

**ESCUELA POLITÉCNICA
SUPERIOR DE CÓRDOBA**
Universidad de Córdoba



Petición de tema de trabajo de fin de grado

Tipología de investigación aplicada

Grado en Ingeniería Informática

**Estudio de nuevos frameworks de aprendizaje automático y su
integración con bibliotecas de clasificación ordinal: aplicación a
un problema de predicción de altura de ola.**

Autor

Francisco Javier Perez Castillo

Directores

Javier Sánchez Monedero

David Guijo Rubio



UNIVERSIDAD DE CÓRDOBA

Córdoba, febrero del 2025

Índice

1. Introducción	5
1.1. Aprendizaje automático	5
1.2. Aprendizaje supervisado	5
1.3. La energía undimotriz	6
2. Objetivos	7
3. Antecedentes	8
3.1. Clasificación ordinal	8
3.2. <i>Frameworks</i>	9
3.2.1. <i>Kedro</i>	9
3.2.2. <i>ORCA-Python</i>	9
3.2.3. <i>mord</i>	10
4. Partes del trabajo de fin de grado	11
5. Fases de desarrollo y cronograma	12
5.1. Estudio y análisis del problema	12
5.2. Preprocesamiento de datos	12
5.3. Implementación y experimentación con modelos	12
5.4. Evaluación y análisis de resultados	13
5.5. Cronograma	13
6. Recursos	14
6.1. Recursos humanos	14
6.2. Recursos <i>software</i>	15
6.3. Recursos <i>hardware</i>	15
Bibliografía	16

Índice de tablas

Tabla 5.5.1: planificación horaria

13

1. Introducción

1.1. Aprendizaje automático

En la actualidad, la inteligencia artificial es el tema de moda. El desarrollo de proyectos informáticos relacionados con este tema ha crecido exponencialmente en los últimos años, pero sus fundamentos siguen recayendo mayormente en el aprendizaje automático. El aprendizaje automático, conocido como *machine learning* en inglés, es una rama de la inteligencia artificial que permite a los sistemas aprender a partir de datos, sin necesidad de ser programados paso a paso. Para lograrlo, se analizan datos que pueden estar etiquetados o no, buscando patrones útiles para tareas como predecir, clasificar o agrupar información.

En esta área, existen varias tareas diferentes [1]. El aprendizaje supervisado utiliza datos con etiquetas para identificar patrones y hacer predicciones sobre una variable dependiente, que puede ser continua (velocidad del viento) o discreta (detección del estadio de una enfermedad), mientras que el no supervisado trabaja con datos sin etiquetar para encontrar relaciones entre estos (generalmente de similitud). También existe el aprendizaje semisupervisado, que combina ambos enfoques, y el aprendizaje por refuerzo, donde el sistema aprende interactuando con un entorno y recibiendo recompensas o penalizaciones según su desempeño.

1.2. Aprendizaje supervisado

Como se explicó previamente, el aprendizaje supervisado consta de extraer información de patrones que han sido etiquetados previamente por un experto. Dentro de esta área, podemos considerar dos grupos: tareas de clasificación y tareas de regresión. Dentro del grupo de clasificación, los métodos se centran en asignar o etiquetar los patrones en una categoría concreta. En el grupo de regresión, los métodos asignan un valor continuo en vez de asignar una etiqueta.

A estas dos tareas principales, se añade la clasificación ordinal. Esta es un tipo de clasificación especial donde las etiquetas poseen una relación de orden entre ellas. Este tipo de problemas aparecen en múltiples situaciones, como por ejemplo en encuestas de satisfacción (“malo”, “regular”, “bueno”, “excelente”), en evaluaciones de riesgos (“riesgo bajo”, “riesgo medio”, “riesgo alto”) o en la intensidad de fenómenos naturales (“leve”, “moderado”, “severo”).

En los últimos años, el desarrollo del aprendizaje automático ha sido impulsado por una explosión de librerías y marcos de trabajo (*frameworks* en inglés) en diversos lenguajes de programación, facilitando la implementación y experimentación de modelos. En particular, en *Python* han surgido herramientas como *Scikit-learn*, *mord* y *ORCA-Python*, que permiten abordar tareas de clasificación y regresión ordinal. Asimismo, *frameworks* como *Kedro* han cobrado relevancia al estructurar este tipo de proyectos de manera modular, optimizando tanto el flujo de trabajo como la trazabilidad de los experimentos.

La integración de estas herramientas no solo mejora el desarrollo de modelos, sino que también contribuye a la automatización y escalabilidad del *software* en los proyectos en esta área.

