Selección y Evaluación de Modelos de AA

Aprendizaje Automático

Borja González Seoane

borja.gonzalez1@uie.edu

Universidad Intercontinental de la Empresa

S07, 16 de octubre de 2024

Agenda

Agenda

Correlación entre variables

Correlación

- Se dice que dos variables están correlacionadas si su valor cambia de forma conjunta
- La correlación no implica causalidad

Ejemplo de correlación sin causalidad

- La cantidad de helados vendidos en un chiringuito de playa está correlacionada con el número de ataques de tiburones en la costa
- Aunque ambas variables están correlacionadas, no hay una relación de causalidad entre ellas

Correlación de Pearson

- La correlación entre dos variables se puede medir con el coeficiente de correlación de Pearson, que toma valores entre -1 y 1, y que se calcula como:

$$\rho_{X_1,X_2} = \frac{cov(X_1, X_2)}{\sigma_{X_1}\sigma_{X_2}} \tag{1}$$

- Donde $cov(X_1, X_2)$ es la covarianza entre X_1 y X_2, y σ_{X_1} y σ_{X_2} son las desviaciones típicas de X_1 y X_2 respectivamente
- Si $\rho_{X_1,X_2} = 1$, las variables están perfectamente correlacionadas de forma positiva
- Si $\rho_{X_1,X_2} = -1$, las variables están perfectamente correlacionadas de forma negativa
- Si $\rho_{X_1,X_2} = 0$, las variables no están correlacionadas en absoluto

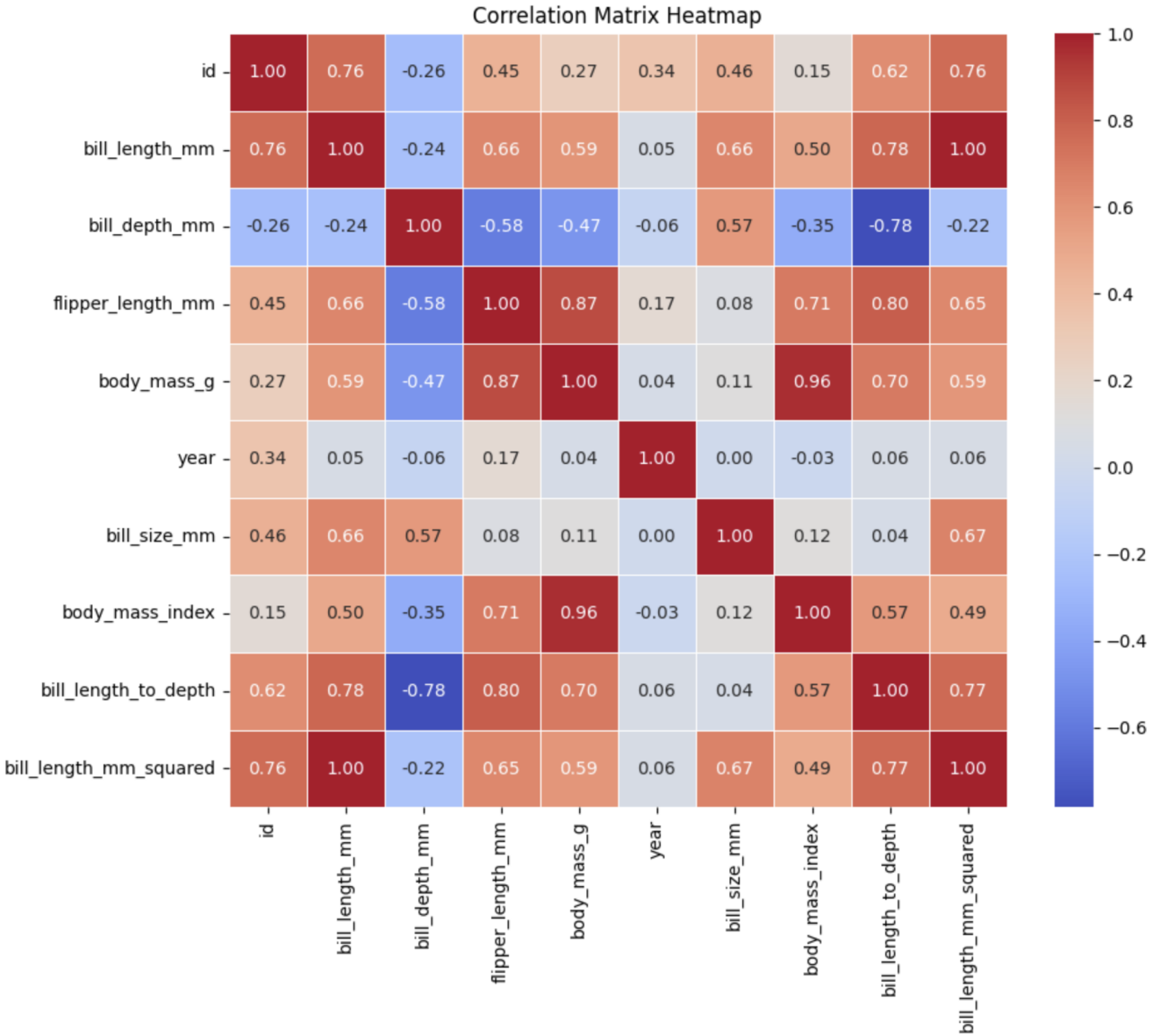
Correlación de Pearson [cont.]

- El signo de la correlación de Pearson indica la dirección de la relación entre las variables
- Una correlación positiva indica que a medida que una variable aumenta, la otra también lo hace
- Una correlación negativa indica que a medida que una variable aumenta, la otra disminuye

Matriz de correlaciones

- La matriz de correlación es una forma útil de visualizar las correlaciones entre variables
- En la matriz de correlación, cada celda muestra el coeficiente de correlación entre dos variables
- La diagonal de la matriz de correlación siempre es 1, ya que una variable está perfectamente correlacionada consigo misma, trivialmente
- La matriz de correlación es simétrica, ya que la correlación entre dos variables es la misma que la correlación entre las mismas variables en sentido inverso
- La matriz de correlación es una herramienta útil para identificar variables que están altamente correlacionadas, lo que puede ser un problema para algunos modelos de AA

Matriz de correlaciones [cont.]



