Universidad Torcuato Di Tella
Maestría en Economía & Finanzas
Segundo Trimestre 2025

Machine Learning para Finanzas
Problem Set 1

Alumna: Paola Nuñez
Profesor: Lionel Modi

---

1 Introducción

Este trabajo presenta `EstimadorOLS`, un implementador propio de regresión lineal de mínimos cuadrados ordinarios (OLS), desarrollado para ser compatible con la API de scikit-learn [1]. El estimador cuenta con la opcionalidad de incluir o no el intercepto y garantiza la validación de los inputs con `validate_data`, así como la integración con `get_params()` y `set_params()`. Para validar su correcto funcionamiento se incorporaron pruebas que cubren tanto el ajuste con constante, sin constante, como escenarios de error de dimensiones.

La implementación respeta las convenciones de docstrings de NumPy y el estilo PEP8.

2 Estructura del proyecto

El código fuente completo está disponible en GitHub: https://github.com/nunezpaola/MLforfinance. El repositorio está organizado de la siguiente manera en la carpeta raíz:

- `ps/ps1.py`: definición de la clase `EstimadorOLS`, con sus métodos `fit` y `predict`, validación de entradas y atributos aprendidos.
- `ps/ps1_test.py`: pruebas unitarias que cubren:
  - Ajuste con intercepto.
  - Ajuste sin intercepto.
  - Escenario de error por dimensiones inválidas.
- `ps/ps1_logs.txt`: registros de terminal generados al ejecutar `ps1_test.py` en los tres casos de prueba.
- `mlfin/printing.py`: utilidades para formatear e imprimir los resultados de validación (función `print_validation_results`) y otras utilidades de logging.

3 Documentación

3.1 Estimador

La clase `EstimadorOLS` está definida en `ps/ps1.py`. A continuación se describen sus componentes principales:

3.1.1 Constructor (`__init__`)

El constructor recibe el parámetro opcional

```
fit_intercept:  bool = True
```

que decide si se incluye una columna de unos en la matriz de diseño para estimar el intercepto. No realiza cálculos colaterales, asegurando compatibilidad con los métodos `get_params()` y `set_params()` de scikit-learn.

3.1.2 Método `fit`

El método `fit(X, y, sample_weight=None)`:

1. Valida y convierte `X` e `y` con `validate_data`, asegurando formato, dimensiones y tipo de datos.
2. Verifica que el número de muestras sea adecuado (debe ser mayor o igual al número de variables, más uno si hay intercepto).
3. Construye la matriz de input `X_fit`, añadiendo columna de unos si corresponde.
4. Resuelve la fórmula cerrada de OLS, manejando singularidad si corresponde.
5. Asigna `intercept_` y `coef_` como atributos aprendidos.
6. Retorna la instancia ajustada (`self`).

3.1.3 Método `predict`

El método `predict(X)`:

- Usa `check_is_fitted` para garantizar que el modelo fue entrenado.
- Valida `X` con `validate_data` para mantener compatibilidad con la API.
- Retorna predicciones.

3.2 Pruebas y validación

El módulo `ps1_test.py` incluye:

- Fijación de semilla (`seed=1234`) y generación de datos sintéticos con `make_regression` (1000 muestras, 10 variables, bias=5, noise=10).
- Validación cruzada 5-fold mediante `print_validation_results`, mostrando puntuaciones, media y desviación estándar.
- Comparación de coeficientes originales vs. estimados tras ajuste completo.
- Tres escenarios de prueba configurables via `test_type`:
  - Ajuste con intercepto.
  - Ajuste sin intercepto.
  - Error por dimensiones inválidas.
- Chequeos de API con `check_estimator` de scikit-learn.
- `ps/ps1_logs.txt` contiene los logs de terminal correspondientes.

3.3 Resultados

La muestra sintética generada cuenta con 1000 observaciones de 10 variables, una constante de 5 y desvío estándar del error de 10. Las Tablas 1 y 2 presentan los resultados de validación cruzada y la comparación de coeficientes (con y sin intercepto), respectivamente.

Test	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Std**
Con intercepto	99.83%	99.84%	99.84%	99.82%	99.79%	99.82%	0.02%
Sin intercepto	99.81%	99.79%	99.77%	99.77%	99.74%	99.78%	0.03%

Table 1: Resultados de validación cruzada (5-fold)

Parámetro	**Original**	**Estimado (a)**	**Estimado (b)**
Intercepto	5.00	5.16	—
Coef. 1	91.24	91.60	91.42
Coef. 2	35.27	34.95	35.08
Coef. 3	54.25	54.28	54.60
Coef. 4	94.50	94.21	94.35
Coef. 5	93.60	94.00	94.05
Coef. 6	13.35	13.26	13.32
Coef. 7	99.02	99.42	99.79
Coef. 8	67.12	67.28	67.39
Coef. 9	55.41	55.24	55.19
Coef. 10	88.83	88.97	88.83

Table 2: Comparación de los coeficientes verdaderos generados por `make_regression` (columna "Original") con los estimados por `EstimadorOLS` entrenado **(a)** con intercepto y **(b)** sin intercepto. Todos los valores están redondeados a dos decimales.

References

[1] Scikit-learn Developers Guide, "Rolling your own estimator", `https://scikit-learn.org/stable/developers/develop.html#rolling-your-own-estimator`. Último acceso: Julio 2025.

[2] NumPy Documentation Guide, "Docstring Standard", `https://numpydoc.readthedocs.io/en/latest/format.html`. Último acceso: Julio 2025.

[3] PEP 8 – Style Guide for Python Code, `https://www.python.org/dev/peps/pep-0008/`. Último acceso: Julio 2025.