1.3. La energía undimotriz

La energía undimotriz es aquella obtenida a través del movimiento de las olas. Se trata de una fuente renovable inagotable y no intermitente, ya que las olas están en continuo movimiento, a diferencia de la energía solar, que depende de la luz del día. Su aprovechamiento es clave para la diversidad energética y para reducir la dependencia de fuentes no renovables cuando las condiciones lo permiten.

Para predecir la cantidad de energía generada por las olas, es fundamental conocer ciertos parámetros de su comportamiento, como su altura y período de energía. Estos parámetros se obtienen a través de mediciones realizadas por boyas equipadas con sensores en aguas costeras y mar adentro. Sin embargo, la precisión de estos datos puede verse afectada por la naturaleza cambiante del oleaje, eventos climáticos extremos y fallos en la adquisición de información [2]. Es por esto que es fundamental el desarrollo de modelos predictivos para abordar esta problemática. Estas técnicas permiten estimar valores futuros a partir de datos históricos, proporcionando herramientas más robustas para la gestión de la energía undimotriz [3].

2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es la integración de *frameworks* que semi-automatizan las fases de un proyecto de aprendizaje automático, aplicándolo a la predicción de altura de ola.

Para obtener esto, se definen los siguientes objetivos, los cuales se buscan cumplir al final del proyecto:

1. Estudio de métodos de predicción para el caso de caracterización de olas en el contexto de energía undimotriz.
2. Diseño e implementación de un proyecto de aprendizaje automático para la predicción de altura de ola sin el uso de herramientas de automatización, generando un código y resultados iniciales que sirvan como referencia.
3. Aprendizaje de *frameworks* de automatización y estructura de proyectos de aprendizaje automático junto con el estudio de bibliotecas científicas relacionadas con procesamiento y modelado de datos.
4. Implementación del flujo estándar de un proyecto de experimentación de aprendizaje automático utilizando los entornos y bibliotecas mencionados.
5. Evaluación de los resultados experimentales, identificando tanto las ventajas como las limitaciones de estos entornos de desarrollo en el contexto del proyecto.

3. Antecedentes

3.1. Clasificación ordinal

La clasificación ordinal es una variante de la clasificación supervisada, donde las etiquetas de las clases poseen un orden. Esta estructura ordinal es crucial, ya que influye tanto en el comportamiento del modelo como en sus predicciones. A diferencia de la clasificación nominal, donde las categorías no tienen relación entre sí, en la clasificación ordinal es fundamental preservar la relación entre clases.

Para poder abordar este problema, y siguiendo el artículo *Ordinal Regression Methods: Survey and Experimental Study* [4], se presenta la siguiente taxonomía:

- **Naïve Approaches:** consiste en la aplicación de métodos que no están especializados en la clasificación ordinal, perdiendo por tanto la estructura de orden de los datos. El problema se trataría como si fuera de regresión (asignando valores numéricos a las clases) o como clasificación nominal.
- **Ordinal Binary Decompositions:** este enfoque consiste en descomponer el problema en múltiples problemas de clasificación binaria. Se puede realizar de dos formas: con un modelo de salida múltiple único (donde cada salida pertenece o no a una clase mayor de un umbral) o con múltiples modelos independientes (se entrenan varios modelos binarios en paralelo).
- **Threshold Models:** este tipo consiste en proyectar los datos de entrada en un espacio latente, típicamente de una dimensión, en el que los valores continuos se asignan a etiquetas ordinales usando umbrales. Existen varias formas, como el uso de máquinas de vectores soporte, clasificación binaria aumentada, aprendizaje discriminante, modelos de enlace acumulativo, entre otros.

3.2. Frameworks

En el ámbito del desarrollo *software*, un *framework* es un conjunto de herramientas, bibliotecas, reglas y prácticas destinadas para facilitar el desarrollo de los programas. Estos permiten establecer una estructura común para que así los desarrolladores tengan resueltos problemas generales [5]. A continuación, se expondrán *frameworks* de interés que serán usados durante el proyecto.

3.2.1. Kedro

Los marcos de trabajo establecidos en las áreas de ciencia de datos y aprendizaje automático, se centran tanto en cada una de las fases como en el proyecto en general. En nuestro contexto, nos encontramos en este último caso, donde el *framework Kedro* [6] se aplica para estructurar los proyectos con un enfoque en los flujos de trabajo (*pipelines* en inglés), buscando crearlos de forma reproducible y mantenible. Su principio es que un proyecto organizado y modular facilita tanto el desarrollo como la escalabilidad. Por ello, este marco permite mejorar la trazabilidad de los experimentos, asegurar la reproducibilidad de los resultados y optimizar la gestión de datos.