Agenda

En Python

- En Python, la matriz de correlación se puede calcular con el método `.corr()` de un *dataframe* de Pandas
- Por defecto, el método `.corr()` calcula la correlación de Pearson
- El método `.corr()` devuelve un *dataframe* de Pandas con la matriz de correlación
- La matriz de correlación se puede visualizar con la función `heatmap()` de Seaborn

Agenda

Acciones a tomar con variables correlacionadas

- Si dos variables están altamente correlacionadas, es posible que una de ellas no aporte información adicional al modelo
- En estos casos, tal vez sea posible eliminar una de las dos variables correlacionadas
- Otra alternativa es combinar las dos variables correlacionadas en una sola variable que represente la información de ambas

Acciones a tomar con variables correlacionadas [cont.]

Ejemplo de cómo combinar variables correlacionadas

- Sea un conjunto de datos con información antropométrica de personas y un caso de uso que consiste en predecir el porcentaje de grasa corporal de una persona
- Probablemente haya una alta correlación entre algunas variables:
  - Peso y altura
    $\rightarrow$ **Índice de masa corporal**
  - Circunferencia del pecho y circunferencia de la cintura, en hombres
    $\rightarrow$ **Ratio cintura/pecho**
  - Circunferencia de la cadera y circunferencia de la cintura, en mujeres
    $\rightarrow$ **Ratio cintura/cadera**

En algunos casos, como el del ejemplo, la combinación de variables, además de emanar numéricamente del cálculo de la correlación, tiene un sentido contextual: mejor interpretabilidad de los resultados

Agenda

Agenda

Matriz de confusión

Real \ Predicción	Positivo	Negativo
Positivo	TP	FN
Negativo	FP	TN

Matriz de confusión [cont.]

- **TP** (*true positives*): los casos positivos que el modelo ha clasificado correctamente
- **TN** (*true negatives*): los casos negativos que el modelo ha clasificado correctamente
- **FP** (*false positives*): los casos negativos que el modelo ha clasificado incorrectamente como positivos
- **FN** (*false negatives*): los casos positivos que el modelo ha clasificado incorrectamente como negativos

Matriz de confusión [cont.]

Existen escenarios en los que un FN es más grave que un FP, y viceversa

Ejemplos de escenarios con diferente relevancia de FN y FP

- Un modelo de detección de spam que clasifica un correo electrónico importante como spam (FP) es peor que clasificar un correo basura como no spam (FN)
- Un modelo de detección de cáncer que clasifica a un paciente con cáncer como sano (FN) es peor que clasificar a un paciente sano como enfermo (FP)

Métricas de clasificación

- Exactitud (*accuracy*), predicciones correctas sobre el total de predicciones

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

¿En qué escenarios no parece una buena métrica la exactitud?

Conjuntos de datos desbalanceados

Se dice que un conjunto de datos está desbalanceado cuando una de las clases es mucho más frecuente que la otra

Ejemplo de conjunto de datos desbalanceado

Sea un conjunto con 1000 de transacciones bancarias, en el que el 99 % de las transacciones son legítimas (etiqueta 1) y el 1 % restante son fraudulentas (etiqueta 0)

Conjuntos de datos desbalanceados [cont.]

ModeloDummy

```
1 class ModeloDummy:
2   def predict(self, X):
3     return np.ones(X.shape[0])
```

¿Qué métrica de exactitud obtendría el `ModeloDummy` sobre el conjunto de datos desbalanceado del ejemplo anterior? El objetivo del modelo sería identificar las transacciones legítimas

Métricas de clasificación [cont.]

- En problemas de clasificación con datos desbalanceados, la exactitud no es una buena métrica
- Porque el modelo puede tener una alta exactitud simplemente prediciendo la clase mayoritaria

Métricas de clasificación [cont.]

- Precisión (*precision*)

$$\text{precision} = \frac{TP}{TP + FP} \tag{3}$$

- Sensibilidad (*recall*)

$$\text{recall} = \frac{TP}{TP + FN} \tag{4}$$

- Especificidad (*specificity*)

$$\text{specificity} = \frac{TN}{TN + FP} \tag{5}$$

- F1, media armónica de precisión y sensibilidad

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

Métricas de clasificación de `ModeloDummy`

¿Qué métricas de precisión y sensibilidad tendría el `ModeloDummy` sobre los datos desbalanceados del ejemplo anterior?

Métricas de `ModeloDummy` [cont.]

