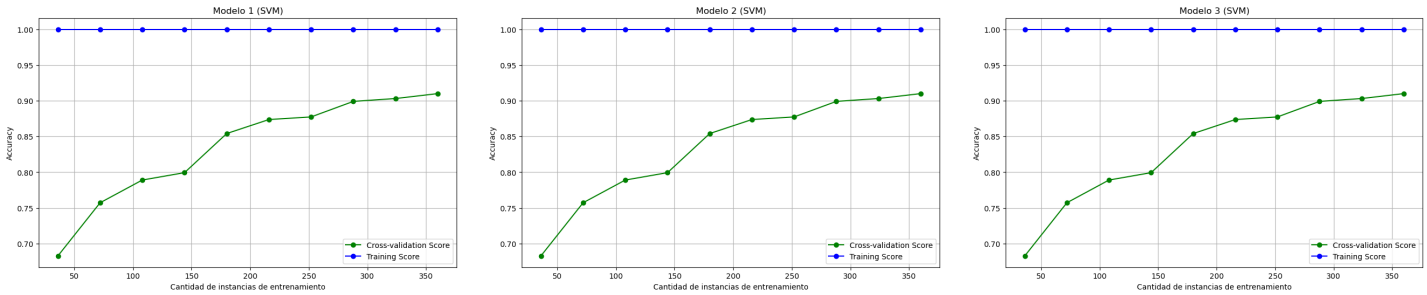
Para los tres mejores modelos se obtuvieron exactamente las mismas curvas de complejidad porque coinciden en todos los hiperparámetros excepto en el que se varía en el gráfico.

Variar el parámetro C reduce la varianza del modelo, ya que la distancia entre las curvas de AUCROC se acorta, aunque no significativamente. Por otro lado, en cuanto al sesgo, pareciera ser mucho más bajo que con árboles, ya que los valores de AUCROC son bastante más grandes para ambas curvas.

4.2. Curvas de aprendizaje



(a) Modelo 1 de ADD (b) Modelo 2 de ADD (c) Modelo 3 de ADD



(d) Modelo 1 de SVM (e) Modelo 2 de SVM (f) Modelo 3 de SVM

Figura 3: Curvas de aprendizaje de ADD y SVM

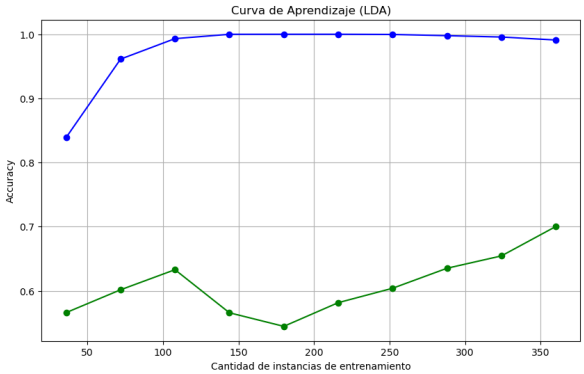


Figura 4: Curva de aprendizaje de LDA

En ningún caso parece haberse alcanzado el límite porque las curvas siguen comportamientos crecientes considerablemente pronunciados para valores altos del tamaño del training set.

4.3. Random Forest

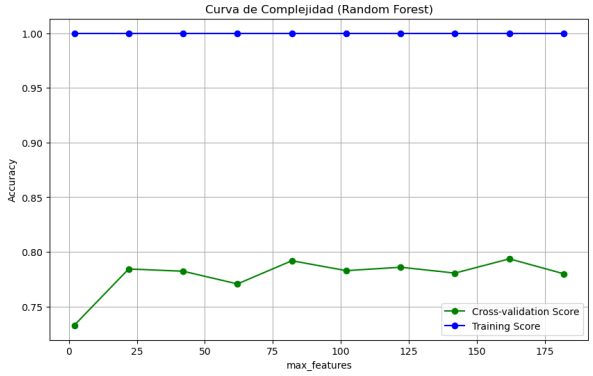


Figura 5: Curva de complejidad de Random Forest con 200 árboles variando max_features

Observamos que variar el hiperparámetro max_features no provoca cambios significativos en accuracy. Recordemos que este hiperparámetro regula la cantidad máxima de atributos elegidos al azar en cada nodo para determinar cuál separa mejor los datos. Esto puede deberse a que no haya atributos que separen mejor a los datos que otros, sino que la importancia de cada uno en la clasificación sea relativamente equitativa.

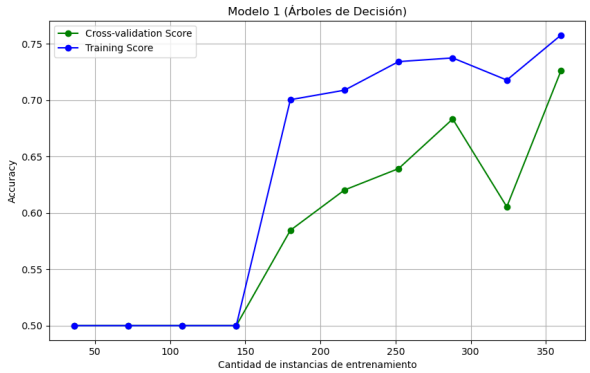


Figura 6: Curva de aprendizaje de RF

Nuevamente parecería que recolectar más datos sea útil para mejorar la performance del modelo.

5. Ejercicio 5

Elegimos el mejor modelo según AUCROC, de todos los entrenados en el ejercicio 3: resultó ser el mejor modelo de SVM. Luego, evaluamos la generalización del mismo con el held-out que separamos al principio del trabajo, obteniendo un AUCROC de aproximadamente 0,9408: esta es la estimación del AUCROC resultante en datos no vistos.

6. Ejercicio 6

En conclusión, los modelos que generalmente dan mejores resultados, como Random Forest, no fueron los mejores para el problema en cuestión. A lo largo del trabajo, lo más desafiante nos pareció fijar las grillas

para Random Search Cross Validation, ya que no es tan fácil elegir los mejores valores a ser sampleados. Así que esto fue más experimental que metódico, probando repetidas veces varias combinaciones con prueba y error.