Segunda Práctica
Reconocimiento de Patrones
Curso 2025-2026

Sergiy Khudoley

2 de febrero de 2026

Índice

1. Preparando el entorno **2**

1.1. Entornos virtuales y `pip` 2

1.2. Pandas 2

1.3. Scikit-learn 2

2. Conceptos teóricos **3**

2.1. Limpieza y transformación de datos 3

2.2. Modelos de clasificación SVC y KNN 4

2.3. Métricas de evaluación 4

3. Ejecución de la práctica **5**

3.1. Selección de características y etiquetas 5

3.2. Limpiando NaN y valores nulos 5

3.3. Estandarizando datos 5

3.4. Entrenando modelos 5

3.5. Métricas y resultados 6

4. Conclusiones **8**

5. Bibliografía **10**

1. Preparando el entorno

Antes de comenzar con la práctica, es fundamental disponer de un entorno de trabajo controlado y reproducible. Para ello, se recomienda el uso de entornos virtuales en Python, junto con la instalación de las librerías necesarias mediante `pip`. Esto permite aislar los paquetes y versiones utilizados, evitando conflictos con otras instalaciones en el sistema.

1.1. Entornos virtuales y `pip`

Un **entorno virtual** es una instalación aislada de Python que permite mantener dependencias específicas para un proyecto sin afectar al resto del sistema. Para crear y activar un entorno virtual, se puede utilizar el módulo `venv` que viene incluido en Python:

```
# Crear un entorno virtual llamado 'venv'
python -m venv venv

# Activar el entorno virtual (Linux )
source venv/bin/activate

# Activar el entorno virtual (Windows)
venv\Scripts\activate
```

Una vez activado el entorno, se puede utilizar `pip`, el gestor de paquetes de Python, para instalar librerías necesarias. Por ejemplo:

```
pip install --upgrade pip   # Actualiza pip a la última versión
```

`pip` permite instalar, actualizar y desinstalar paquetes de manera sencilla y es el estándar en el ecosistema Python.

1.2. Pandas

`Pandas` es una librería de Python especializada en la manipulación y análisis de datos. Su estructura principal es el `DataFrame`, similar a una tabla en Excel o una hoja de cálculo, que permite almacenar datos de manera eficiente y realizar operaciones como filtrado, limpieza, agrupamiento o combinación de datasets. Para instalar `pandas`:

```
pip install pandas
```

Una vez instalada, `pandas` permite, por ejemplo, leer archivos CSV, limpiar valores nulos y transformar datos de forma sencilla, lo que resulta esencial antes de aplicar modelos de machine learning.

1.3. Scikit-learn

`Scikit-learn` es una librería de Python ampliamente utilizada para **aprendizaje automático**, proporcionando herramientas para clasificación, regresión, clustering y evaluación de modelos. Incluye implementaciones de algoritmos como SVC (Support Vector Classifier) y KNN (K-Nearest Neighbors), además de utilidades para escalado de datos, selección de características y validación cruzada. Su instalación se realiza mediante `pip`:

```
pip install scikit-learn
```

Con `scikit-learn`, es posible entrenar modelos predictivos de forma sencilla y aplicar métricas de evaluación estándar como exactitud (*accuracy*), precisión (*precision*) o exhaustividad (*recall*), entre otras.

2. Conceptos teóricos

En esta sección se presentan los fundamentos teóricos necesarios para comprender el desarrollo de la práctica. Se aborda la preparación de los datos, los modelos de clasificación utilizados y las métricas de evaluación más relevantes en aprendizaje automático supervisado.