Métricas de `ModeloDummy`

- Exactitud

$$\text{accuracy} = \frac{990}{1000} = 0{,}99 \quad (7)$$

- Precisión

$$\text{precision} = \frac{990}{990 + 10} = 0{,}99 \quad (8)$$

- Sensibilidad

$$\text{recall} = \frac{990}{990 + 0} = 1 \quad (9)$$

- Especificidad

$$\text{specificity} = \frac{0}{0 + 10} = 0 \quad (10)$$

- F1

$$\text{F1} = 2 \cdot \frac{0{,}99 \cdot 1}{0{,}99 + 1} = 0{,}995 \quad (11)$$

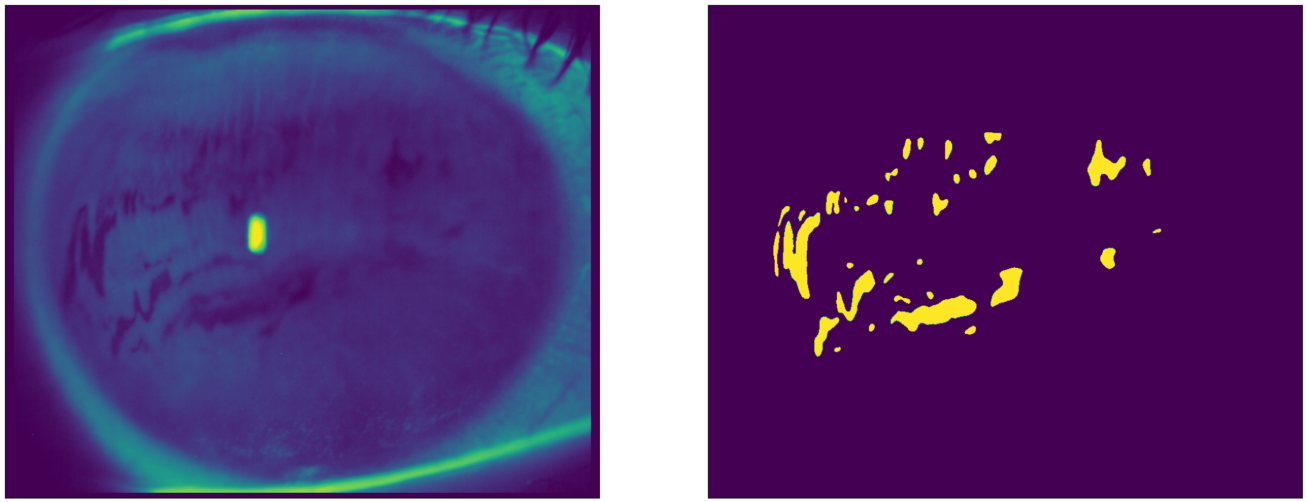
Métricas de clasificación [cont.]

- La precisión mide la calidad de las predicciones positivas
- La sensibilidad mide la fracción de positivos que se han identificado correctamente
- La especificidad mide la fracción de negativos que se han identificado correctamente
- La F1 es útil cuando se desea encontrar un equilibrio entre precisión y sensibilidad

Elección de las métricas de clasificación

- La elección de las métricas de clasificación depende del problema y de la importancia relativa de los falsos positivos y los falsos negativos
- También depende de si el conjunto de datos está desbalanceado o no
- Obviamente, trabajar con conjuntos de datos bien balanceados es lo ideal, tanto para el aprendizaje de los modelos como para la evaluación de su rendimiento

Métricas para casos particulares: visión por computador



De izquierda a derecha: imagen de un ojo con posible síndrome de ojo seco y máscara obtenida con un modelo de aprendizaje profundo de las zonas de ruptura de la película lagrimal

Métricas para casos particulares: visión por computador [cont.]

De izquierda a derecha: máscara anotada manualmente por un experto médico y máscara obtenida con el mismo modelo de aprendizaje profundo de la transparencia anterior

Métricas para casos particulares: visión por computador [cont.]

- En visión por computador, a la clasificación de píxeles en función de su contenido se le llama segmentación semántica
- Se obtienen máscaras como las anteriores, que codifican la pertenencia de cada píxel a una clase
- Por naturaleza, las máscaras están desbalanceadas, ya que la mayoría de los píxeles suelen pertenecer a la clase de fondo
- Además, como dificultad añadida, las máscaras anotadas manualmente normalmente no son perfectas, por lo que comparar la predicción con la anotación manual de forma directa tampoco es ideal

Métricas para casos particulares: visión por computador [cont.]

- En estos casos, se utilizan métricas como la similitud de Jaccard, también conocida como índice de similitud de Jaccard (*Jaccard index*)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{12}$$

- Donde A y B son dos conjuntos, en este caso, dos máscaras
- El índice de similitud de Jaccard mide la similitud entre dos conjuntos y toma valores entre 0 y 1
- Un valor de 1 indica que los conjuntos son idénticos
- Un valor de 0 indica que los conjuntos no tienen elementos en común

Agenda

Métricas de regresión

- En el caso de las tareas de regresión, lo que se desea predecir es un valor numérico
- Por tanto, las métricas de regresión miden la diferencia entre los valores reales y los valores predichos: el error

Métricas de regresión absolutas

- Error absoluto medio (*mean absolute error*, MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{13}$$

- Error cuadrático medio (*mean squared error*, MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{14}$$

- Raíz del error cuadrático medio (*root mean squared error*, RMSE)

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{15}$$

Métricas de regresión relativas

- Error absoluto relativo (*mean absolute percentage error*, MAPE)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{16}$$

- Error porcentual cuadrático medio (*mean squared percentage error*, MSPE)

$$\text{MSPE} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2 \tag{17}$$

- Raíz del error porcentual cuadrático medio (*root mean squared percentage error*, RMSPE)

$$\text{RMSPE} = \sqrt{\text{MSPE}} \tag{18}$$

Elección de las métricas de regresión

- La elección de las métricas de regresión depende del problema y de la importancia relativa de los errores
- Por ejemplo, el MAE es menos sensible a los valores atípicos que el RMSE
- Por tanto, si los valores atípicos son importantes en el problema, es mejor utilizar el MAE
- Se debe razonar numéricamente la elección de las métricas de regresión, en función de las características del problema

Elección de las métricas de regresión absolutas o relativas

- Típicamente se utilizan métricas de regresión absolutas, como el MAE o el RMSE, para el entrenamiento de los modelos
- Ello es así porque estas métricas preservan la escala de la variable objetivo
- No obstante, las métricas de regresión relativas, como el MAPE o el RMSPE, son útiles para la interpretación de los resultados, sobre todo cuando no se tiene demasiado contexto del problema o cuando se quiere obtener una perspectiva más general

Agenda

Métricas de aprendizaje no supervisado

- En el caso de las tareas de aprendizaje no supervisado, no se dispone de una variable objetivo
- Por tanto, no se pueden utilizar las métricas de clasificación o regresión
- En su lugar, se utilizan métricas específicas de cada tarea
- Por ejemplo, en el caso de la reducción de la dimensionalidad, se utilizan métricas como la varianza explicada o la capacidad de reconstrucción
- En modelos basados en distancias, también se pueden trabajar con la dispersión de los clústeres

