



ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

D. Raúl Darío Marcelino Docío

para a obtención do Título de Graduado en Enxeñaría Informática

Mytilus: Herramienta predictiva de cierre de rías gallegas



Setembro, 2024

Traballo de Fin de Grao N°: 23/24-95

Titor/a: María Encarnación González Rufino, Lorena Otero Cerdeira

Área de coñecemento: Linguaxes e Sistemas Informáticos

Departamento: Informática

Índice de contenidos

1. Introducción.....	6
2. Objetivos.....	7
2.1. Investigación previa.....	7
2.2. Recopilación y procesamiento de datos.....	7
2.3. Investigación sobre algoritmos de aprendizaje automático.....	7
2.4. Obtención de predicciones.....	8
2.5. Presentación de los resultados.....	8
2.6. Despliegue de la herramienta.....	8
3. Resumen de la solución propuesta.....	8
4. Planificación y seguimiento.....	9
4.1. Evolución del propósito de la herramienta.....	11
4.1.1. Versión 1 - Predicción de cantidad de mejillón recogida.....	11
4.1.1.1. Versión 1 - Sprint 0: Investigación previa y especificación de requisitos.....	11
4.1.1.2. Versión 1 - Sprint 1: Obtención y procesamiento de datos.....	13
4.1.2. Versión 2 - Predicción de probabilidad de cierre de rías por toxina, índice de afloramiento y fecha.....	16
4.1.2.1. Versión 2 - Sprint 0: Investigación previa y especificación de requisitos.....	16
4.1.2.2. Versión 2 - Sprint 1: Obtención y procesamiento de datos.....	17
4.1.2.3. Versión 2 - Sprint 2: Estudio de modelos de predicción.....	20
4.1.2.4. Versión 2 - Sprint 3: Generación de predicciones.....	20
4.1.3. Versión 3 - Predicción de probabilidad de cierre de rías por fecha.....	23
4.1.3.1. Versión 3 - Sprint 0: Investigación previa y especificación de requisitos.....	23
4.1.3.2. Versión 3 - Sprint 1: Obtención y procesamiento de datos.....	23
4.1.3.3. Versión 3 - Sprint 2: Estudio de modelos de predicción.....	24
4.1.3.4. Versión 3 - Sprint 3: Generación de predicciones.....	24
4.1.3.5. Versión 3 - Sprint 4: Creación de la interfaz web.....	27
4.1.3.6. Versión 3 - Sprint 5: Implementación en servidor.....	29
4.2. Distribución de tiempo por sprints.....	29
4.2.1. Sprint 0: Investigación previa y especificación de requisitos.....	30
4.2.2. Sprint 1: Obtención y procesamiento de datos.....	32
4.2.3. Sprint 2: Estudio de modelos de predicción.....	33
4.2.4. Sprint 3: Generación de predicciones.....	33
4.2.5. Sprint 4: Creación de la interfaz web.....	34
4.2.6. Sprint 5: Implementación en servidor.....	35
4.7. Revisión y evaluación de la planificación.....	36
5. Arquitectura.....	36

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

5.1. Capa de recolección.....	37
5.2. Capa de procesamiento.....	38
5.3. Capa de predicción.....	38
5.4. Capa de presentación.....	38
6. Tecnologías e integración de productos de terceros.....	39
6.1 Tecnologías y librerías.....	39
6.1 Productos de terceros.....	41
7. Especificación y análisis de requisitos.....	43
7.1. Requisitos funcionales.....	43
7.2. Requisitos no funcionales.....	43
8. Diseño de software.....	44
9. Gestión de datos e información.....	47
10. Pruebas llevadas a cabo.....	49
10.1. Pruebas unitarias.....	50
10.2. Pruebas de integración.....	52
10.3. Pruebas de sistema.....	52
11. Manual de usuario.....	53
12. Principales aportaciones.....	53
13. Conclusiones.....	54
13.1. Conclusiones técnicas.....	54
13.2. Conclusiones personales.....	54
14. Vías de trabajo futuro.....	55
15. Referencias.....	56
16. Anexos.....	58
16.1. Consentimiento de uso de datos.....	58
16.2. Manual de usuario.....	59
16.2.1 Requisitos mínimos.....	59
16.1.1 Software.....	59
16.1.2. Hardware.....	59
16.2.2. Manual de instalación.....	59
16.2.2.1 Instalación de requisitos de software mínimos.....	59
16.2.2.2 Despliegue en Windows.....	60
16.2.2.3. Despliegue en Linux.....	63
16.2.3 Manual de uso.....	64
16.2.3.1. Mediante servidor web.....	64
16.2.3.2. Simulando comportamiento del servidor en equipo local.....	65

Índice de ilustraciones

Imagen 1. Diagrama del flujo de trabajo de Scrum.....	11
Imagen 2. Ejemplo de estado de las rías del 2023.....	14
Imagen 4. Visor de datos de índice de afloramiento.....	19
Imagen 5. Ejemplo de comparación entre valores reales y predichos con regresión logística.....	20
Imagen 6. Ejemplo de comparación entre valores reales y predichos con árbol de decisión.....	21
Imagen 7. Ejemplo de comparación entre valores reales y predichos con bosque aleatorio.....	21
Imagen 8. Ejemplo de comparación entre valores reales y predichos con máquina de soporte vectorial.....	21
Imagen 9. Ejemplo de comparación entre valores reales y predichos con redes neuronales.....	21
Imagen 10. Importancia de las características en la predicción del estado de rías.	22
Imagen 11. Ejemplo de comparación entre valores reales y predichos con regresión logística.....	26
Imagen 12. Ejemplo de comparación entre valores reales y predichos con regresión logística.....	26
Imagen 13. Ejemplo de comparación entre valores reales y predichos con regresión logística.....	26
Imagen 14. Ejemplo de comparación entre valores reales y predichos con regresión logística.....	27
Imagen 15. Formulario de iteración de rías.....	28
Imagen 16. Gráfico de probabilidad de cierre mostrado en interfaz web.....	28
Imagen 17. Información relativa a métricas mostrada en interfaz web.....	28
Imagen 18. Reparto del tiempo dedicado a cada sprint.....	36
Imagen 19. Diagrama de la arquitectura del sistema.....	37
Imagen 20. Interacción entre los componentes de Mytilus.....	44
Imagen 21. Ejemplo archivo diascierre_ayer.csv.....	45
Imagen 22. Ejemplo archivo abiertocerrado_scrap.csv.....	45
Imagen 23. Ejemplo archivo abiertocerrado1.csv.....	45
Imagen 24. Ejemplo archivo forecast_A Pobra A.csv.....	46
Imagen 25. Ejemplo archivo forecast_A Pobra A.png.....	46
Imagen 26. Ejemplo archivo porcentaje_rangofechas.csv.....	46
Imagen 27. Ejemplo de archivo CSV de la herramienta.....	47
Imagen 28. Estructura de directorios de la herramienta.....	48
Imagen 29. Historial de cambios de archivo CSV.....	49
Imagen 30. Consola de Windows.....	60
Imagen 31. Comandos de creación y acceso a carpeta.....	60

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Imagen 32. Clonado de proyecto.....	61
Imagen 33. Error de permisos de ejecución de scripts.....	61
Imagen 34. Entorno virtual activado.....	61
Imagen 35. Dependencias instaladas.....	62
Imagen 36. Ejecución de app.py.....	62
Imagen 37. Resultado de la ejecucion de app.py.....	63
Imagen 38. Interfaz web de Mytilus.....	63
Imagen 39. Interfaz web.....	64
Imagen 40. Interfaz web con gráfico de Villagarcía A.....	65
Imagen 41. Formulario de iteración de rías.....	65
Imagen 43. Ejecución de script scrap24h.py.....	66
Imagen 44. Archivo abiertocerrado_scrap.csv.....	67
Imagen 45. Resultados de la ejecución de forecast24h.py.....	67
Imagen 46. Ejecución de app.py.....	67
Imagen 47. Resultado de la ejecución de integrate3m.py.....	68
Imagen 48. Interfaz web de Mytilus.....	68

Índice de tablas

Tabla 1. Ejemplo de factores medioambientales de la ría de Bueu.....	13
Tabla 2. Ejemplo de estado de la ría de Cangas B.....	15
Tabla 3. Ejemplo de albaranes de venta de mejillón.....	15
Tabla 4. Ejemplo de histórico de toxinas.....	18
Tabla 5. Ejemplo de índices de afloramiento.....	18
Tabla 6. Extracto de predicción de la ría A Pobra A.....	25
Tabla 7. Ejemplo de métricas de predicción.....	25
Tabla 8. Planificación inicial en sprints.....	31
Tabla 9. Product Backlog.....	32
Tabla 10. Sprint Backlog del Sprint 1.....	32
Tabla 11. Sprint Backlog del Sprint 2.....	33
Tabla 12. Sprint Backlog del Sprint 3.....	34
Tabla 13. Sprint Backlog del Sprint 4.....	34
Tabla 14. Sprint Backlog del Sprint 5.....	35

1. Introducción

En los últimos años, la **acuicultura** ha experimentado un crecimiento significativo, desempeñando un papel fundamental como fuente de alimentos y motor económico en numerosas regiones costeras del mundo. En particular, en **Galicia**, la **producción de mejillones** ha florecido como una industria de vital importancia tanto a nivel local como internacional. En Galicia hay 3337 bateas, que la convierten en la **principal potencia mundial en el cultivo y comercialización del mejillón** [1].

Sin embargo, la producción de mejillones en Galicia no está exenta de desafíos. Se enfrenta a factores ambientales fluctuantes, como cambios de temperatura y calidad en el agua, salinidad, pH y presencia de patógenos, entre otros. Este último quizás sea el más determinante, ya que **define totalmente la posibilidad de poder recolectar mejillón**. Al ser este un fenómeno natural de gran magnitud, no se puede abordar la problemática erradicando la toxina de las rías de Galicia. Estas toxinas son termoestables, por lo que tampoco desaparecen con procesos de depuración, esterilización o cocción [2].