2.1. Limpieza y transformación de datos

Antes de aplicar cualquier algoritmo de aprendizaje automático, es imprescindible garantizar que los datos sean consistentes, completos y adecuados para el análisis. La limpieza y transformación de datos consisten en una serie de pasos clave:

- **Eliminación de columnas irrelevantes:** En muchos datasets existen columnas que no aportan información útil para el modelo, como identificadores únicos (`student_id`) o datos externos al problema (`country`). Estas columnas se eliminan para reducir la complejidad y evitar que el modelo aprenda patrones espurios.
- **Tratamiento de valores faltantes:** Las celdas vacías o valores nulos (`NaN`) pueden afectar negativamente al rendimiento del modelo. Se pueden eliminar filas incompletas, reemplazar los valores nulos por la media, mediana o un valor constante, o aplicar técnicas más avanzadas de imputación.
- **Transformación de variables categóricas:** Muchas columnas contienen valores no numéricos (por ejemplo, `Beginner, Intermediate, Advanced`) que deben convertirse a un formato numérico que los algoritmos puedan interpretar. Esto se puede hacer mediante *label encoding*, donde cada categoría recibe un valor entero, o *one-hot encoding*, donde cada categoría se transforma en una columna binaria.
- **Estandarización o normalización de datos:** Las columnas numéricas suelen tener rangos distintos (por ejemplo, `age` puede ir de 18 a 40, mientras que `hours_spent_learning` puede ir de 0 a 40). Para evitar que las características con valores más grandes dominen el aprendizaje, se aplican técnicas de escalado, como *standardization*, que centra los datos en media cero y desviación estándar uno.
- **Agrupación de valores para clasificación:** En este caso, la nota final del estudiante (`final_exam_score`) se agrupa en rangos categóricos (*suspenso, aprobado, notable, sobresaliente*) para convertir un problema de regresión en uno de clasificación.

Estos pasos permiten obtener un conjunto de datos limpio y homogéneo, listo para entrenar modelos de aprendizaje automático, reduciendo la probabilidad de errores y aumentando la calidad de las predicciones.

2.2. Modelos de clasificación SVC y KNN

Para la práctica se han utilizado dos modelos de clasificación supervisada ampliamente conocidos:

- **Support Vector Classifier (SVC):** SVC es un algoritmo basado en la teoría de máquinas de soporte vectorial. Su objetivo es encontrar un *hiperplano* que separe las clases del dataset de manera óptima, maximizando la *margen* entre las categorías. Para problemas no lineales, SVC puede utilizar *kernels* (por ejemplo, `rbf` o `linear`) que permiten transformar los datos a un espacio de mayor dimensión donde la separación es más sencilla.
- **K-Nearest Neighbors (KNN):** KNN es un algoritmo de clasificación basado en instancias. Para predecir la clase de un nuevo ejemplo, se consideran los k vecinos más cercanos en el espacio de características y se asigna la clase más frecuente entre ellos. Es un modelo simple pero eficaz, especialmente cuando los datos están bien escalados y no presentan ruido excesivo.

Ambos modelos presentan ventajas y desventajas: SVC es potente para datos complejos y no lineales, pero puede ser costoso computacionalmente; KNN es intuitivo y fácil de implementar, pero sensible a la escala de los datos y a valores atípicos.

2.3. Métricas de evaluación

Una vez entrenado un modelo, es fundamental evaluar su rendimiento utilizando métricas adecuadas. En clasificación, las más utilizadas incluyen:

- **Exactitud (*Accuracy*):** Proporción de predicciones correctas sobre el total de ejemplos. Es intuitiva, pero puede ser engañosa si las clases están desbalanceadas.
- **Error:** Complemento de la exactitud, calculado como $1 - \text{accuracy}$. Representa la proporción de predicciones incorrectas.
- **Precisión (*Precision*):** Indica qué proporción de los ejemplos clasificados como positivos son realmente positivos. Es especialmente útil cuando las falsas alarmas tienen un coste alto.
- **Exhaustividad (*Recall*):** Mide qué proporción de los ejemplos positivos reales fueron correctamente identificados. Es importante cuando no se quiere dejar de detectar casos positivos.

Estas métricas proporcionan una visión completa del rendimiento del modelo y permiten compararlo con diferentes configuraciones y algoritmos, asegurando la elección del modelo más adecuado para el problema planteado.

3. Ejecución de la práctica

En esta sección se describen los pasos realizados para llevar a cabo la práctica, desde la selección de datos hasta el entrenamiento de los modelos y la evaluación de su rendimiento. Se documenta tanto la metodología seguida como la justificación de cada decisión tomada durante el análisis y modelado.