Agenda

Métricas como funciones de pérdida

- En muchos casos, se utiliza una de las métricas anteriores como función de pérdida
- Por ejemplo, en regresión lineal, la función de pérdida habitual es el error cuadrático medio

Función de pérdida

Se llama función de pérdida a la función que mide el error entre las predicciones del modelo y los valores reales, y calcula una penalización por dicho error que se utiliza para ajustar los parámetros del modelo

- El resto de métricas que no se utilicen como funciones de pérdida se pueden emplear para evaluar el rendimiento del modelo

Agenda

Métricas de aprendizaje automático en Python

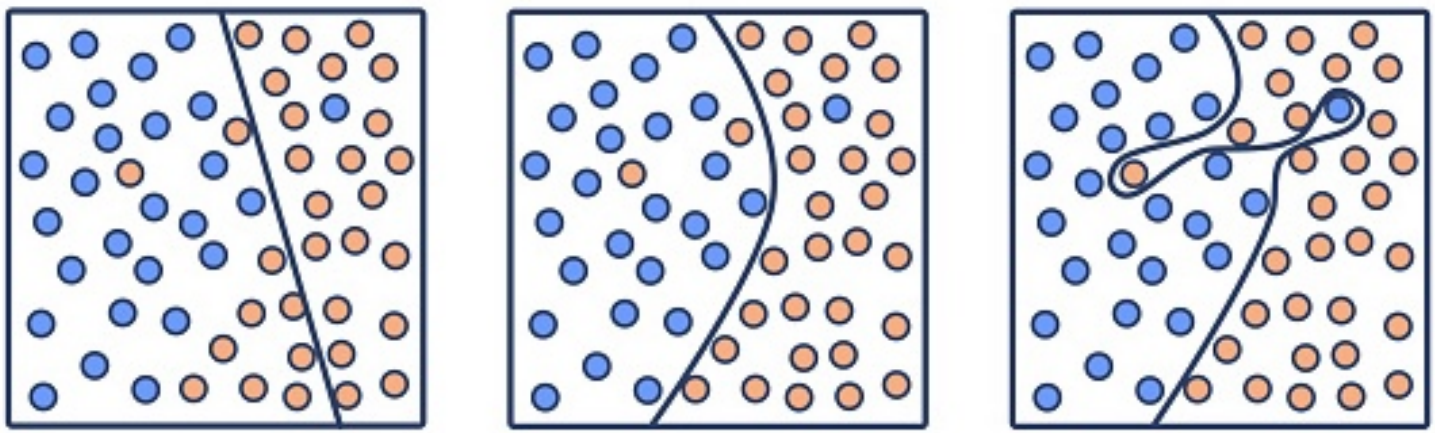
- En Python, las métricas de aprendizaje automático se pueden calcular con las funciones de Scikit-Learn. Módulo `sklearn.metrics`
- Las métricas de clasificación se pueden calcular con las funciones `accuracy_score()`, `precision_score()`, `recall_score()`, `f1_score()`, `confusion_matrix()`, `classification_report()`, etc.
- Las métricas de regresión se pueden calcular con las funciones `mean_absolute_error()`, `mean_squared_error()`, `mean_squared_log_error()`, `r2_score()`, etc.
- También se podría optar por implementaciones *ad hoc* de las métricas, si se desea

Agenda

Sobreajuste y subajuste

Sobreajuste o sobreentrenamiento (*overfitting*)

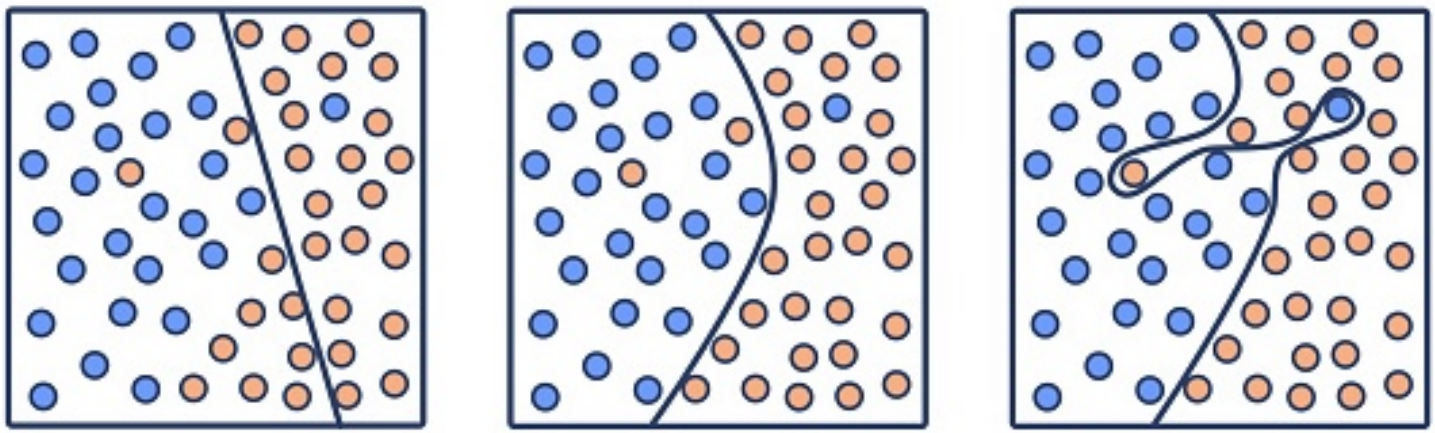
- Se dice que un modelo está sobreajustado cuando se ajusta demasiado bien a los datos de entrenamiento
- Un modelo sobreajustado es capaz de predecir muy bien los datos de entrenamiento, pero no generaliza bien a datos nuevos



Sobreajuste y subajuste

Subajuste o infraentrenamiento (*underfitting*)

- Se dice que un modelo está subajustado cuando no es capaz de capturar la estructura subyacente de los datos
- Un modelo subajustado no es capaz de predecir bien ni los datos de entrenamiento ni los datos nuevos
- Normalmente, un modelo subajustado es demasiado simple para el problema que se quiere resolver