Debido a esto, el **Instituto Tecnológico para el Control del Medio Marino de Galicia (INTECMAR)** juega un papel crucial en la gestión y regulación de la producción de mejillones. INTECMAR es responsable de definir los plazos y procedimientos para la recolección de mejillones, los cuales varían según los niveles de toxinas presentes en las aguas. Este organismo monitorea constantemente las concentraciones de biotoxinas y establece restricciones temporales en las áreas afectadas hasta que los mejillones vuelvan a ser seguros para el consumo. Este sistema de control y vigilancia es esencial para mantener la **calidad y seguridad de los productos marinos de Galicia**, así como para proteger la salud pública y la economía local [3].

A esta **prohibición temporal de la recolección de mejillones en las rías se le conoce como cierre de rías** [4]. Son estuarios costeros típicos de la región de Galicia, debido a la presencia de biotoxinas marinas. Estas toxinas, producidas por microalgas, pueden acumularse en los mejillones y otros mariscos, representando un riesgo para la salud humana si se consumen.

Este **trabajo de fin de grado** se centra en el desarrollo de una **herramienta de predicción de posibilidad de cosecha de mejillón**. Esta herramienta aprovecha datos históricos de cierre de rías combinados con técnicas de aprendizaje automático. Aunque INTECMAR se basa en la monitorización constante de biotoxinas para tomar decisiones inmediatas sobre la recolección, los datos históricos proporcionan un contexto valioso que permite **identificar patrones y tendencias a largo plazo**. Al utilizar estos datos históricos, la herramienta de predicción puede **ofrecer estimaciones sobre el estado futuro de las rías a medio plazo**, lo que contribuirá a la **planificación y gestión eficientes de las operaciones en el sector**.

2. Objetivos

El presente trabajo de fin de grado tiene como **objetivo principal el desarrollo de una herramienta de predicción que genere pronósticos fiables de posibilidades de cierre**

de las rías gallegas, combinando el aprendizaje automático con análisis de datos históricos de cierres de rías.

Para abordar este objetivo principal, se identifican una serie de **objetivos secundarios**:

- *Investigación previa*
- *Recopilación y procesamiento de datos*
- *Investigación sobre algoritmos de aprendizaje automático*
- *Obtención de predicciones*
- *Presentación de resultados*

2.1. Investigación previa

Antes de desarrollar la herramienta, es crucial **investigar el conocimiento actual en el campo**. Esto implica **revisar la literatura** para identificar proyectos y estudios similares sobre el crecimiento del mejillón, **analizar los factores ambientales, biológicos y estacionales que influyen en su crecimiento**, y **comunicarse con expertos** que hayan liderado estudios relevantes. Este enfoque establece una base teórica y práctica sólida, aprovechando al máximo los conocimientos y recursos disponibles en la comunidad científica y profesional.

2.2. Recopilación y procesamiento de datos

Para desarrollar el sistema, es fundamental **recopilar y procesar datos históricos sobre los días en que las rías han estado cerradas**, disponibles en la web de INTECMAR. La recopilación de estos datos detallados es esencial para entrenar y validar el sistema de predicción. Posteriormente, **los datos se transformarán y limpiarán** para asegurar su calidad y coherencia, incluyendo la normalización de formatos y corrección de errores. Además, se automatizará la obtención y actualización de los datos de cierre de cada ría para que el sistema sea autónomo y no dependa de cargas de datos externas.

2.3. Investigación sobre algoritmos de aprendizaje automático

Se **evaluarán diversos algoritmos de aprendizaje automático** que permitan un enfoque en series temporales, como regresión lineal y logística, árboles de decisión, bosques aleatorios, máquinas de soporte vectorial, redes neuronales y Prophet. Se **realizarán pruebas para comparar su precisión y sensibilidad**, evaluando su capacidad para predecir a diferentes plazos (por ejemplo, mensual, trimestral, semestral y anual). Además, se **emplearán técnicas de optimización** de hiperparámetros para garantizar la configuración óptima de los modelos, crucial para mejorar la precisión de las predicciones de cierres de rías.

2.4. Obtención de predicciones

Se desarrollarán modelos predictivos utilizando el algoritmo de aprendizaje automático seleccionado y datos históricos de cierres de rías. Estos modelos buscarán prever probabilidades de cierre basadas en patrones y tendencias aprendidos de las variables relevantes. Posteriormente, los modelos evaluados se implementarán en una herramienta práctica, para demostrar su utilidad y eficacia en las predicciones de cierre de rías.

2.5. Presentación de los resultados

Se creará una aplicación web dedicada para presentar los resultados de las predicciones sobre los estados de las rías de manera accesible a usuarios de todos los niveles técnicos. Esto incluirá la generación de gráficos individuales para cada ría, destacando rangos de probabilidades con diferentes colores para una visualización clara y comprensible. Además, se desarrollará una interfaz web intuitiva que permitirá a los usuarios explorar fácilmente los resultados, alternando entre diferentes rías y mostrando detalles como el porcentaje de precisión histórica y el periodo de datos utilizados para entrenar los modelos predictivos.

2.6. Despliegue de la herramienta

Para asegurar la disponibilidad y accesibilidad, se desplegará la herramienta en un servidor. Esto incluye la configuración del servidor para ser accesible desde el exterior, mantener la herramienta activa siempre y el desarrollo y automatización de scripts de ejecución de la herramienta.

3. Resumen de la solución propuesta

Con el fin de alcanzar los objetivos previamente mencionados en el punto anterior de este documento, se propone la creación de una herramienta robusta y avanzada, estructurada en una arquitectura de capas. Cada capa estará compuesta por scripts en *Python*, que se ejecutarán periódicamente y se encargará de una función específica dentro del sistema. Además, la herramienta contará con una interfaz web que integrará y mostrará los resultados generados por estos scripts.

Estos scripts están diseñados para extraer información de la página web de Intecmar, dónde aparecen los días de cierre por ría [5], almacenar estos datos extraídos, entrenar el modelo de predicción, generar métricas relevantes para los usuarios, generar predicciones futuras de probabilidades de cierre de rías y crear gráficos que muestren de una forma clara estas predicciones.

Por otro lado, la interfaz web se encarga de mostrar los gráficos creados, permitiendo iterar entre las diferentes rías para mostrar gráficos específicos para cada una, y presentar tanto el porcentaje de precisión histórica de las predicciones como el rango de fechas utilizado para el entrenamiento.

4. Planificación y seguimiento

Para la elaboración de este proyecto se siguió el marco de trabajo conocido como Scrum [6]. Scrum es un proceso ágil que se basa en la colaboración, la autoorganización y la entrega iterativa de productos de software.

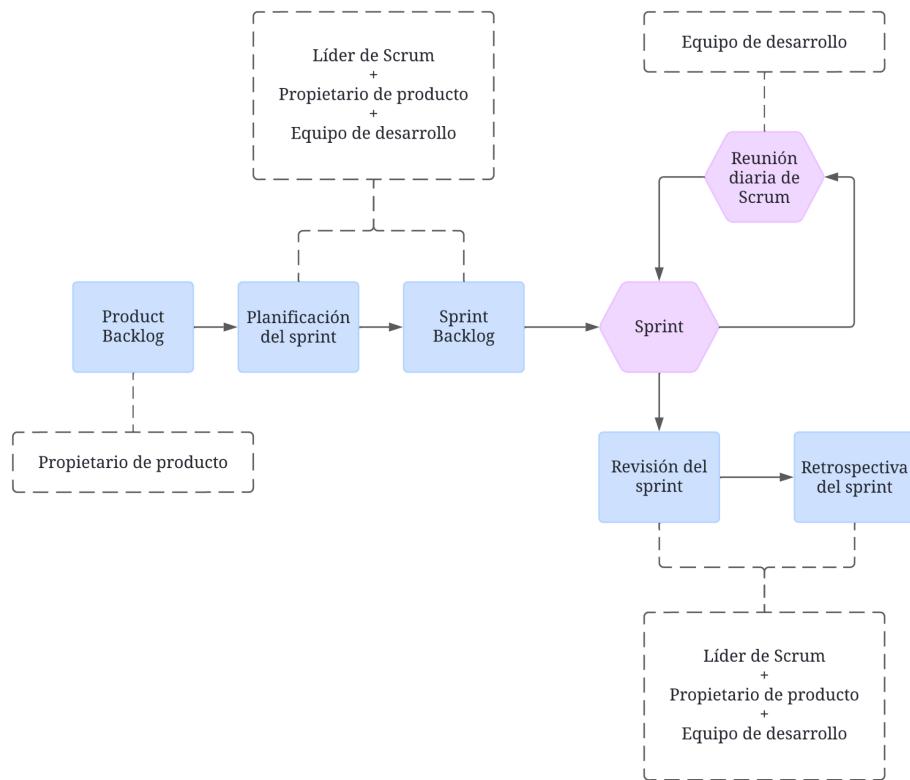


Imagen 1. Diagrama del flujo de trabajo de Scrum

La esencia de Scrum es un **equipo organizado** por cuenta propia que aporta valor al cliente en un período limitado conocido como **Sprint**. Scrum define los artefactos, los roles y los eventos asociados a cada *sprint*.

Los **equipos de trabajo de Scrum** utilizan herramientas denominadas **artefactos de Scrum** para resolver problemas y administrar los proyectos. Los artefactos de Scrum proporcionan información crítica de planificación y sobre la tarea tanto a los miembros del equipo de trabajo como a las partes interesadas. Estos son los tres principales artefactos:

- 1. Product Backlog:** Consiste básicamente en la lista de tareas del equipo, que se revisa constantemente para ajustar las prioridades y adaptarse a los cambios que se producen en el mercado. El propietario del producto es quien mantiene y actualiza la lista. Está encargado de eliminar los elementos irrelevantes o de agregar nuevas solicitudes realizadas por los clientes.
- 2. Sprint Backlog:** Se refiere a una lista de los elementos que el equipo de desarrollo debe completar durante el ciclo *sprint* en curso. Antes de cada *sprint*,

el equipo escoge en qué elementos trabajará a partir del Product Backlog. Un *Sprint Backlog* es flexible y puede evolucionar durante un *sprint*.