Si bien existen otras herramientas con propósitos similares [7], como *ZenML* [8] y *Metaflow* [9], *Kedro* se distingue por su énfasis en la ingeniería de software, ofreciendo una estructura de proyecto estandarizada y una amplia gama de *plugins* para mejorar la gestión y visualización de *pipelines*. En contraste, *ZenML* prioriza la flexibilidad y la integración con múltiples plataformas, mientras que *Metaflow* se enfoca en simplificar la ejecución de flujos de trabajo, especialmente en entornos basados en AWS (*Amazon Web Services*). En este contexto, *Kedro* representa la opción más adecuada para nuestro proyecto, ya que su enfoque modular y su robustez en la gestión de *pipelines* permiten garantizar la escalabilidad y mantenibilidad a largo plazo, integrando eficientemente varios modelos y la realización de sus pruebas.

3.2.2. ORCA-Python

El *framework ORCA* [10] (*Ordinal Regression and Classification Algorithms*) es un conjunto de métodos de clasificación ordinal implementado en los lenguajes *Matlab* y *Octave*. Fue desarrollado por el grupo de investigación AYRNA [11], perteneciente al Departamento de Informática y Análisis Numérico de la Universidad de Córdoba.

Con el objetivo de ampliar su accesibilidad y facilitar su integración con herramientas modernas de análisis de datos, Iván Bonaque Muñoz desarrolló *ORCA-Python* en su Trabajo de Fin de Grado [12]. Posteriormente, varios alumnos ampliaron este *framework*. Este es una adaptación de *ORCA* al lenguaje *Python*, permitiendo el uso de los métodos originales usando dicho lenguaje. *ORCA-Python* mantiene las funcionalidades del *framework* original, pero se beneficia de las ventajas de *Python*, como su compatibilidad con bibliotecas como *NumPy* [13] y *Scikit-learn* [14]. Gracias a esto, los usuarios pueden usar fácilmente *ORCA* en flujos de trabajo de ciencia de datos, realizar experimentos en clasificación ordinal y desarrollar programas sin la necesidad de utilizar *Matlab* u *Octave*.

3.2.3. *mord*

La librería *mord* [15] (*Multiclass Ordinal Regression*) es una implementación en *Python* de métodos lineales de regresión ordinal. Su compatibilidad con *Numpy*, *Scikit-learn* y otras librerías facilita su integración en entornos de aprendizaje automático, permitiendo su uso en combinación con herramientas de preprocesamiento y evaluación de modelos. Esta librería organiza sus modelos en tres tipos: basados en umbrales, que establecen cortes en una variable para asignar clases; basados en regresión, donde se genera un valor continuo y luego se discretiza en categorías; y basados en clasificación, que aplican restricciones para respetar el orden de las clases.

Gracias a estas características, se podrán usar los modelos de *mord* para abordar el problema de predicción de las olas, asegurando que la información ordinal en los datos sea aprovechada de manera adecuada.

4. Partes del trabajo de fin de grado

Dada la naturaleza del proyecto, con foco en la implementación y obtención de resultados a través de principios científicos, se categoriza en el tipo de Investigación aplicada. Por ende, el documento futuro constará de las siguientes partes:

1. **Introducción:** se introducirá al tema de *frameworks* aplicados al aprendizaje automático, como *Kedro*, junto al uso de librerías científicas aplicadas a la clasificación ordinal. Adicionalmente, se hablará de la problemática de la predicción de características de las olas, indicando sus motivaciones.
2. **Estado de la técnica:** aquí se expondrá los conocimientos necesarios y teoría relacionada con el tema a trabajar. Se mostrará desarrollos previos de este área junto con la parte que se busca abarcar. En lo referente a *Kedro*, se expondrá su uso y estructura, mostrando las características de su implementación.
3. **Formulación del problema y objetivos:** se describirá el problema a tratar junto con su fragmentación en objetivos, para lograr el cumplimiento de este.
4. **Metodología del trabajo:** en este apartado, se encontrará los métodos, diseño y forma de estudio para solventar el problema del proyecto. Se comentarán las tecnologías y herramientas usadas, destacando la automatización, junto con los procedimientos para el tratamiento de las series temporales y el modelado de estos datos.
5. **Desarrollo y experimentación:** se redactará cómo se ha estructurado el proyecto a nivel de código en esta parte, junto con las diferentes pruebas realizadas para validar el modelo. Se indicará el uso concreto de cada *framework* y librería que se haya usado para ello.
6. **Resultados y discusión:** los resultados obtenidos de predicción de altura de ola, junto con otros resultados de interés que estén relacionados con el problema, se expondrán en este punto.
7. **Conclusiones y recomendaciones:** finalmente, se resumirá las ideas más relevantes sacadas del proyecto, con las implicaciones que conllevan, para poder ser aplicadas al problema real.
8. **Bibliografía:** listado de todas las fuentes usadas durante todo el trabajo.