Detección de sobreajuste

- La detección de sobreajuste se puede hacer de forma visual, comparando las curvas de aprendizaje del modelo, o también numéricamente, comparando las métricas de rendimiento del modelo en los datos de entrenamiento y en los datos de validación o test
- Un modelo sobreajustado tendrá un rendimiento muy bueno en los datos de entrenamiento, pero un rendimiento **mucho** peor en los datos de validación o test

Detección de subajuste

- La detección del subajuste resulta más sencilla, ya que el rendimiento del modelo en los datos de entrenamiento y en los datos de validación o test será malo
- Difícilmente se obtendrán buenas métricas que puedan conducir a engaño

Agenda

Particiones de validación y test

- La aproximación más simple de repartir los datos en un proyecto de AA es en dos particiones: entrenamiento y test
- Sin embargo, es habitual trabajar con una tercera partición, la de **validación**

Particiones de validación y test [cont.]

Partición de entrenamiento

- Se utiliza para ajustar los parámetros (pesos) del modelo
- Es la partición más grande y de la que emerge el aprendizaje

Partición de validación

Se utiliza para ajustar los hiperparámetros del modelo y comprobar el proceso de entrenamiento

Partición de test

Se utiliza para evaluar el rendimiento del modelo al final del proceso de entrenamiento

Particiones de validación y test [cont.]

En Python

- En Python, las particiones de validación y test se pueden hacer con la función `train_test_split()` de Scikit-Learn
- La función `train_test_split()` divide los datos en dos particiones: entrenamiento y test
- Para dividir los datos en tres particiones, simplemente se puede utilizar la función `train_test_split()` dos veces

Estratificación de las particiones

- En problemas de clasificación, es importante que las particiones de entrenamiento, validación y test mantengan la misma proporción de clases que el conjunto de datos original
- Para ello, se utiliza la estratificación
- Se dice que una partición está estratificada con respecto a una variable si mantiene la proporción de clases de esa variable entre las distintas subparticiones

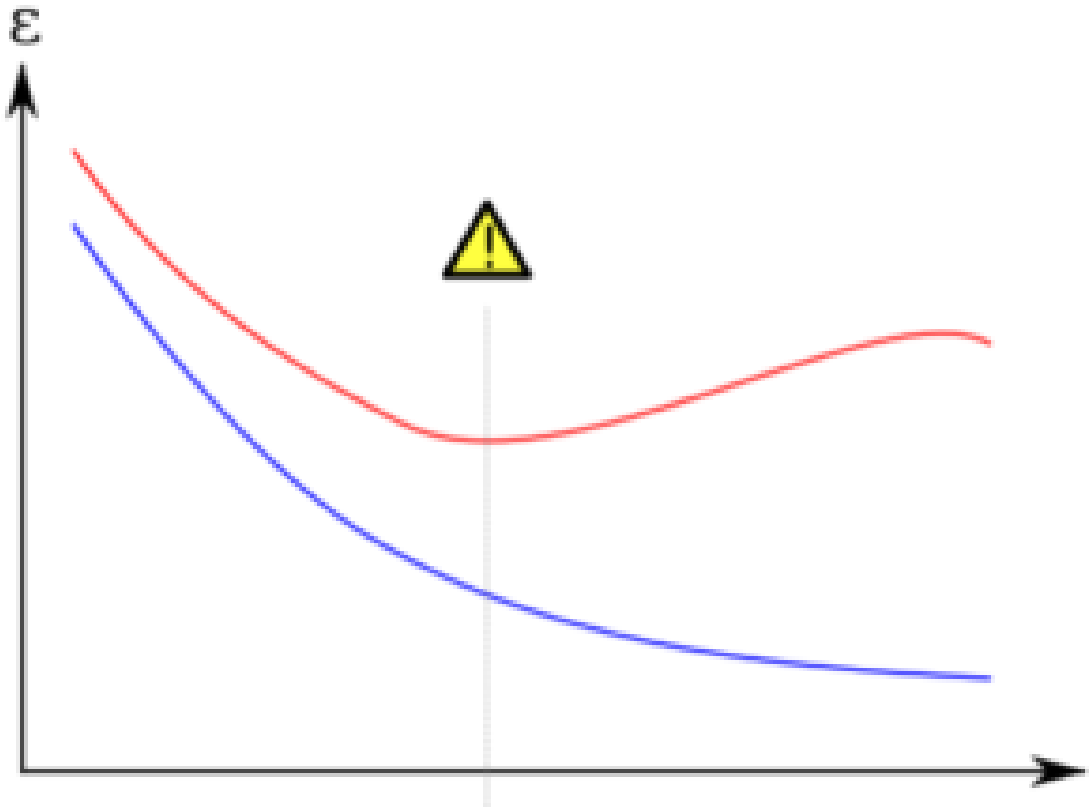
En Python

- En Python, la estratificación de las particiones se puede hacer con el argumento `stratify` de la función `train_test_split()` de Scikit-Learn
- El argumento `stratify` recibe la variable objetivo y mantiene la proporción de clases en las particiones