3. **Incremento:** Es un avance hacia la consecución de un logro o una visión. Es el producto final y utilizable que surge a partir de un *sprint*. Los equipos pueden adoptar diferentes métodos para definir y demostrar los objetivos del *sprint*. A pesar de la flexibilidad, el objetivo fundamental del *sprint*, lo que el equipo desea alcanzar a partir del *sprint* en curso, no puede verse comprometido.

Un equipo de trabajo de Scrum necesita tres roles específicos: un **propietario de producto, un líder de Scrum y un equipo de desarrollo**.

1. **Propietario de producto:** se concentra en garantizar que el equipo de desarrollo proporcione el mayor valor al negocio. Comprende y prioriza las necesidades cambiantes de los usuarios finales.
2. **Líder de SCRUM:** se encarga de velar por la metodología Scrum dentro de los equipos de trabajo. Son responsables de la efectividad del equipo. Forman a los equipos y a los propietarios de los productos para mejorar los procesos y optimizar la entrega.
3. **Equipo de desarrollo de SCRUM:** está formado por los profesionales con los conocimientos técnicos necesarios. Se encargan de desarrollar el proyecto llevando a cabo las historias de usuario que se comprometen al inicio de *sprint*.

Los **eventos de Scrum** son un conjunto de reuniones secuenciales que los equipos de Scrum sostienen periódicamente. Estos son algunos de los eventos de Scrum:

1. **Planificación del sprint:** durante este evento, el equipo estima el trabajo que se debe completar durante el siguiente *sprint*. Los miembros definen los objetivos del *sprint*, que a su vez deben ser específicos, medibles y viables. Tras la reunión de planificación, cada miembro del equipo de Scrum debe comprender cómo se puede entregar cada incremento durante el *sprint*.
2. **Sprint:** periodo durante el cual el equipo de Scrum trabaja conjuntamente para la consecución de un incremento. Habitualmente un *sprint* dura dos semanas. Sin embargo, puede variar en función de las necesidades del proyecto y del equipo. Entre más complejo sea el trabajo y más factores desconocidos haya, el periodo del *sprint* debe ser más corto.
3. **Reunión diaria de SCRUM:** sesión breve en la que los miembros del equipo se reportan y planifican el día. Informan sobre el trabajo concluido y expresan cualquier desafío para alcanzar los objetivos del *sprint*. Se conoce como reunión de pie porque se busca que esta sea lo más breve posible, como cuando todos están de pie.
4. **Revisión del sprint:** Al finalizar el *sprint*, el equipo se reúne en el marco de una sesión informal para revisar el trabajo concluido y exponerlo ante las partes

interesadas. Es posible que el propietario de producto también ajuste el *Product Backlog* en función del *sprint* en curso.

5. **Retrospectiva del sprint:** El equipo de trabajo se reúne para documentar y hablar sobre qué funcionó y qué no durante el *sprint*. Las ideas generadas se utilizan para mejorar los próximos *sprints*.

Debido al conocimiento previo sobre el mundo del mejillón y al carácter individual de este TFG, Raúl Marcelino Docío, alumno que desarrolla este TFG, **asume todos los roles del equipo de trabajo de Scrum**. También se **adaptan y ajustan los eventos de Scrum** para satisfacer las necesidades de un único individuo.

El uso de Scrum en el desarrollo de **Mytilus** se debe a que permite adaptarse a los cambios en datos y requisitos y en la clara relación de los objetivos con los entregables al final de cada *sprint*.

A continuación, se enumeran los *sprints* y se describe el trabajo realizado en cada uno de ellos:

4.1. Evolución del propósito de la herramienta

Desde un primer momento, el objetivo de este TFG fue **desarrollar una herramienta de predicción de cosechas de mejillón**. Sin embargo, el propósito final de Mytilus fue variando hasta encontrar la forma de que:

- fuera una herramienta **posible de desarrollar con los datos disponibles**
- ofrezca **utilidad** al sector mejillonero
- su desarrollo se encuentre dentro del marco de las **300 horas** asignadas para este TFG.

Debido a esto, la herramienta pasa por **3 versiones diferentes** que se detallan a continuación:

4.1.1. Versión 1 - Predicción de cantidad de mejillón recogida

4.1.1.1. Versión 1 - Sprint 0: Investigación previa y especificación de requisitos

Inicialmente, Mytilus **iba a predecir la cantidad de mejillón recogido en determinada fecha**. La idea de que la predicción fuese de esta forma es la necesidad que surge en los bateeiros de estimar ventas y períodos de prohibición de cosecha, para así optimizar recursos y obtener un mayor rendimiento económico.

Para determinar la cantidad de mejillón recogido en una fecha específica, era fundamental **identificar y comprender los factores que influyen significativamente en el crecimiento de los mejillones**.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

El resultado de la investigación concluyó que la variable más crucial para el ciclo de vida del mejillón es la **disponibilidad de fitoplancton en el agua** [7]. Una mayor concentración de fitoplancton favorece el crecimiento del mejillón, lo que contribuye a una mejor calidad, un precio más alto, un mayor peso y una reducción en la mortalidad.

Los factores que influyen en la **concentración de fitoplancton** son diversos y se detallan a continuación:

- **Estacionalidad:** la concentración de fitoplancton varía con las estaciones.
- **Temperatura del agua:** la temperatura del agua afecta a la tasa metabólica y al crecimiento del fitoplancton. A temperaturas óptimas, el fitoplancton prolifera rápidamente, mientras que las temperaturas extremas pueden limitar su crecimiento.
- **Salinidad:** en general, las variaciones en salinidad significativas pueden ser estresantes para el fitoplancton.
- **pH:** un pH extremo puede ser perjudicial para el fitoplancton y afectar su crecimiento y reproducción.
- **luz solar:** es esencial para la fotosíntesis, proceso mediante el cual el fitoplancton obtiene alimento. La cantidad y calidad de luz que penetra en el agua pueden influir significativamente en la producción de fitoplancton.

Además, también serían necesarios **datos relacionados con períodos donde se prohíbe recoger mejillón (cierre de rías)**, para determinar qué fechas sería más probable que hubieran variaciones en las cosechas, e históricos de cosechas de los productores, para poder realizar entrenamientos y comparativas entre las predicciones futuras y los datos reales.

El propósito de utilizar las **fechas de cierre de rías** es que, al conocer el estado de las rías en distintos períodos, podríamos **identificar las condiciones que influyen en la productividad de las cosechas de mejillón**. Esto nos permitiría determinar con mayor precisión las épocas en las que es más probable **obtener cosechas más abundantes o experimentar disminuciones en la producción**.

El uso de datos históricos de cosechas de mejillón sería fundamental para **identificar patrones y tendencias en la producción**, lo que facilitaría una previsión más precisa de futuros rendimientos. Estos datos permitirían modelar y ajustar los algoritmos para optimizar la gestión y la planificación de cosechas. Además, al comparar los resultados actuales con los datos históricos, se puede evaluar la precisión del modelo y detectar anomalías para mejorar continuamente las prácticas de cultivo.

Una vez terminada la investigación de qué factores afectan al crecimiento del mejillón, se dispuso a obtener los **datos relacionados con estos factores, tal y como se detalla en el siguiente apartado**.

4.1.1.2. Versión 1 - Sprint 1: Obtención y procesamiento de datos

Los factores que iban a influir en la predicción eran los **factores ambientales que afectan al crecimiento de mejillón [8]**, los periodos de tiempo dónde se prohíbe recoger mejillón y los históricos de cosechas de productores.

A continuación, se muestran extractos de los datos que se iban a utilizar para realizar predicciones en esta **primera versión de la herramienta**:

Estación	Data	VAR_0	COD_0	VAR_1	COD_1	VAR_2	COD_2	VAR_3	COD_3
Bueu	27/12/2023 9:16:50	14,8404	0	34,8548	0	7,382	0	80,29	0
Bueu	27/12/2023 9:16:50	14,8448	0	34,8548	0	7,383	0	78,48	0
Bueu	27/12/2023 9:16:50	14,8453	0	34,8546	0	7,383	0	86,773	0
Bueu	27/12/2023 9:16:50	14,8449	0	34,8553	0	7,382	0	83,215	0
Bueu	27/12/2023 9:16:50	14,8424	0	34,8551	0	7,383	0	77,833	0
Bueu	27/12/2023 9:16:50	14,8415	0	34,8545	0	7,383	0	69,302	0
Bueu	27/12/2023 9:16:50	14,8432	0	34,854	0	7,382	0	61,649	0
Bueu	27/12/2023 9:16:50	14,843	0	34,8544	0	7,383	0	59,645	0
Bueu	27/12/2023 9:16:50	14,8428	0	34,854	0	7,382	0	60,87	0
Bueu	27/12/2023 9:16:50	14,8422	0	34,8535	0	7,383	0	57,721	0

Tabla 1. Ejemplo de factores medioambientales de la ría de Bueu

En la **Tabla 1** se representa un extracto de los datos obtenidos de factores medioambientales, en este caso, de la ría de Bueu. A continuación se describen la variable representada por cada una de las columnas:

- **Estación:** nombre de la estación a la que pertenece
- **Data:** momento que fue tomada la muestra
- **VAR_0:** temperatura del agua
- **VAR_1:** salinidad
- **VAR_2:** pH
- **VAR_3:** irradiancia

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

- **COD_1, COD_2, COD_3, COD_4:** representan si el valor muestreado es válido o no, representando los números 0, 1 y 2 que el valor es válido, y el 3, 4 y 9 que es erróneo.

Los datos de factores medioambientales provienen de los **sensores de INTECMAR**, concretamente de sus **estaciones oceanográficas**. Los datos recogidos de esos sensores son expuestos mediante una aplicación web [9] , donde además de poder consultar, puedes descargar los datos en formato de archivo de texto, filtrando por estación, parámetros y rango de fechas.

Además, también se obtiene de la web de INTECMAR [4], mediante procesamiento manual, **los datos históricos del estado de la rías**. INTECMAR toma muestras del agua de las rías y actualiza el estado de las rías diariamente. Además, permite visualizar los datos históricos por años de todas las rías, como se muestra en la **Imagen 2**.