3.1. Selección de características y etiquetas

El primer paso consiste en identificar cuáles columnas del conjunto de datos serán utilizadas como características (*features*) y cuál será la variable objetivo (*label*) a predecir.

En nuestro caso, el dataset contiene información sobre estudiantes y sus hábitos de estudio, incluyendo variables como edad, semanas cursadas, horas de estudio semanales, práctica de ejercicios, participación en foros y confianza autodetectada en Python. La columna objetivo es `final_exam_score`, que indica la nota obtenida en el examen final.

Se seleccionan como características todas las columnas relevantes para la predicción, eliminando aquellas que no aportan información significativa, como el identificador del estudiante (`student_id`) y el país de procedencia (`country`). La columna `passed_exam` también se elimina, dado que se trata de un derivado de la nota final y podría sesgar el modelo.

3.2. Limpiando NaN y valores nulos

Una vez seleccionadas las columnas, se procede a filtrar los valores nulos o ausentes. La presencia de `NaN` o datos incompletos puede provocar errores en los algoritmos de aprendizaje automático o reducir su rendimiento.

En la práctica, se eliminan todas las filas cuya columna `prior_programming_experience` contenga valores nulos, asegurando que el modelo reciba únicamente datos completos. Este procedimiento es fundamental para mantener la consistencia y la calidad de los datos de entrada.

3.3. Estandarizando datos

Los datos numéricos presentan rangos y escalas diferentes, lo que puede afectar al comportamiento de ciertos algoritmos, como KNN o SVC. Para mitigar este efecto, se aplica una estandarización (*standardization*), que transforma cada columna de manera que tenga media cero y desviación estándar uno.

Además, se transforman las variables categóricas relevantes en valores numéricos mediante *label encoding*. Por ejemplo, la experiencia previa en programación se codifica como `Beginner = 0`, `Intermediate = 1` y `Advanced = 2`. De manera similar, la nota final se agrupa en categorías (*suspenso, aprobado, notable, sobresaliente*) y se codifica de 0 a 3. Este procedimiento permite que los algoritmos interpreten correctamente las relaciones entre categorías.

3.4. Entrenando modelos

Con los datos preparados, se divide el conjunto en entrenamiento (70 %) y prueba (30 %) para evaluar posteriormente el desempeño de los modelos en datos no vistos.

Se entrenan dos modelos de clasificación:

	A	B	C	D	E	F
	params	mean_test_score	std_test_score	rank_test_score	accuracy	error
	{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
	{'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}	0.697636748426308	0.0119760354591843	28	0.692439862542955	0.307560137457045
	{'C': 0.1, 'gamma': 'auto', 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
	{'C': 0.1, 'gamma': 'auto', 'kernel': 'rbf'}	0.696163446928587	0.0129429204501795	29	0.689003436426117	0.310996563573883
	{'C': 0.1, 'gamma': 0.01, 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
	{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}	0.669600607770784	0.0207913275681122	33	0.661512027491409	0.338487972508591
	{'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
	{'C': 0.1, 'gamma': 0.1, 'kernel': 'rbf'}	0.697639461688735	0.011677402367191	27	0.694158075601375	0.305841924398625
	{'C': 0.1, 'gamma': 1, 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
	{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}	0.47861406555242	0.00146821346473355	40	0.436426116838488	0.563573883161512
	{'C': 1, 'gamma': 'scale', 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
	{'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}	0.705749403082266	0.0193389050441149	26	0.695876288659794	0.304123711340206
	{'C': 1, 'gamma': 'auto', 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
	{'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}	0.708701432602561	0.0188204532431758	23	0.699312714776632	0.300687285223368
	{'C': 1, 'gamma': 0.01, 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
	{'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}	0.70647927067506	0.0161076955048562	25	0.699312714776632	0.300687285223368
	{'C': 1, 'gamma': 0.1, 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
	{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}	0.707225417842414	0.0189237985211335	24	0.699312714776632	0.300687285223368
	{'C': 1, 'gamma': 1, 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
	{'C': 1, 'gamma': 1, 'kernel': 'rbf'}	0.571516171044063	0.0215129812701254	39	0.570446735395189	0.429553264604811
	{'C': 10, 'gamma': 'scale', 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
	{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}	0.673268938571739	0.0444738590448016	31	0.670103092783505	0.329896907216495
	{'C': 10, 'gamma': 'auto', 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
	{'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}	0.672530931191665	0.0451940016117807	32	0.690721649484536	0.309278350515464
	{'C': 10, 'gamma': 0.01, 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
	{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.736724006945952	0.0204431596151525	11	0.714776632302406	0.285223367697595
	{'C': 10, 'gamma': 0.1, 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
	{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}	0.67621554156718	0.0458030959888387	30	0.673539518900344	0.326460481099656
	{'C': 10, 'gamma': 1, 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
	{'C': 10, 'gamma': 1, 'kernel': 'rbf'}	0.594364553939657	0.0332559478693312	37	0.584192439862543	0.415807560137457
	{'C': 100, 'gamma': 'scale', 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
	{'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}	0.626817885825917	0.0338646108462252	36	0.618556701030928	0.381443298969072
	{'C': 100, 'gamma': 'auto', 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
	{'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}	0.63640112871717	0.0408293057034803	34	0.630584192439863	0.369415807560138
	{'C': 100, 'gamma': 0.01, 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
	{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}	0.715338072498372	0.0265568521226799	22	0.709621993127148	0.290378006872852
	{'C': 100, 'gamma': 0.1, 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
	{'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}	0.630505209463859	0.0338618111952958	35	0.623711340206186	0.376288659793814
	{'C': 100, 'gamma': 1, 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
	{'C': 100, 'gamma': 1, 'kernel': 'rbf'}	0.594364553939657	0.0332559478693312	37	0.584192439862543	0.415807560137457

Figura 1: Métricas de distintos modelos de SVC entrenados

- **Support Vector Classifier (SVC):** Se realiza una búsqueda en rejilla (*GridSearchCV*) para encontrar los mejores parámetros (`C`, `kernel`, `gamma`) utilizando validación cruzada a 5 pliegues y como métrica principal la exactitud (*accuracy*). El modelo seleccionado se ajusta con los datos de entrenamiento y se evalúa posteriormente con los datos de prueba.
- **K-Nearest Neighbors (KNN):** De manera similar, se emplea *GridSearchCV* para explorar combinaciones de parámetros (`n_neighbors`, `weights`, `metric`). Se ajusta el modelo con los datos de entrenamiento y se calcula la exactitud sobre el conjunto de prueba.

Este enfoque permite seleccionar automáticamente la configuración que proporciona el mejor rendimiento promedio en la validación cruzada, asegurando una estimación robusta del desempeño de los modelos.

3.5. Métricas y resultados

Finalmente, se generan métricas de evaluación sobre el conjunto de prueba para cada combinación de parámetros explorada. Las métricas calculadas incluyen:

- **Exactitud (*accuracy*):** Proporción de predicciones correctas.
- **Error:** Complemento de la exactitud (1 − accuracy).