Validación en el bucle de entrenamiento

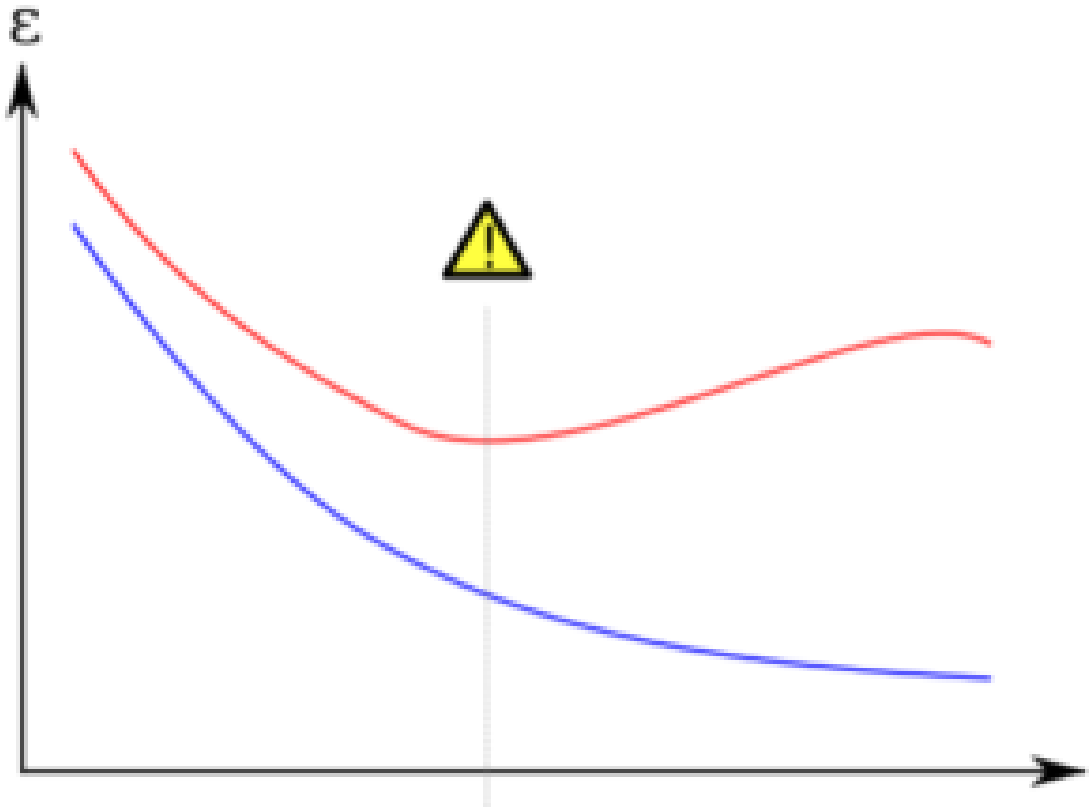
- El entrenamiento de un modelo de AA es un proceso iterativo en el que se van ajustando los parámetros del modelo para capturar la estructura subyacente de los datos
- Cuando se trabaja con una partición de datos de validación, se utilizará esta partición para ajustar los hiperparámetros del modelo al final de cada iteración del bucle
- Además, se puede utilizar la partición de validación para comprobar el proceso de entrenamiento y ajustar la tasa de aprendizaje
- Existen diferentes estrategias...

Curvas de aprendizaje



- Si en cada iteración vamos guardando el error (ϵ) de entrenamiento y el error de validación, al final podremos pintar las curvas de aprendizaje, como el ejemplo de la figura

Curvas de aprendizaje [cont.]



- ¿Cuál es la curva de entrenamiento y cuál es la curva de validación?
- ¿Qué está ocurriendo en el modelo de la figura?

Curvas de aprendizaje [cont.]

- La curva de entrenamiento (azul) desciende progresivamente, ya que el modelo va aprendiendo de los datos de entrenamiento
- Sin embargo, se aprecia que la curva de validación (roja) se estanca y no mejora, lo que indica que el modelo no generaliza bien a datos nuevos
- Esto es un claro indicio de **sobreajuste**

Early stopping

- Basándose también en las curvas de aprendizaje, se puede implementar la técnica de *early stopping* de forma muy intuitiva
- La técnica de *early stopping* consiste en detener el entrenamiento del modelo cuando el error de validación deja de disminuir
- Se puede considerar una tolerancia de un número de iteraciones sin mejora, para evitar detener el entrenamiento en un punto de estancamiento local
- En la práctica, se pueden hacer dos cosas:
  - Detener realmente el bucle
  - Ir guardando el modelo que mejor rendimiento tenga en la partición de validación y sólo sobreescribirlo si se obtiene un modelo mejor en la iteración en curso

Agenda

Aleatoriedad en la partición de los datos

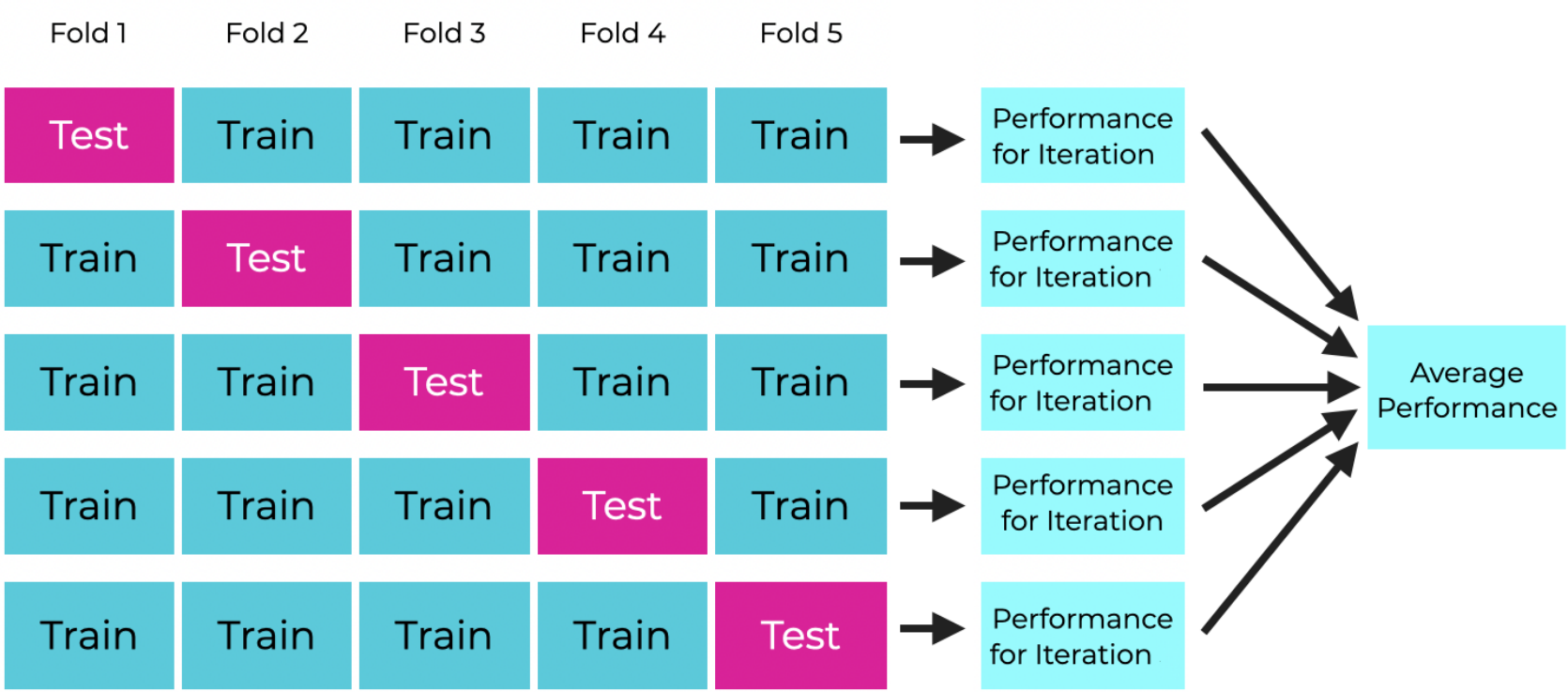
- La partición de los datos en entrenamiento, validación y test es un proceso aleatorio
- Por tanto, el rendimiento del modelo puede depender de la selección de las particiones
- Este problema se agrava en conjuntos de datos pequeños