Períodos de prohibición de extracción: Ano 2023 (Cultivos en Viveiros Flotantes)

Ares-Betanzos		Xaneiro	Febreiro	Marte	Abril	Mai	Xullo	Xullo	Agosto	Setembro	Outubro	Novembro	Decembro
Sada 1		DSP	DSP	DSP									
Sada 2		DSP	DSP	DSP									

Camarñas		Xaneiro	Febreiro	Marte	Abril	Mai	Xullo	Xullo	Agosto	Setembro	Outubro	Novembro	Decembro
Comurias A		DSP	DSP	DSP									
Comurias B													

Muros-Noia		Xaneiro	Febreiro	Marte	Abril	Mai	Xullo	Xullo	Agosto	Setembro	Outubro	Novembro	Decembro
Muros B		DSP	DSP	DSP									
Muros C		DSP	DSP	DSP									
Muros A		DSP	DSP	DSP									
Noia A		DSP	DSP	DSP									
Noia C		DSP	DSP	DSP									
Muros B		DSP	DSP	DSP									
Muros C		DSP	DSP	DSP									
Muros A		DSP	DSP	DSP									
Muros D		DSP	DSP	DSP									
Muros E		DSP	DSP	DSP									
Muros F		DSP	DSP	DSP									
Muros G		DSP	DSP	DSP									
Muros H		DSP	DSP	DSP									
Muros I		DSP	DSP	DSP									
Muros J		DSP	DSP	DSP									
Muros K		DSP	DSP	DSP									
Muros L		DSP	DSP	DSP									
Muros M		DSP	DSP	DSP									
Muros N		DSP	DSP	DSP									
Muros O		DSP	DSP	DSP									
Muros P		DSP	DSP	DSP									
Muros Q		DSP	DSP	DSP									
Muros R		DSP	DSP	DSP									
Muros S		DSP	DSP	DSP									
Muros T		DSP	DSP	DSP									
Muros U		DSP	DSP	DSP									
Muros V		DSP	DSP	DSP									
Muros W		DSP	DSP	DSP									
Muros X		DSP	DSP	DSP									
Muros Y		DSP	DSP	DSP									
Muros Z		DSP	DSP	DSP									
Muros AA		DSP	DSP	DSP									
Muros BB		DSP	DSP	DSP									
Muros CC		DSP	DSP	DSP									
Muros DD		DSP	DSP	DSP									
Muros EE		DSP	DSP	DSP									
Muros FF		DSP	DSP	DSP									
Muros GG		DSP	DSP	DSP									
Muros HH		DSP	DSP	DSP									
Muros II		DSP	DSP	DSP									
Muros JJ		DSP	DSP	DSP									
Muros KK		DSP	DSP	DSP									
Muros LL		DSP	DSP	DSP									
Muros MM		DSP	DSP	DSP									
Muros NN		DSP	DSP	DSP									
Muros OO		DSP	DSP	DSP									
Muros PP		DSP	DSP	DSP									
Muros QQ		DSP	DSP	DSP									
Muros RR		DSP	DSP	DSP									
Muros SS		DSP	DSP	DSP									
Muros TT		DSP	DSP	DSP									
Muros UU		DSP	DSP	DSP									
Muros VV		DSP	DSP	DSP									
Muros WW		DSP	DSP	DSP									
Muros XX		DSP	DSP	DSP									
Muros YY		DSP	DSP	DSP									
Muros ZZ		DSP	DSP	DSP									
Muros AA		DSP	DSP	DSP									
Muros BB		DSP	DSP	DSP									
Muros CC		DSP	DSP	DSP									
Muros DD		DSP	DSP	DSP									
Muros EE		DSP	DSP	DSP									
Muros FF		DSP	DSP	DSP									
Muros GG		DSP	DSP	DSP									
Muros HH		DSP	DSP	DSP									
Muros II		DSP	DSP	DSP									
Muros JJ		DSP	DSP	DSP									
Muros KK		DSP	DSP	DSP									
Muros LL		DSP	DSP	DSP									
Muros MM		DSP	DSP	DSP									
Muros NN		DSP	DSP	DSP									
Muros OO		DSP	DSP	DSP									
Muros PP		DSP	DSP	DSP									
Muros QQ		DSP	DSP	DSP									
Muros RR		DSP	DSP	DSP									
Muros SS		DSP	DSP	DSP									
Muros TT		DSP	DSP	DSP									
Muros UU		DSP	DSP	DSP									
Muros VV		DSP	DSP	DSP									
Muros WW		DSP	DSP	DSP									
Muros XX		DSP	DSP	DSP									
Muros YY		DSP	DSP	DSP									
Muros ZZ		DSP	DSP	DSP									
Muros AA		DSP	DSP	DSP									
Muros BB		DSP	DSP	DSP									
Muros CC		DSP	DSP	DSP									
Muros DD		DSP	DSP	DSP									
Muros EE		DSP	DSP	DSP									
Muros FF		DSP	DSP	DSP									
Muros GG		DSP	DSP	DSP									
Muros HH		DSP	DSP	DSP									
Muros II		DSP	DSP	DSP									
Muros JJ		DSP	DSP	DSP									
Muros KK		DSP	DSP	DSP									
Muros LL		DSP	DSP	DSP									
Muros MM		DSP	DSP	DSP									
Muros NN		DSP	DSP	DSP									
Muros OO		DSP	DSP	DSP									
Muros PP		DSP	DSP	DSP									
Muros QQ		DSP	DSP	DSP									
Muros RR		DSP	DSP	DSP									
Muros SS		DSP	DSP	DSP									
Muros TT		DSP	DSP	DSP									
Muros UU		DSP	DSP	DSP									
Muros VV		D											

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

- **Día:** día en el que se muestrea el estado de la ría
- **abierto/cerrado:** el estado de la ría, representando el número 0 la posibilidad de recoger mejillón y 1 la prohibición de recoger mejillón.

Fecha	Cliente	Zona	Batea	Kilos
29/12/2015	CULTIVOS MARINOS	2	CAMBONO6	4.160,00
29/12/2015	CULTIVOS MARINOS	2	CAMBONO6	364
05/01/2016	Pescados Marcelino, S.L.	2	CAMBONO1	11.500,00
07/01/2016	Pescados Marcelino, S.L.	2	CAMBONO1	11.840,00
07/01/2016	Pescados Marcelino, S.L.	2	CAMBONO1	12.200,00
08/01/2016	Pescados Marcelino, S.L.	2	CAMBONO1	12.560,00
12/01/2016	Pescados Marcelino, S.L.	2	CAMBONO1	12.460,00
25/01/2016	Pescados Marcelino, S.L.	3	CD7	14.040,00
28/01/2016	Pescados Marcelino, S.L.	3	CD7	14.020,00
28/01/2016	Pescados Marcelino, S.L.	3	CD7	13.360,00
01/02/2016	Pescados Marcelino, S.L.	3	CD7	14.100,00
01/02/2016	Pescados Marcelino, S.L.	3	CD7	12.800,00
03/02/2016	Pescados Marcelino, S.L.	3	CD7	13.340,00
18/10/2016	Pescados Marcelino, S.L.	2	CAMBONO6	14.260,00
14/12/2016	MARES DE GALICIA, S.A.	PORTONOVO	EXPTAMBO3	7.420,00

Tabla 3. Ejemplo de albaranes de venta de mejillón

En la **Tabla 3** se representa un extracto de los albaranes de venta de un productor de mejillón:

- **Fecha:** el día en el que se cosecha
- **Cliente:** el cliente al que se le venden los kilogramos recogidos
- **Zona:** la zona de donde se extrae el mejillón
- **Batea:** de la batea de la que se extrae el mejillón
- **Kilos:** la cantidad de mejillón extraído en kilogramos.

Estos datos provienen de los **registros mantenidos por los productores de mejillón**. Dado que son datos privados, los productores otorgaron su consentimiento para que se utilicen y se presenten en este trabajo de fin de grado, como se muestra en el **anexo 16.1.**

Impedimentos de desarrollo

A pesar de contar con datos abundantes y de alta calidad, **aparecieron obstáculos** que impidieron avanzar con esta primera versión de la herramienta.

En primer lugar, **la falta de datos históricos de cultivo** representó un problema significativo, ya que si no hay constancia de cuánta cría de mejillón se está plantando, no se puede realizar un análisis de tendencias y patrones a lo largo del tiempo. Sin esta información, **la herramienta no podría ofrecer pronósticos óptimos**.

Por otro lado, **la escasez de estudios sobre crecimiento del mejillón** planteó dificultades adicionales. Estos estudios son fundamentales para comprender cómo varían los cultivos bajo diferentes condiciones ambientales y de manejo. Sin esta base de conocimientos, **las predicciones de la herramienta carecían de contexto y precisión, limitando su utilidad para los usuarios finales**.

Estos desafíos combinados obstaculizaron el desarrollo de la herramienta hasta el punto de **no poder continuar con la idea inicial**.

4.1.2. Versión 2 - Predicción de probabilidad de cierre de rías por toxina, índice de afloramiento y fecha

4.1.2.1. Versión 2 - Sprint 0: Investigación previa y especificación de requisitos

Tras no contar con suficientes datos para continuar con la primera versión de la herramienta, que se centraba en predecir la cantidad de cosecha, **se decidió cambiar el enfoque**.

Aprovechando los datos obtenidos en la versión anterior sobre el cierre de rías como los mostrados en el ejemplo de la tabla 2, surgió la idea de que Mytilus **genere predicciones sobre la probabilidad de cierre de rías**. Este nuevo enfoque ofrece apoyo a los trabajadores del sector para gestionar eficientemente sus recursos, considerando la posibilidad de que puedan o no recolectar mejillón. Esto ayuda a reducir gastos y promueve una producción de mejillón más sostenible.

Antes de realizar un estudio sobre los factores que influyen en el cierre de rías, **se realizó una revisión del estado del arte actual en predicción de estados de rías**.

