Informe de Aprendizaje Automático

Chavez, Mauro
a@gmail.com

Lewkowicz, Iván
a@gmail.com

Drelewicz, Santiago
a@gmail.com

Torrez, Matías
matiastorrez157@gmail.com

Culaciati, Dante
a@gmail.com

26 de abril de 2025

Índice

1. Ejercicio 1

En este inciso se pide separar los datos en conjuntos de entrenamiento y evaluación, donde no se debe utilizar la libreria `train_test_split` de `sklearn`.

Primero se realizo una exploracion de los datos, donde se observa que el dataset posee 200 features, todas numericas, y 500 filas. Se observa que el dataset no tiene valores nulos y que ademas se trata de un problema desbalanceado, donde el %70 de los datos pertenecen a la clase 1 y el % restante pertenece a la clase 0, por lo que no es necesario realizar un preprocesamiento de los datos. Se decide entonces utilizar el 80 % de los datos para entrenamiento y el 20 % restante para evaluacion.

Como la proporción de los datos es desbalanceada, realizamos un `stratified split` en la separación de los datos, procurando mantener la proporción del dataset original para los datos de entrenamiento y evaluación.

2. Ejercicio 1.1

3. Ejercicio 2

Para la primera parte de este ejercicio, entrenamos un arból de decisión con altura máxima 3 y estimamos la performance del modelo con K fold cross validation para distintas métricas. Las metricas utilizadas son *Accuracy*, *AUPRC* y *AUC ROC* y se realizo un *K-fold* con $K = 5$.

En la tabla 1 se muestran los resultados obtenidos para cada una de las 5 permutaciones de los datos, asi como el promedio de cada métrica para todas las permutaciones y el resultado global, el cual se obtiene al calcular las metricas utilizando el conjuto de predicciones formado a partir de concatenar las predicciones de cada fold.

Permutación	**Accuracy** (training)	**Accuracy** (validación)	**AUPRC** (training)	**AUPRC** (va
1	0.8125	0.6375	0.6710	0.322
2	0.840625	0.5875	0.7337	0.333
3	0.825	0.6875	0.6431	0.343
4	0.81875	0.7	0.6573	0.362
5	0.84375	0.65	0.6958	0.414
Promedios	0.828125	0.6525	0.6802	0.355
Global	(NO)		(NO)	

Cuadro 1: Resultados por permutación y métricas

Se observa que este modelo presenta un buen desempeño en el conjunto de entrenamiento, pero su desempeño en el conjunto de validación es bastante bajo, lo que podría indicar que el modelo está sobreajustado a los datos de entrenamiento.

Para la segunda parte del ejercicio, se exploraron diferentes combinaciones de hiperparámetros para el modelo de árbol de decisión, utilizando `GridSearchCV` de `sklearn`. Se probaron diferentes valores para la profundidad máxima del árbol y el cirterio de corte. Se utilizó `StratifiedKFold` con $K = 5$ para la validación cruzada. En la tabla 2 se muestran los resultados obtenidos para cada combinación de hiperparámetros, así como el promedio de *Accuracy* para cada combinación.

Altura máxima	**Criterio de corte**	**Accuracy** (training)	**Accuracy** (validación)
3	Gini	0.6375	0.6710
5	Gini	0.5875	0.7337
Infinito	Gini	0.6875	0.6431
3	Entropía	0.7	0.6573
5	Entropía	0.65	0.6958
Infinito	Entropía	0.828125	0.828125

Cuadro 2: Resultados por permutación y métricas

4. Ejercicio 3

En esta seccion se exploraron diferentes combinaciones de hiperparametros para los modelos de arboles de decision, *KNN* y *SVM* y se los comparo contra los algoritmos *LDA* y *GaussianNB* sin realizar una busqueda de hiperparametros . Se buscó identificar el mejor modelo de cada familia de algoritmos buscando maximizar el *AUC ROC*.

Para realizar la busqueda de hiperparametros y la estimacion del rendimieinto de los modelos se utilizo la tecnica de *Nested Cross Validation*, utilizando *RondomizedSearchCV* para la busqueda de hiperparametros y, al igual que en la seccion anterior, se utilizo *StratifiedKFold* para la creacion de los *Kfolds*. *Nested Cross Validation* es una técnica que permite evaluar el rendimiento de un modelo de aprendizaje automático mientras se optimizan sus hiperparámetros. En este enfoque, se utilizan dos bucles de validación cruzada: uno externo para evaluar el rendimiento del modelo y otro interno para ajustar los hiperparámetros. Esto ayuda a evitar el sobreajuste y proporciona una estimación más precisa del rendimiento del modelo en datos no vistos, en la figura 1 se muestra un esquema de como se realiza este proceso.