Validación cruzada (*cross-validation*)

- La validación cruzada es una técnica que se utiliza para evaluar el rendimiento de un modelo
- La validación cruzada consiste en dividir los datos en k particiones, y entrenar y evaluar el modelo k veces, utilizando una partición distinta como conjunto de validación en cada iteración
- De esa forma, se minimiza el riesgo de que el rendimiento del modelo dependa de la selección de las particiones de entrenamiento y validación [1]
- Al final del proceso, se calcula la media de las métricas de rendimiento del modelo en las k iteraciones y se selecciona el modelo con mejor rendimiento

[1] Es decir, se reduce el peso de la aleatoriedad en la evaluación del modelo

Validación cruzada [cont.]



Validación cruzada [cont.]

- La validación cruzada es una técnica muy potente, pero también computacionalmente costosa
- En escenarios con conjuntos de datos pesados, la validación cruzada puede ser inviable
- Pesados no implica necesariamente grandes, por ejemplo: conjuntos de datos de vídeo, audio, imágenes de alta resolución, etc.

Agenda

Validación cruzada en Python

- En Python, la validación cruzada se puede hacer con la función `cross_val_score()` de Scikit-Learn
- La función `cross_val_score()` entrena y evalúa el modelo k veces, utilizando una partición distinta como conjunto de validación en cada iteración
- La función `cross_val_score()` devuelve una lista con las métricas de rendimiento del modelo en las k iteraciones
- La función `cross_val_score()` también permite paralelizar el proceso de validación cruzada, para reducir el tiempo de cómputo

Agenda

Hiperparámetros

Hiperparámetros

- Los hiperparámetros son parámetros que afectan al desempeño de un modelo de AA
- Pero que no se ajustan durante el proceso de entrenamiento del modelo: no emanan directamente del aprendizaje de los datos
- Para ajustar los hiperparámetros de un modelo, se suele partir de un enfoque de simple prueba y error
- Se seleccionarán los hiperparámetros que proporcionen el mejor rendimiento del modelo, sin mayor justificación
- Una de las primeras cosas que hacer cuando se considera un modelo para un proyecto es revisar su documentación y ver, entre otras cosas, qué hiperparámetros se podrían ajustar

Exploración de hiperparámetros exhaustiva vs. aleatoria

- La posibilidad de probar todas las combinaciones posibles de hiperparámetros de un modelo es una opción viable en problemas pequeños
- Pero en problemas más grandes, la búsqueda exhaustiva es inviable, o cuanto menos, muy costosa
- En estos casos, se puede recurrir a la búsqueda de hiperparámetros aleatoria
- La búsqueda de hiperparámetros aleatoria seleccionará subconjuntos aleatorios de hiperparámetros y los evaluará
- El proceso se puede repetir un número determinado de veces, o hasta que se alcance un criterio de parada
- Obviamente, lo ideal es tener en cuenta el conocimiento del dominio del que se disponga y la experiencia con las arquitecturas de los modelos a considerar para restringir lo máximo posible el espacio de búsqueda de hiperparámetros

Agenda

Optimización de hiperparámetros en Python

`GridSearchCV`

- La función `GridSearchCV` ajusta los hiperparámetros del modelo uno a uno, evaluando todas las combinaciones posibles
- La función `GridSearchCV` recibe como argumento un diccionario con los hiperparámetros a ajustar y sus valores
- La función `GridSearchCV` evalúa el rendimiento del modelo en todas las combinaciones posibles de hiperparámetros, utilizando **validación cruzada**
- La función `GridSearchCV` selecciona los hiperparámetros que proporcionen el mejor rendimiento del modelo

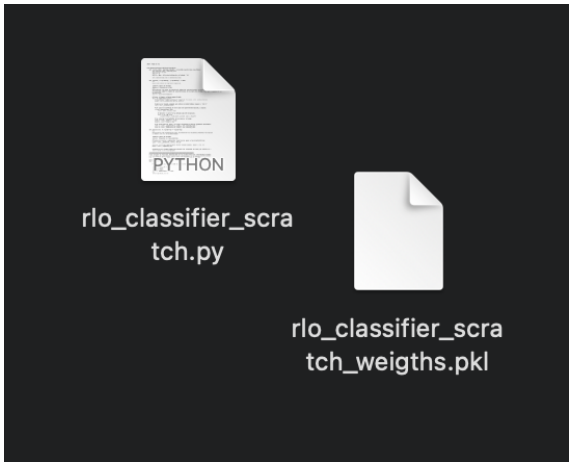
Optimización de hiperparámetros en Python [cont.]