En el momento en el que se desarrolla Mytilus, tan sólo existe un proyecto con una finalidad similar: **Empromar [10]**. Empromar es un servicio de predicción de mareas rojas para las rías gallegas. Su objetivo principal es anticiparse al cierre de una ría determinada 4 días antes de que esto suceda.

Para conocer un poco más en profundidad cómo se generan las predicciones en Empromar, se contactó con **Sergio Ferrol, su fundador**. En Empromar, se usan **índices de afloramiento [11]**, además de otras variables, para predecir la aparición futura de toxinas.

Para determinar los datos necesarios para generar la predicción de cierre de rías, se llevó a cabo un **análisis de los factores que influyen en el cierre de las rías**.

El cierre de una ría **depende totalmente de la concentración de toxinas presente en la misma [12]**. Una vez sobrepasado determinado umbral, INTECMAR ordena el cierre de la ría afectada por la toxina. Cuando el nivel de toxina está por debajo del umbral, INTECMAR vuelve a permitir la cosecha de mejillón.

El **índice de afloramiento** es un factor importante que influye en la aparición de estas toxinas. Mide la intensidad del ascenso de aguas profundas, ricas en nutrientes, a la superficie del océano. Este proceso favorece el crecimiento de microalgas que pueden producir biotoxinas. **Cuando el índice de afloramiento es alto, aumenta la probabilidad de proliferación de estas microalgas tóxicas.**

También, con motivo de entrenamiento y comparación de los datos predichos y reales, sería necesario el **histórico de cierre de rías por fecha** (obtenido previamente en la versión anterior).

Tras finalizar el estudio sobre los factores que impactan en el cierre de rías, se empezó a recolectar los **datos asociados a estos factores**, tal y como se explica en el próximo apartado.

4.1.1.2. Versión 2 - Sprint 1: Obtención y procesamiento de datos

Para llevar a cabo el desarrollo de esta versión, se trató generar predicciones a través de **tres variables:**

- *Histórico del nivel de toxinas exacto de cada ría por día.*
- *Histórico del índice de afloramiento.*
- *Históricos de cierre de rías por fecha.*

Data_Mos traxe	Ria	Subzona	Punto	Cuad rícula	Profundida de	Especie	Equiv PSP	Equiv ASP	Bioensaio	Equiv AO	Equiv AZA	Equiv YTX	Equiv STX
4/1/2010	PON	I.1	CAN GAS-A	6	1-5-10 m.	Mexil lón	< 380	< 2,0	Negativo	---	---	---	---
19/9/2019	PON	I.1	CAN GAS-A	6	1-5-10 m	Mexil lón	---	---	---	60,5	---	---	---
19/9/2019	PON	I.2	CAN GAS-B	5	1-5-10 m	Mexil lón	< 380	< 2,0	---	153,6 ±31,5	< LQ	< LQ	---

Tabla 4. Ejemplo de histórico de toxinas

En la **tabla 4** se muestra un extracto representativo del histórico del nivel de toxinas provisto por INTECMAR.

- **Data_Mostraxe:** día del muestreo.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

- **Ria:** ría muestreada.
- **Subzona:** zona específica muestreada .
- **Cuadrícula:** cuadrícula específica dentro de la ría muestreada .
- **Profundidad:** profundidad a la que se toma la muestra.
- **Especie:** especie afectada.
- **Equiv PSP, ASP, Bioenasio, AO AZA, YTX, STX:** nivel de diferentes toxinas.

Estos datos provienen de INTECMAR, pero a diferencia de los datos de cierre de rías, **no están expuestos al público, a pesar de ser públicos**. Para obtenerlos, se tuvo que realizar una **solicitud formal [13]** por medio del formulario de acceso a datos públicos de la **Xunta**.

fecha	UI
2013-07-01	5741.902
2013-07-02	599.992
2013-07-03	4356.649
2013-07-04	4301.625
2013-07-05	750.194
2013-07-06	109.866
2013-07-07	425.649
2013-07-08	-43.239
2013-07-09	411.826
2013-07-10	1672.602

Tabla 5. Ejemplo de índices de afloramiento

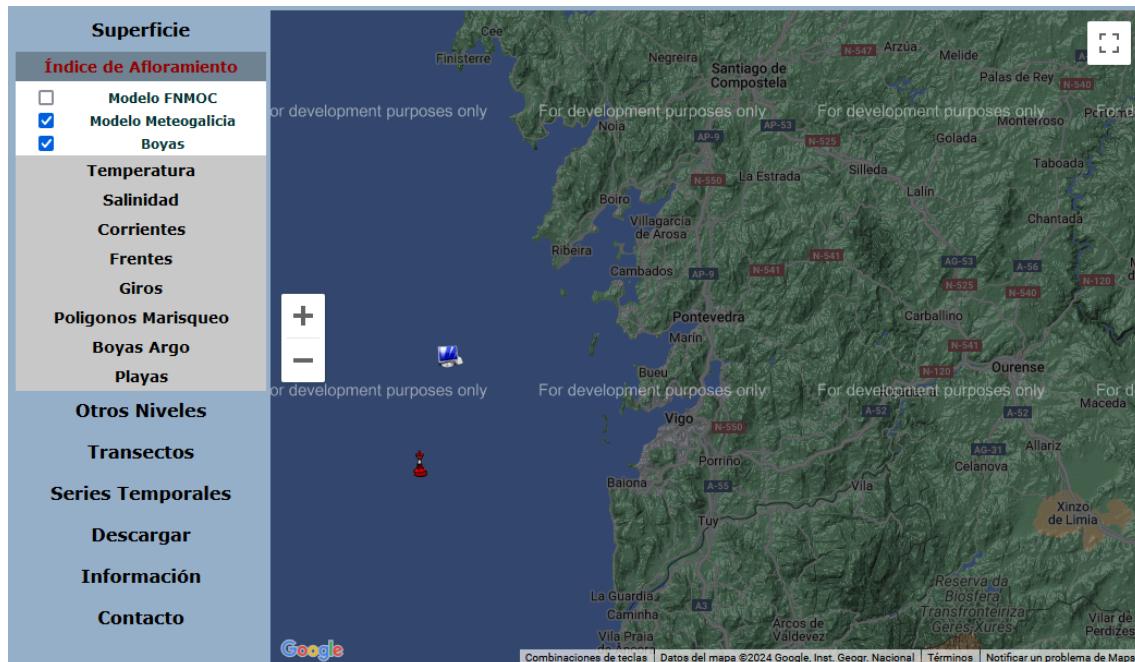
En la **tabla 5** se muestra un extracto representativo del histórico del nivel de toxinas provisto por **indice de afloramiento** [11].

- **fecha:** día del muestreo.
- **UI:** índice de afloramiento.

Estos datos provienen de **indice de afloramiento**. Son expuestos mediante una aplicación web y se pueden descargar en formato de texto. Para obtenerlos, se debe acceder a su web, y en el visor de datos de índices de afloramiento, seleccionar la estación más cercana a la ría que nos interesa y seleccionar el modo de descarga, tal y como se demuestra en la **Imagen 4**.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024



Índice de Afloramiento / Upwelling Index

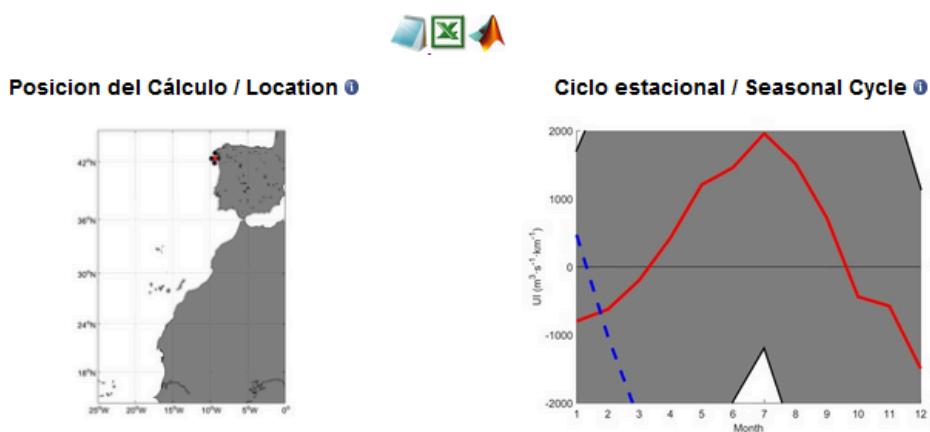


Imagen 4. Visor de datos de índice de afloramiento

Los datos relativos a los **históricos de cierre** de rías fueron obtenidos en el **sprint 1 de la versión 1**, por lo que no fue necesario obtenerlos nuevamente.

Tras obtener todos los datos, se procedió a **estudiar qué modelos y algoritmos de predicción eran los más óptimos para generar las predicciones futuras de cierre de rías**.

4.1.1.2. Versión 2 - Sprint 2: Estudio de modelos de predicción

Para generar predicciones precisas, se llevó a cabo un estudio minucioso para identificar los **algoritmos y modelos de predicción más óptimos**.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Este estudio concluyó que los mejores algoritmos para desarrollar la solución eran los aplicables con **series temporales**. Entre todos ellos, se consideraron:

- **regresión logística:** método de análisis estadístico utilizado para predecir la probabilidad de una variable binaria basada en una o más variables predictoras.
- **árbol de decisión:** modelo predictivo que utiliza un conjunto de reglas de decisión derivadas de los datos.
- **bosques aleatorios:** conjunto de árboles de decisión entrenados con variaciones en los datos y las características.
- **máquinas de soporte vectorial:** modelos de aprendizaje supervisado utilizados tanto para clasificación como para regresión.
- **redes neuronales:** modelos de aprendizaje automático inspirados en la estructura y el funcionamiento del cerebro humano.

Después de seleccionar los modelos a utilizar, se procedió a **generar predicciones y comparaciones entre cada uno de estos modelos**.