5. Fases de desarrollo y cronograma

5.1. Estudio y análisis del problema

En esta fase se revisarán modelos de clasificación ordinal y técnicas de recuperación de datos faltantes para aplicarlas en caso de ser necesario, asegurando una base sólida para el desarrollo del proyecto. En un estudio más práctico, se conocerá en profundidad el lenguaje *Python*, junto con la herramienta *Kedro* asociada a este lenguaje, buscando con ello cómo estructurar el proyecto. Tras ello, se estudiará la documentación relacionada con las librerías con las que implementaremos los modelos de datos. Una vez abordado lo anterior, se profundizará en el conocimiento teórico necesario para empezar a tratar el problema de predicción de altura de olas.

5.2. Preprocesamiento de datos

Se realizará un preprocesamiento que incluirá la limpieza de datos, gestión de valores faltantes así como otras técnicas que sean necesarias. Este proceso garantizará la calidad y adecuación de los datos para su uso en modelos predictivos, permitiendo una implementación más precisa y robusta.

5.3. Implementación y experimentación con modelos

El lenguaje *Python* será el lenguaje con el que se implementará la solución al problema. *Python* cuenta con un amplio abanico de librerías que resuelven directamente problemas de codificación presentados en este ámbito. Al ser un lenguaje interpretado, permite ejecutar y probar diferentes partes del código de manera incremental, evitando largos tiempos de compilación. Se utilizarán los modelos de bibliotecas mencionadas, así como el uso de otras en diversas partes del proyecto.

En lo referente a la estructura interna del proyecto, se realiza a través de la herramienta *Kedro*. Este *framework*, nos permitirá estructurar el proyecto de forma modular y reutilizable. Esto junto con la posibilidad de crear *pipelines* de datos, supondrá la capacidad de automatizar diversas partes del proyecto, evitar re-ejecutar partes innecesarias, controlar dependencias y tener una mejor trazabilidad del programa.

5.4. Evaluación y análisis de resultados

En el ámbito de las pruebas, se comprobará el correcto funcionamiento de los modelos, así como la evaluación de los resultados que se obtengan. El objetivo es que los modelos obtengan el mejor rendimiento posible.

La incorporación de *Kedro* en este ámbito permite la realización de pruebas de varias partes del código a través de las *pipelines*, permitiendo un rápido desarrollo de las pruebas. También se utilizará para probar diferentes configuraciones, tanto como para los modelos como para rutas de datos. Todo esto con el objetivo de conseguir una mejor predicción en los modelos.

5.5. Cronograma

La planificación temporal tendrá como objetivo establecer un uso eficiente del tiempo para este proyecto, buscando realizar todos los elementos mencionados en las fases de desarrollo. En la tabla 5.5.1 se muestra la estimación horaria junto con sus fases.

Fase de desarrollo	Mes 1	Mes 2	Mes 3	Mes 4	Horas estimadas
<i>Estudio y análisis del problema</i>	30	20	0	0	50
<i>Preprocesamiento de datos</i>	20	30	10	0	60
<i>Implementación y experimentación con modelos</i>	0	30	40	30	100
<i>Evaluación y análisis de resultados</i>	0	0	20	10	30
<i>Documentación</i>	10	10	10	30	60
Total	60	90	80	70	300

Tabla 5.5.1: planificación horaria

6. Recursos

6.1. Recursos humanos

Los recursos humanos del proyecto son:

Autor: Francisco Javier Perez Castillo

Alumno del Grado de Ingeniería informática.

- **e-mail:** i72pecaf@uco.es
- **Titulación:** Grado en Ingeniería Informática
- **Mención:** computación

Director: D. Javier Sánchez Monedero

Investigador Doctor en la Universidad de Córdoba. Miembro investigador del grupo AYRNA.