`RandomizedSearchCV`

- La función `RandomizedSearchCV` ajusta los hiperparámetros del modelo de forma aleatoria
- La función `RandomizedSearchCV` recibe como argumento un diccionario con los hiperparámetros a ajustar y sus valores, y un número determinado de iteraciones
- La función `RandomizedSearchCV` selecciona subconjuntos aleatorios de hiperparámetros y evalúa el rendimiento del modelo en cada iteración, utilizando **validación cruzada**
- La función `RandomizedSearchCV` selecciona los hiperparámetros que proporcionen el mejor rendimiento del modelo

Agenda

Persistencia de modelos



- A nivel programático, un modelo de AA estaría compuesto por la arquitectura del modelo —la clase, el código— y los pesos ajustados durante el proceso de entrenamiento
- La persistencia de modelos es el proceso de guardar un modelo de AA entrenado para poder utilizarlo en el futuro
- La persistencia de modelos es importante para no tener que volver a entrenar el modelo cada vez que se quiera utilizar
- Además, es necesaria para poder desplegar los modelos en un entorno productivo

Persistencia de modelos [cont.]

- En verdad, la persistencia de los modelos consiste simplemente en serializar el modelo en un archivo salvable en disco
- Ergo cualquier protocolo de serialización con implementación en Python sería válido para persistir un modelo de AA
- Hay algunas opciones bastante populares: Pickle, Joblib, HDF5, etc.
- Su elección dependerá de las necesidades del proyecto, cuestiones de arquitectura del entorno u otras consideraciones de carácter puramente técnico

Lectura recomendada y propuesta para el F1

- *Models Persistence*[2], de la documentación de Scikit-Learn
- Lectura recomendada en general
- Propuesta también para resumir y comentar en el F1

[2] Scikit-Learn Developers. (s.f.). *Model Persistence - Scikit-Learn 1.5.2 documentation*. Consultado 14 de octubre de 2024, desde https://scikit-learn.org/stable/model_persistence.html

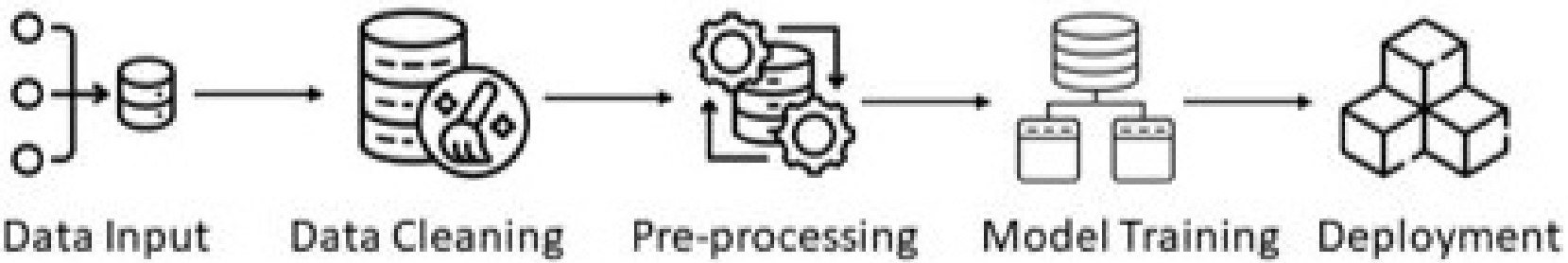
Agenda

Gestión de proyectos de AA

- A día de hoy, los *notebooks* se han convertido en la herramienta de facto para el desarrollo de proyectos de AA, sobre todo en fases más iniciales
- Son la herramienta perfecta para la experimentación y la iteración a la que recurren los científicos e ingenieros de datos
- Los proyectos más sencillos se pueden desarrollar íntegramente en un único *notebook*
- Pero en escenarios más realistas, es conveniente pensar en los proyectos como canalizaciones o *pipelines* de datos

Gestión de proyectos de AA [cont.]

- Así pues, se podrían ir separando los diferentes pasos de un proyecto de AA en diferentes *notebooks* o en diferentes scripts de Python
- El resultado de un paso se persiste de alguna forma y se recupera en el paso siguiente como entrada



Gestión de proyectos de AA [cont.]

- Una buena recomendación sería utilizar nombres de archivo descriptivos y coherentes para los *notebooks* o los scripts de Python
- También es buena idea utilizar índices para numerar los *notebooks* o los scripts de Python, para indicar el orden en el que se deben ejecutar dentro del *pipeline*

Ejemplo de nombrado en un proyecto tipo

- `n01_carga_datos.ipynb`
- `n02_limpieza_datos.ipynb`
- `n03_ingenieria_variables.ipynb`
- `n04_entrenamiento_modelo.ipynb`
- `n05_evaluacion_modelo.ipynb`

Si se trabaja con Python, se debería evitar nombrar archivos comenzando por un número, ya que Python no permite importar módulos con nombres que comiencen por un número

Gestión de proyectos de AA [cont.]

Recomendación extracurricular: Git

- Cuando se trabaja en cualquier tipo de proyecto de código, es recomendable utilizar un sistema de control de versiones, como Git
- Git es un sistema de control de versiones distribuido, que permite llevar un registro de los cambios en el código
- Se puede saber en todo momento quién ha hecho qué, cuándo y por qué. Se puede volver a versiones anteriores del código, etc.
- Permite una mejor colaboración entre los miembros de un equipo
- **Git se quede completamente fuera del alcance de este curso**

El aprendizaje automático no es magia, es sólo matemáticas aplicadas a los datos.

— Peter Norvig

Selección y Evaluación de Modelos de AA

Aprendizaje Automático

Borja González Seoane

borja.gonzalez1@uie.edu

Universidad Intercontinental de la Empresa

S07, 16 de octubre de 2024

Versión del documento: 14 de octubre de 2024