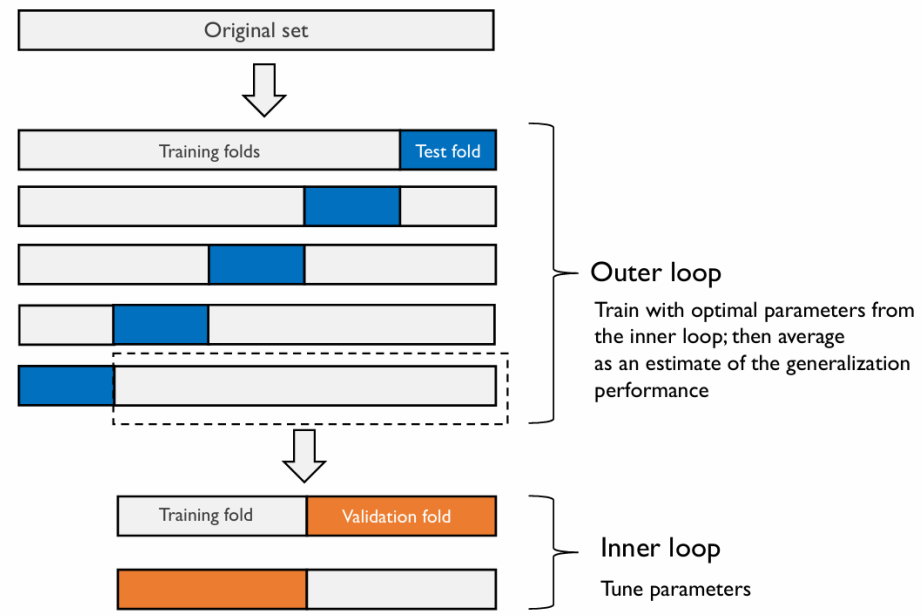

Figura 1: Esquema de *Nested Cross Validation*, donde el bucle interno se utiliza para la búsqueda de hiperparámetros y el bucle externo para la evaluación del rendimiento.

4.1. Arbol de decisión

A continuación se detallan los hiperparámetros utilizados junto con sus respectivos valores de prueba:

- `max_depth` (int o Ninguno): Altura máxima del árbol, mientras mas profundo sea el árbol, más complejo será el modelo y habra mas posibilidades de sobreajusta.
- `criterion` (gini, entropia): Criterio de corte,
- `splitter`: Estrategia de división, se probaron los valores `best` y `random`.
- `min_samples_leaf`: Cantidad mínima de muestras por hoja, se probaron los valores 1, 2 y 3.
- `min_samples_split`: Cantidad mínima de muestras para dividir un nodo, se probaron los valores 2, 3 y 4.
- `class_weight`: Peso de las clases, se probaron los valores `balanced` y `None`.

4.2. *KNN*

A continuación se detallan los hiperparámetros utilizados junto con sus respectivos valores de prueba:

- `n_neighbors`: Cantidad de vecinos a considerar, se probaron valores aleatorios en el rango de 3 a 50.
- `weights`: Estrategia de ponderación, se probaron los valores `uniform` y `distance`.
- `metric`: Métrica de distancia, se probaron las métricas `euclidean`, `manhattan`, `minkowski` y `cosine`.
- `p`: Potencia de la métrica de Minkowski, se probaron valores aleatorios entre 1 y 4.
- `algorithm`: Algoritmo de búsqueda de vecinos, se probaron los valores `auto`, `ball_tree`, `kd_tree` y `brute`.

4.3. *SVM*

A continuación se detallan los hiperparámetros utilizados junto con sus respectivos valores de prueba:

- `C`: Parámetro de regularización, se probaron valores aleatorios entre 0,1 y 10.
- `kernel`: Tipo de kernel a utilizar, se probaron los valores `linear`, `poly`, `rbf` y `sigmoid`.
- `degree`: Grado del polinomio, se probaron valores aleatorios entre 1 y 5.
- `gamma`: Coeficiente del kernel, se probaron valores aleatorios entre 0,01 y 1.
- `coef0`: Término independiente en el kernel, se probaron valores aleatorios entre 0 y 1.

4.4. *LDA* y *GaussianNB*

A pesar de que para comparar se utilizaron los algoritmos de *LDA* y *GaussianNB* sin realizar una búsqueda de hiperparámetros, se mencionan los posibles hiperparámetros que podrían ser utilizados para estos algoritmos:

- `solver`: Método de optimización, se probaron los valores `svd`, `lsqr` y `eigen`.
- `priors`: Probabilidades a priori de las clases, se probaron valores aleatorios entre 0 y 1.
- `shrinkage`: Regularización, se probaron valores aleatorios entre 0 y 1.

4.5. Resultados

5. Ejercicio 4

Presente los resultados obtenidos, como métricas de evaluación, gráficos de desempeño, etc.

6. Ejercicio 5

Analice los resultados, las limitaciones del modelo y posibles mejoras.

7. Conclusión

Resuma los hallazgos principales y las conclusiones del informe.

Referencias

Incluya las referencias bibliográficas utilizadas en el informe.