A	B	C	D	E	F
params	mean_test_score	std_test_score	rank_test_score	accuracy	error
{'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
{'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}	0.697636748426308	0.0119760354591843	28	0.692439862542955	0.307560137457045
{'C': 0.1, 'gamma': 'auto', 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
{'C': 0.1, 'gamma': 'auto', 'kernel': 'rbf'}	0.696163446928587	0.0129429204501795	29	0.689003436426117	0.310996563573883
{'C': 0.1, 'gamma': 0.01, 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}	0.669600607770784	0.0207913275681122	33	0.661512027491409	0.338487972508591
{'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
{'C': 0.1, 'gamma': 0.1, 'kernel': 'rbf'}	0.697639461688735	0.011677402367191	27	0.694158075601375	0.305841924398625
{'C': 0.1, 'gamma': 1, 'kernel': 'linear'}	0.745574668981984	0.0168333223805757	1	0.718213058419244	0.281786941580756
{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}	0.47861406555242	0.00146821346473355	40	0.436426116838488	0.563573883161512
{'C': 1, 'gamma': 'scale', 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
{'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}	0.705749403082266	0.0193389050441149	26	0.695876288659794	0.304123711340206
{'C': 1, 'gamma': 'auto', 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
{'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}	0.708701432602561	0.0188204532431758	23	0.699312714776632	0.300687285223368
{'C': 1, 'gamma': 0.01, 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
{'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}	0.70647927067506	0.0161076955048562	25	0.699312714776632	0.300687285223368
{'C': 1, 'gamma': 0.1, 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}	0.707225417842414	0.0189237985211335	24	0.699312714776632	0.300687285223368
{'C': 1, 'gamma': 1, 'kernel': 'linear'}	0.733028543520729	0.0160825068505051	17	0.723367697594502	0.276632302405498
{'C': 1, 'gamma': 1, 'kernel': 'rbf'}	0.571516171044063	0.0215129812701254	39	0.570446735395189	0.429553264604811
{'C': 10, 'gamma': 'scale', 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
{'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}	0.673268938571739	0.0444738590448016	31	0.670103092783505	0.329896907216495
{'C': 10, 'gamma': 'auto', 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
{'C': 10, 'gamma': 'auto', 'kernel': 'rbf'}	0.672530931191665	0.0451940016117807	32	0.690721649484536	0.309278350515464
{'C': 10, 'gamma': 0.01, 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}	0.736724006945952	0.0204431596151525	11	0.714776632302406	0.285223367697595
{'C': 10, 'gamma': 0.1, 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}	0.67621554156718	0.0458030959888387	30	0.673539518900344	0.326460481099656
{'C': 10, 'gamma': 1, 'kernel': 'linear'}	0.737456587801172	0.0134032846053836	6	0.716494845360825	0.283505154639175
{'C': 10, 'gamma': 1, 'kernel': 'rbf'}	0.594364553939657	0.0332559478693312	37	0.584192439862543	0.415807560137457
{'C': 100, 'gamma': 'scale', 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
{'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}	0.626817885825917	0.0338646108462252	36	0.618556701030928	0.381443298969072
{'C': 100, 'gamma': 'auto', 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
{'C': 100, 'gamma': 'auto', 'kernel': 'rbf'}	0.63640112871717	0.0408293057034803	34	0.630584192439863	0.369415807560138
{'C': 100, 'gamma': 0.01, 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
{'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}	0.715338072498372	0.0265568521226799	22	0.709621993127148	0.290378006872852
{'C': 100, 'gamma': 0.1, 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
{'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}	0.630505209463859	0.0338618111952958	35	0.623711340206186	0.376288659793814
{'C': 100, 'gamma': 1, 'kernel': 'linear'}	0.735980573041025	0.0151759333451596	12	0.716494845360825	0.283505154639175
{'C': 100, 'gamma': 1, 'kernel': 'rbf'}	0.594364553939657	0.0332559478693312	37	0.584192439862543	0.415807560137457

Figura 2: Métricas de distintos modelos knn entrenados

- **Precisión (*precision*):** Qué proporción de las predicciones positivas son correctas.
- **Exhaustividad (*recall*):** Qué proporción de los casos positivos reales fueron identificados correctamente.

Se almacenan los resultados en ficheros CSV separados (`svc_all_models_metrics.csv` y `knn_all_models_metr` lo que permite comparar de manera sistemática el rendimiento de todos los modelos generados por la búsqueda en rejilla y facilita el análisis posterior de las mejores configuraciones.