4.1.1.2. Versión 2 - Sprint 3: Generación de predicciones

Como se muestra en las siguientes imágenes, las predicciones del cierre de rías en grandes temporalidades tienen una **precisión elevada**:

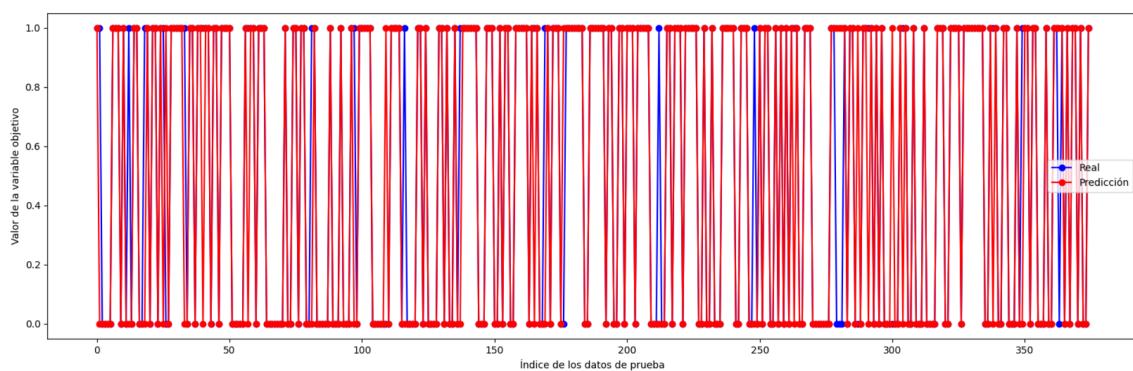
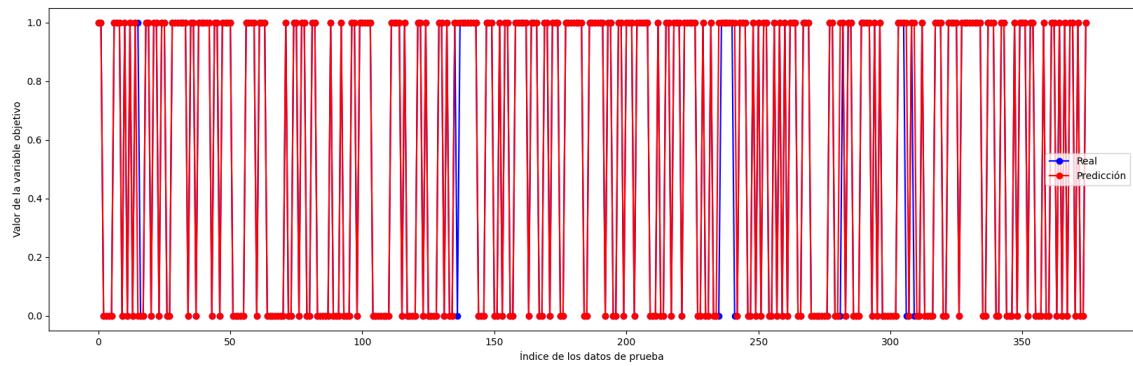


Imagen 5. Ejemplo de comparación entre valores reales y predichos con regresión logística



Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Imagen 6. Ejemplo de comparación entre valores reales y predichos con árbol de decisión

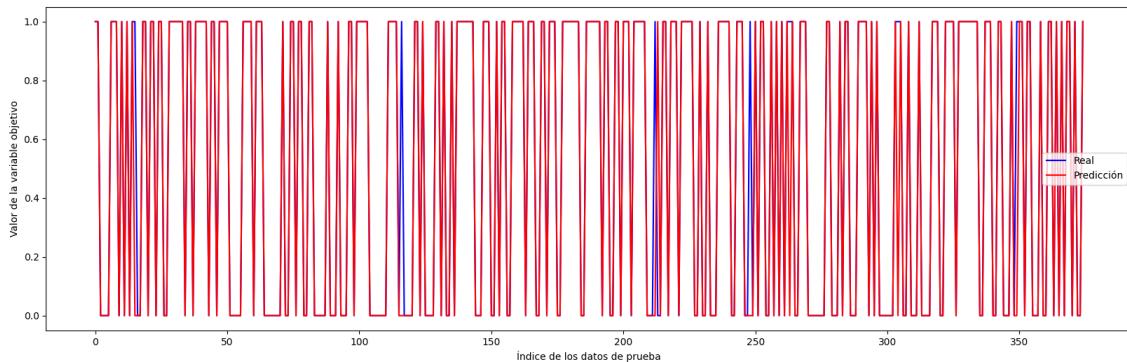


Imagen 7. Ejemplo de comparación entre valores reales y predichos con bosque aleatorio

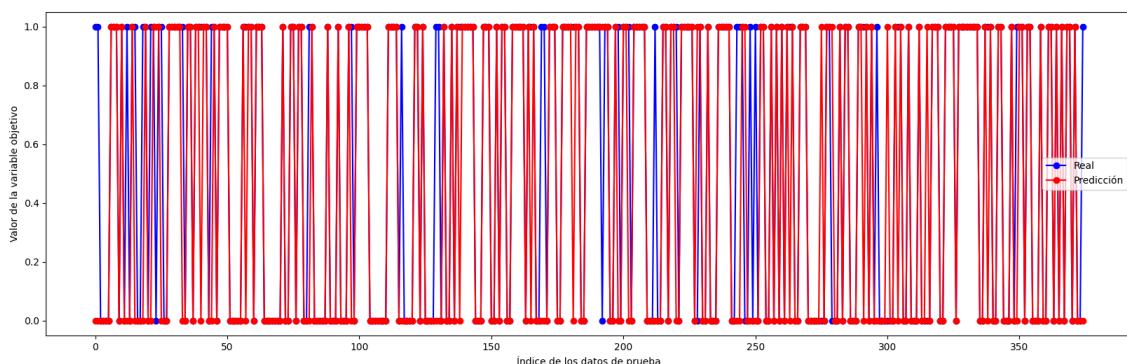


Imagen 8. Ejemplo de comparación entre valores reales y predichos con máquina de soporte vectorial

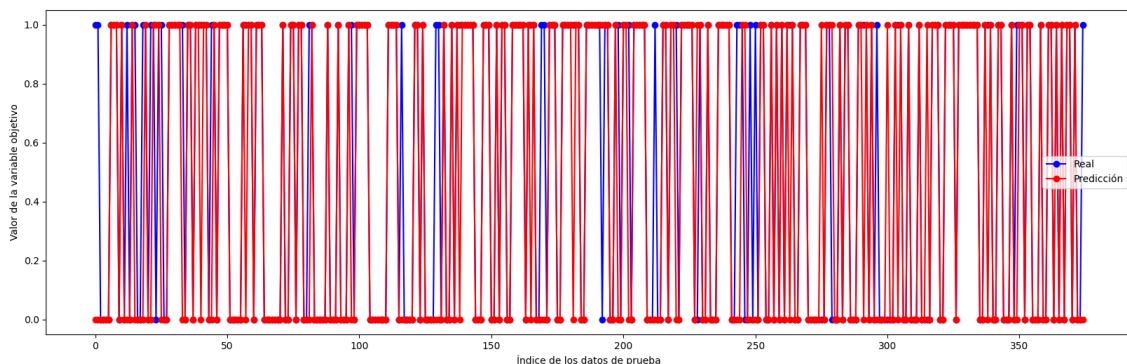


Imagen 9. Ejemplo de comparación entre valores reales y predichos con redes neuronales

En la **imágenes 5-9** se muestran valores del cierre de rías para la ría Cangas B desde el 2014 al 2019. En rojo los valores predichos y en azul, los reales:

- **Imagen 5:** regresión logística, 92,27% de acierto.
- **Imagen 6:** árbol de decisión, 98,13% de acierto.
- **Imagen 7:** regresión logística, 97,87% de acierto.
- **Imagen 8:** regresión logística, 88,27% de acierto.

- **Imagen 9:** regresión logística, 89.07% de acierto.

Como se puede observar, la gran mayoría de veces y utilizando diversos algoritmos, los valores predichos coinciden con los reales, lo que significa que **conociendo el valor de la toxina, índice de afloramiento y fecha, conocemos con una certeza casi absoluta el estado de la ría en ese momento.**

También se generó un **gráfico de importancia de características** para comprobar el peso de cada variable en la predicción.

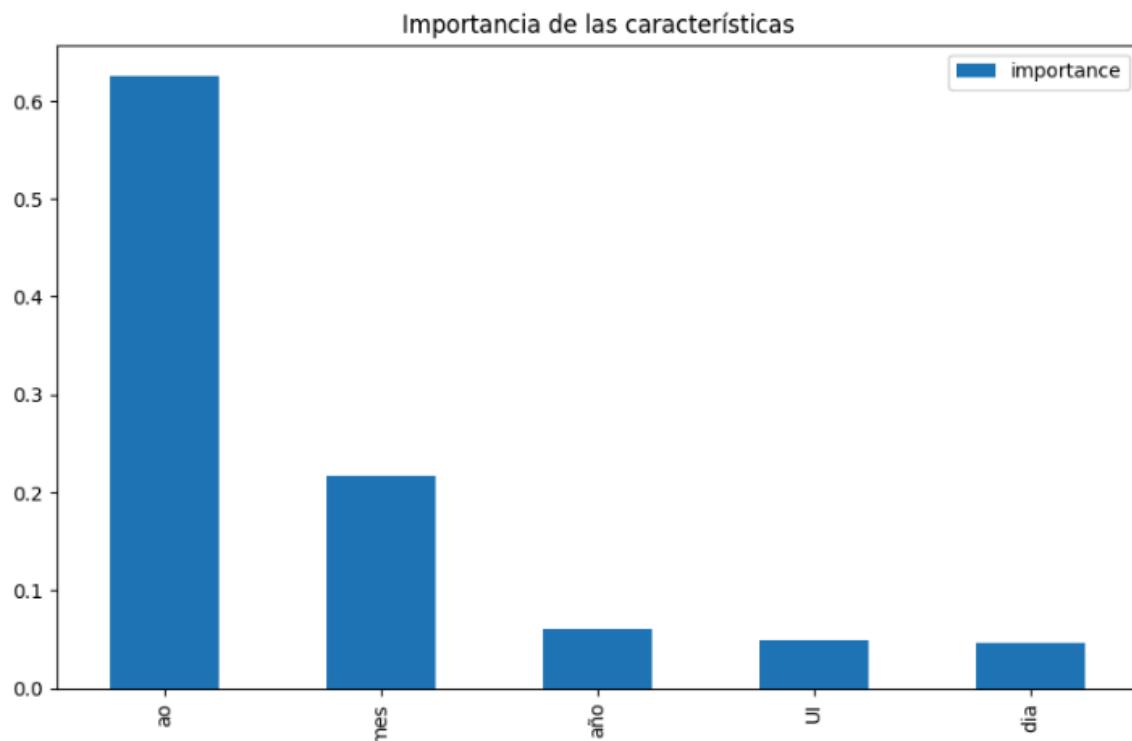


Imagen 10. Importancia de las características en la predicción del estado de rías

En la **imagen 10** se muestra la importancia de las características en la predicción del estado de la ría.