- **e-mail:** jsanchezm@uco.es
- **Departamento:** Informática y Análisis Numérico
- Escuela Politécnica Superior de Córdoba
- Universidad de Córdoba

Director: D. David Guijo Rubio

Profesor Permanente Laboral en la Universidad de Córdoba. Miembro investigador del grupo AYRNA.

- **e-mail:** dguijo@uco.es
- **Departamento:** Informática y Análisis Numérico
- Escuela Politécnica Superior de Córdoba
- Universidad de Córdoba

6.2. Recursos *software*

Los recursos software que se usarán son:

- **Windows 10:** sistema operativo para el desarrollo, pruebas y generación de documentación.
- **Python:** lenguaje de programación.
- **Kedro:** *framework* de *Python* para la orquestación de proyectos de ciencia de datos .
- **ORCA-Python:** *framework* de *Python* formado por conjunto de métodos de clasificación ordinal.
- **mord:** librería de *Python* que implementa métodos lineales de regresión ordinal.
- **Git:** *software* de control de versiones.
- **Visual Studio Code:** entorno de desarrollo integrado.
- **Herramientas ofimáticas:** *Google Docs*, *Word*, *Latex* y otras herramientas de generación de gráficos.

6.3. Recursos *hardware*

Como recursos hardware, se utilizará el ordenador del proyectista, cuyas características son las siguientes:

- Intel ® Core (TM) i7-7700 CPU @ 3.60 GHz (8 CPUs)
- Memoria RAM 16GB DDR4
- Controlador gráfico GeForce GTX 1060 3GB GDDR5
- Disco duro Toshiba HDWD110 1TB + SSD Kingston SA40 1TB

Bibliografía

[1] **C. M. Bishop**, «Pattern Recognition and Machine Learning», *Springer*, 2006, pp. 2-10.

[2] **J. Thomson, A. Fisher, y C.J. Rusch**, «The Next Wave: Buoy Arrays for Deterministic Wave Prediction in Real-time», *Nature Communications*, 2023.

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://www.nature.com/articles/s43247-023-00819-0>.

[3] **D. Guijo-Rubio, A. M. Gómez-Orellana, P. A. Gutiérrez, y C. Hervás-Martínez**, «Short- and long-term energy flux prediction using Multi-Task Evolutionary Artificial Neural Networks», *Ocean Engineering*, vol. 216, 2020, p. 108089.

[4] **P. A. Gutiérrez, M. Pérez-Ortiz, J. Sánchez-Monedero, F. Fernández-Navarro, y C. Hervás-Martínez**, «Ordinal Regression Methods: Survey and Experimental Study», *IEEE Transactions on Knowledge and Data Engineering*, 2016.

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://ieeexplore.ieee.org/document/7161338>.

[5] «IEEE Standards Association, ISO/IEC/IEEE 42010:2011 - Systems and software engineering — Architecture description», 2011.

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://ieeexplore.ieee.org/document/6129467>.

[6] *Sitio oficial de Kedro*.

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://kedro.org/>.

[7] «Kedro vs ZenML vs Metaflow: How to Choose the Best MLOps Framework for Your Needs», *Neptune.ai*, 2023.

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://neptune.ai/blog/kedro-vs-zenml-vs-metaflow>.

[8] *Sitio oficial de ZenML*.

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://www.zenml.io/>.

[9] *Sitio oficial de Metaflow.*

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://metaflow.org/>.

[10] **J. Sánchez-Monedero, P. A. Gutiérrez, y M. Pérez-Ortiz**, «ORCA: A Matlab/Octave Toolbox for Ordinal Regression», *Journal of Machine Learning Research*, vol. 20, 2019, pp. 1-4.

[11] *Departamento de Informática y Análisis Numérico de la Universidad de Córdoba, Aprendizaje y redes neuronales artificiales.*

[En línea. Último acceso: 05-02-2025].

Disponible en: <https://www.uco.es/grupos/ayrna/index.php/es>.

[12] **Ivan Bonaque Muñoz, P. A. Gutiérrez Peña, y J. Sánchez-Monedero**, «Trabajo de Fin de Grado. Framework en Python para problemas de clasificación ordinal», 2019.

[13] *Sitio oficial de NumPy.*

[En línea. Último acceso: 06-02-2025].

Disponible en: <https://numpy.org/>.

[14] *Sitio oficial de Scikit-Learn.*

[En línea. Último acceso: 06-02-2025].

Disponible en: <https://scikit-learn.org/stable/>.

[15] *Sitio oficial de mord.*

[En línea. Último acceso: 06-02-2025]

Disponible en: <https://pythonhosted.org/mord/>.