Este enfoque garantiza que los modelos sean evaluados de manera objetiva y reproducible, proporcionando una visión clara de cómo cada combinación de parámetros afecta al rendimiento en datos no vistos.

```
Mejores parámetros: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Mejor accuracy CV: 0.7455746689819839
Accuracy SVC: 0.718213058419244
Mejores parámetros KNN: {'metric': 'manhattan', 'n_neighbors': 11, 'weights': 'distance'}
Mejor accuracy CV KNN: 0.662204254395485
Accuracy KNN: 0.6460481099656358
Matriz de confusión SVC:
[[206  48   0   0]
 [ 62 191   9   0]
 [  1  40  21   0]
 [  0   1   3   0]]

Reporte de clasificación SVC:
              precision    recall  f1-score   support

           0       0.77      0.81      0.79       254
           1       0.68      0.73      0.70       262
           2       0.64      0.34      0.44        62
           3       0.00      0.00      0.00         4

    accuracy                           0.72       582
   macro avg       0.52      0.47      0.48       582
weighted avg       0.71      0.72      0.71       582

Matriz de confusión KNN:
[[208  46   0   0]
 [ 97 157   8   0]
 [  7  44  11   0]
 [  0   1   3   0]]

Reporte de clasificación KNN:
              precision    recall  f1-score   support

           0       0.67      0.82      0.73       254
           1       0.63      0.60      0.62       262
           2       0.50      0.18      0.26        62
           3       0.00      0.00      0.00         4

    accuracy                           0.65       582
   macro avg       0.45      0.40      0.40       582
weighted avg       0.63      0.65      0.63       582
```

Figura 3: Matriz de confusión para resultados finales.

4. Conclusiones

Tras la ejecución de la práctica y el entrenamiento de los modelos de clasificación, se pueden extraer varias conclusiones relevantes sobre el comportamiento de los algoritmos y la calidad de los datos.

En primer lugar, respecto al modelo **SVC**, la búsqueda en rejilla (*GridSearchCV*) indicó que los mejores parámetros corresponden a un clasificador lineal con `C = 0.1` y `gamma = 'scale'`. Al evaluar el modelo sobre el conjunto de prueba, se obtuvo una exactitud de 0,718, un error del 28 % y métricas de *precision* y *recall* de 0,521 y 0,470, respectivamente. Estos resultados muestran que el modelo SVC lineal es capaz de capturar correctamente la mayor parte de la información relevante para clasificar las notas finales de los estudiantes.

Por otro lado, el modelo **K-Nearest Neighbors (KNN)** presentó un desempeño inferior en comparación con SVC. Los mejores parámetros obtenidos fueron `n_neighbors = 11`, `weights = 'distance'` y `metric = 'manhattan'`. Evaluar sobre el conjunto de prueba se obtuvo una exactitud de 0,646. Este resultado refleja que KNN, aunque útil para captar patrones locales, es más sensible a la distribución de los datos y a la presencia de ruido, mostrando un desempeño

general menor frente al SVC lineal.

La principal limitación observada en ambos modelos es el fuerte desbalance de clases presente en los datos: las clases 0 y 1 dominan ampliamente el conjunto, mientras que la clase 2 tiene pocos ejemplos y la clase 3 es extremadamente minoritaria (solo 4 casos en el conjunto de prueba). Esto provoca que ambos modelos fallen casi por completo en las clases minoritarias, especialmente en la 3.

Analizando la tabla de resultados de SVC, se observa que los kernels no lineales (`rbf`) tienden a producir menor exactitud y mayor variabilidad, especialmente con valores altos de `gamma`, donde el modelo sobreajusta y su desempeño en test se deteriora significativamente. Esto refuerza la idoneidad de un kernel lineal para este conjunto de datos, posiblemente debido a que las relaciones entre características y la etiqueta son mayormente lineales o aproximadamente lineales.

En conclusión, la práctica demuestra que:

- La correcta limpieza y normalización de los datos es crucial para obtener modelos consistentes y comparables.
- El modelo SVC lineal supera a KNN en exactitud y estabilidad, siendo más adecuado para este conjunto de datos.
- La selección de hiperparámetros mediante *GridSearchCV* y validación cruzada permite identificar configuraciones robustas, evitando sobreajuste y mejorando la generalización.

En definitiva, esta práctica proporciona un ejemplo completo del flujo de trabajo en aprendizaje automático: desde la preparación de datos, pasando por la selección y ajuste de modelos, hasta la evaluación sistemática mediante métricas objetivas.

5. Bibliografía

- https://scikit-learn.org/stable/
- https://pandas.pydata.org/
- Python Data Science Handbook - *Jake VanderPlas*
- Machine Learning Pocket Reference - *Matt Harrison*