- **ao:** importancia del nivel de toxina en la predicción
- **mes:** importancia del mes en la predicción
- **año:** importancia del año en la predicción
- **UI:** importancia del índice de afloramiento en la predicción
- **dia:** importancia del día en la predicción

Como se puede ver en la **imagen 10**, la mayoría del peso de la predicción recae en la **toxina (ao)**, y en menor medida, el **mes**.

A pesar de que en teoría, el índice de afloramiento podría llegar a ser una de las variables determinantes de la predicción, en la práctica, **apenas tenía importancia (5% aproximadamente)**, como muestra el siguiente gráfico de la imagen 10.

Impedimentos de desarrollo

A pesar de que conociendo el nivel de toxina, podemos saber con gran acierto el estado de la ría, **predecir el cierre de rías mediante toxinas, índices de afloramiento y fecha no fue posible**, debido a:

- **Dificultad de obtención de datos de toxinas:** cada vez que se necesiten datos de toxinas, habría que enviar una solicitud por medio de la Xunta a INTECMAR. A pesar de ser un proceso que parece rápido y sencillo, la espera de respuesta la primera vez se demoró en más de 2 semanas.
- **Falta de otros aspectos relevantes en la aparición de toxinas:** en la actualidad, no existen los suficientes mecanismos externos que faciliten la obtención de otros datos relevantes en la aparición de toxinas, como predicciones futuras de variables ambientales o datos históricos de toxinas e índices de afloramientos sin actualizar.

4.1.3. Versión 3 - Predicción de probabilidad de cierre de rías por fecha

4.1.3.1. Versión 3 - Sprint 0: Investigación previa y especificación de requisitos

Dado que no se pudo emplear ni el índice de afloramiento ni el nivel de toxina para prever el estado futuro de las rías, el único dato constante que poseemos es **la fecha**. Por lo tanto, **el propósito de la herramienta permanece igual**, tan sólo cambian las variables que se usan para generar predicciones sobre la probabilidad de cierre de rías.

Debido a que el propósito no varía, **no fue necesario realizar ningún tipo de investigación adicional a la realizada en el sprint 0 de la versión 2**.

4.1.3.2. Versión 3 - Sprint 1: Obtención y procesamiento de datos

Todos los datos que se recopilan **de forma manual (históricos)** necesarios para el desarrollo de esta versión ya fueron obtenidos en el **sprint 1 de la versión 2**, por lo que **tampoco fue necesario realizar ningún tipo de proceso extra**, en este caso.

Por el consumo de tiempo que requería una recolección manual de los históricos de cierre de todas las rías de Galicia, **tan sólo se recopilaron datos de 4 de ellas (Cangas B, Sada 1, A Pobra A, Villagarcía A)**. La elección se debe a ser rías distantes entre ellas y tener patrones totalmente diferentes, lo que favorecería a la diversidad en las predicciones.

Para poder obtener los **datos actuales de cierre de rías**, se desarrollaron **2 scripts**, que **recopilan los datos actuales y cruzan estos datos actuales con los históricos**, tal y como se explica en el apartado **8. Diseño de software**.

4.1.1.2. Versión 3 - Sprint 2: Estudio de modelos de predicción

Debido a que ya no se contó con las variables de toxina e índice de afloramiento como en la versión 2, se decidió **descartar el uso de los algoritmos y modelos previamente seleccionados**.

Durante este *sprint*, se concluyó que el algoritmo más óptimo para nuestro tipo de predicciones, totalmente dependientes de la fecha, es **Prophet**, un modelo de descomposición aditiva para series temporales que permite hacer pronósticos a largo plazo.

El proceso se desarrolló de la siguiente manera:

1. **Evaluación de la Disponibilidad de Datos:** Al no contar con las variables de toxina e índice de afloramiento, se realizó una evaluación exhaustiva de los datos disponibles. Se determinó que las predicciones podían basarse exclusivamente en la fecha, lo que hizo necesario replantear la estrategia de modelado.
2. **Selección de Prophet:** Prophet fue seleccionado debido a su capacidad para manejar datos de series temporales con patrones estacionales y tendencias a largo plazo. Este modelo es particularmente eficaz en escenarios donde las predicciones son dependientes de la fecha.

Con estos pasos, se logró **obtener un modelo capaz de generar predicciones basadas únicamente en la fecha**, aprovechando la estacionalidad observada en los cierres de rías. Prophet permitió crear un patrón de probabilidades de cierre para cada ría en temporalidades grandes, asegurando una herramienta eficaz para la toma de decisiones, basada en datos históricos y tendencias estacionales.

4.1.3.3. Versión 3 - Sprint 3: Generación de predicciones

Una vez escogido el algoritmo a usar, Prophet, se procedió a **generar las predicciones futuras de cierres de rías (tabla 6), los gráficos representativos de estas predicciones (imágenes 11-14) y las métricas de predicción (tabla 7)** para mostrar en la interfaz web.

El proceso se desarrolló de la siguiente manera:

1. **Implementación de Prophet:** Se integró el algoritmo Prophet en el entorno de trabajo, asegurando que estuviera correctamente configurado para procesar los datos históricos de cierres de rías y generar predicciones precisas.
2. **Generación de Predicciones:** Utilizando Prophet, se generaron las predicciones futuras de cierres de rías. Estos datos fueron estructurados y almacenados en un archivo csv (tabla 6) para facilitar su uso posterior en la generación de gráficos.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

3. **Creación de Gráficos:** Se elaboraron gráficos representativos de las predicciones utilizando librerías de visualización de datos. Estos gráficos (imágenes 11-14) fueron diseñados para ilustrar claramente las variaciones en los cierres de rías predichos.
4. **Cálculo de Métricas:** Se calcularon diversas métricas para evaluar la precisión y rendimiento del modelo de predicción. Estas métricas fueron organizadas en archivo *csv* (tabla 7), proporcionando una visión clara de la efectividad del algoritmo Prophet.

ds	yhat
2024-07-17	0.2217384808260282
2024-07-18	0.21482694812094522
2024-07-19	0.1976027674792269
2024-07-20	0.18931696840940826

Tabla 6. Extracto de predicción de la ría A Pobra A

En la **tabla 6** se muestra un extracto de predicción de la ría de A Pobra A:

- **ds:** fecha de la predicción
- **yhat:** probabilidad de cierre de ría

zona	porcentaje	rango
A Pobra A	95.4	2014-01-16 - 2023-12-31
Cangas B	69.3	2014-01-16 - 2023-12-31
Sada 1	71.2	2014-01-16 - 2023-12-31
Vilagarcía A	97.3	2014-01-16 - 2023-12-31

Tabla 7. Ejemplo de métricas de predicción

En la **tabla 7** se muestra un ejemplo de métricas de las rías disponibles en esta versión de Mytilus:

- **zona:** ría a la que pertenecen las métricas
- **porcentaje:** porcentaje de acierto histórico
- **rango:** rango de fechas de los datos históricos

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024



Imagen 11. Ejemplo de comparación entre valores reales y predichos con regresión logística

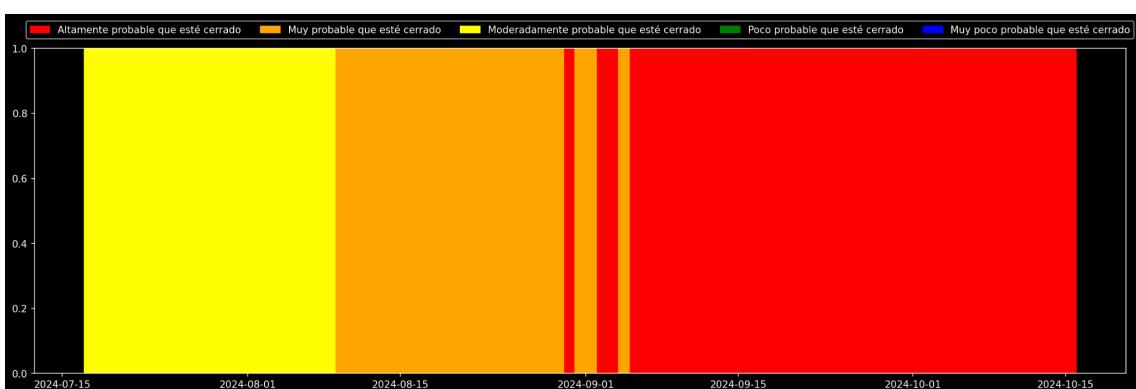


Imagen 12. Ejemplo de comparación entre valores reales y predichos con regresión logística

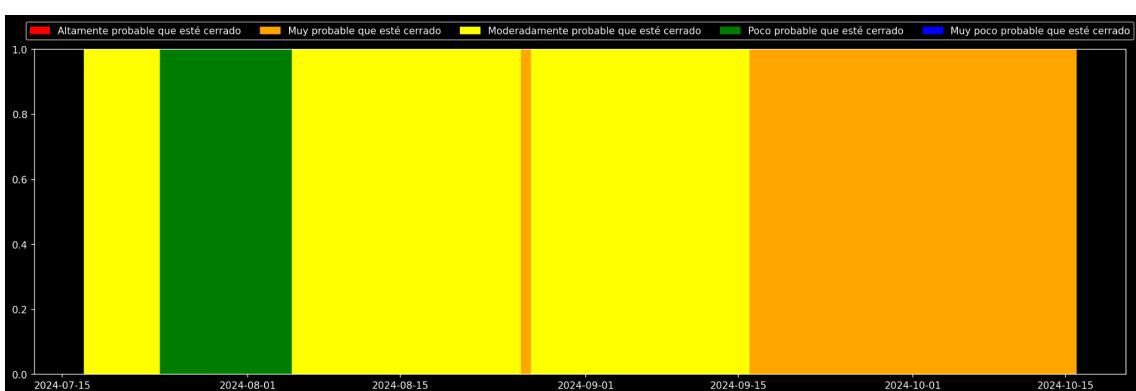


Imagen 13. Ejemplo de comparación entre valores reales y predichos con regresión logística

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024



Imagen 14. Ejemplo de comparación entre valores reales y predichos con regresión logística

En la **imágenes 11-14** se muestran los gráficos representativos del cierre de rías del día 15 de julio de 2024 al 15 de octubre de 2024:

- **Imagen 11:** ría de A Pobra A
- **Imagen 12:** ría de Cangas B
- **Imagen 13:** ría de Sada 1
- **Imagen 14:** ría de Villagarcía A

Mediante el siguiente código de colores:

- **Azul:** 0-20% de probabilidad de cierre
- **Verde:** 20-40% de probabilidad de cierre
- **Amarillo:** 40-60% de probabilidad de cierre
- **Naranja:** 60-80% de probabilidad de cierre
- **Rojo:** 80-100% de probabilidad de cierre

Después de obtener las predicciones, los gráficos asociados a las predicciones de cada ría y las métricas para cada ría, se procedió a **crear la interfaz web**.

4.1.3.4. Versión 3 - Sprint 4: Creación de la interfaz web

Para diseñar la interfaz web, se utilizaron las **tecnologías HTML, CSS y JS**. Con el fin de que fuera una interfaz sencilla y accesible para cualquier tipo de usuario, la única funcionalidad que el usuario puede realizar es **iterar entre las diferentes rías** implementadas en la herramienta, tal y como demuestra la **imagen 15**.

El desarrollo de la interfaz siguió estos pasos:

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

1. **Estructura HTML:** Se definió una estructura *HTML* clara y semántica, asegurando que cada elemento tuviera su propósito bien definido.
2. **Estilos CSS:** Se aplicaron estilos *CSS* para mejorar la apariencia visual de la interfaz.
3. **Interactividad con JS:** Se integró *JavaScript* para añadir interactividad a la interfaz. Esto incluyó la funcionalidad de navegación entre las diferentes rías mediante formulario y la actualización dinámica del contenido mostrado al usuario, sin necesidad de recargar la página.

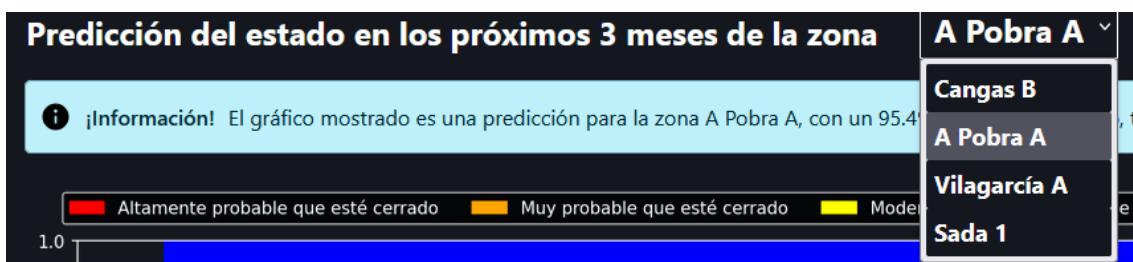


Imagen 15. Formulario de iteración de rías

Así, los usuarios pueden la información de la ría de interés, permitiendo mostrar el gráfico de predicción de cierre de rías (**Imagen 16**) y las métricas relevantes para esa ría (**Imagen 17**)



Imagen 16. Gráfico de probabilidad de cierre mostrado en interfaz web



Imagen 17. Información relativa a métricas mostrada en interfaz web

Una vez implementada la interfaz web con los datos generados en los anteriores *sprints*, se procedió a **alojar la herramienta en un servidor web**.

4.1.3.5. Versión 3 - Sprint 5: Implementación en servidor

Para facilitar el uso de la herramienta, se implementó la herramienta, así como el ciclo de ejecución de scripts en un **servidor**.

En concreto, se escogió una instancia **EC2 de AWS [14]**, debido a que ofrece una capa gratis muy interesante en cuanto a capacidades. Para ello, se creó la instancia con las siguientes especificaciones: tipo de instancia **t2.micro**, que proporciona un equilibrio adecuado entre rendimiento y costos para necesidades básicas. Se configuró el sistema operativo a **Ubuntu 20.04 LTS**.

Una vez creada la instancia, se realizaron los siguientes pasos para la configuración:

1. **Configuración de la Red y Seguridad:** Se ajustaron las reglas del grupo de seguridad para permitir el acceso HTTP desde cualquier dirección IP.
2. **Instalación de Dependencias:** Se instalaron las dependencias necesarias para la herramienta mediante comandos de apt-get y se aseguraron las últimas actualizaciones del sistema operativo.
3. **Configuración del Entorno de Ejecución:** Se configuró el entorno de ejecución de scripts, incluyendo la interfaz web siempre esté activa y accesible, mediante la ejecución ininterrumpida del archivo app.py.
4. **Automatización de Scripts:** Se programaron tareas de automatización mediante cron jobs para garantizar que los scripts se ejecuten en los intervalos deseados sin necesidad de intervención humana manual.

Con esta configuración, la herramienta está disponible de manera remota, permitiendo una fácil gestión y ejecución de scripts en el servidor AWS EC2, y por lo tanto, **se completa el desarrollo de Mytilus**.

4.2. Distribución de tiempo por sprints

Aunque **siempre se ha utilizado la metodología Scrum en el desarrollo de Mytilus**, hay algunas razones clave por las que la distribución de tiempo por *sprint* solo se detalla en la documentación de la **versión 3**, a pesar del consumo de tiempo requerido por la **versión 1 y 2**:

- **Enfoque en la Versión Final:** la versión 3 es la versión final de nuestra herramienta, por lo que consideramos crucial proporcionar una documentación completa y detallada que refleje con precisión nuestro proceso de desarrollo.
- **Madurez y Estabilidad del Proyecto:** las versiones anteriores, aunque desarrolladas con Scrum, estaban en etapas más tempranas y menos estables del proyecto. Documentar la distribución de tiempo por *sprint* en estas versiones no aportaría tanto valor como hacerlo en la versión final.

- **Documentación Enfocada y Concisa:** para mantener la documentación clara y concisa, decidimos centrar la información detallada en la versión más representativa y final del proyecto.

A raíz de esto, nace la necesidad de **contabilizar por separado las fases de desarrollo de las versiones 1-2 y la versión final, la 3**. La mayor parte de esta fase previa al desarrollo de versión final se llevó a cabo durante los primeros meses del proyecto, desde octubre-noviembre de 2023, dónde se presentó la idea a las tutoras, hasta abril de 2024.

Este periodo se caracteriza por los siguientes aspectos clave:

- **Primer punto crítico del proyecto:** al no contar con los datos ni el conocimiento necesario, resultaba difícil determinar el alcance del proyecto y validar la viabilidad de las propuestas.
- **Inversión de tiempo considerable:** la búsquedas de fuentes fiables de datos, el contacto con profesionales del sector y la investigación sobre estudios relacionados con el mejillón demandó una cantidad significativa de tiempo y esfuerzo, especialmente debido a la pobre digitalización de los datos y la información relativa al sector mejillonero.
- **Restricciones de tiempo:** durante estos primeros meses, el tiempo disponible para avanzar en el proyecto fue limitado, por lo que se ralentizó el progreso en general.

Por todo ello, **la fase de desarrollo de las versiones 1 y 2 tiene una repercusión de horas elevadas en el proyecto**.

4.2.1. Sprint 0: Investigación previa y especificación de requisitos

Tal y como se indica en el apartado **4.1.3.1. Versión 3 - Sprint 0: Investigación previa y especificación de requisitos, no fue necesario ningún tipo de investigación previa**, dado que ya se había realizado en los *sprint 0* de las versiones 1 y 2. Debido a esto, el **sprint 0 de la versión final no tiene repercusión en las horas de desarrollo del proyecto**.

Sin embargo, en el *sprint 0* de la versión 3 se realizaron las siguientes tareas, que no fueron posibles de realizar hasta empezar el *sprint 0* de la versión final:

- Se especificaron 5 *sprints*, empezando la primera semana de abril de 2024, tal y como se muestra en la siguiente tabla.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Sprint	Semana									
	1	2	3	4	5	6	7	8	9	10
1	01-04 a 14-04									
2			15-04 a 28-04							
3					29-04 a 12/05					
4							13-05 a 26-05			
5									27-05 A 09-06	

Tabla 8. Planificación inicial en sprints

- Se especifica un **Product Backlog** a través de los requisitos del sistema. Al pasar al **Sprint Backlog** correspondiente, se le asigna una duración en horas estimada. Se opta por definir el **Product Backlog** en requisitos del sistema en vez de en historias de usuario debido a que muchas de las acciones de la herramienta no las realiza el usuario final.

Ítem	Descripción
P1	Recopilar datos históricos de cierre de rías manualmente
P2	Establecer conexión con la web de INTECMAR
P3	Descarga los datos de días de cierre por ría de INTECMAR
P4	Construir archivo de estado de rías actual
P5	Cruzar datos históricos y actuales de estado de rías
P6	Generar predicciones futuras de estado de rías
P7	Generar gráficos de predicciones futuras de estado de rías
P8	Generar métricas relevantes
P9	Estudio de algoritmos y modelos de predicción
P10	Comparación de algoritmos y modelos de predicción
P11	Diseñar interfaz web de la herramienta
P12	Mostrar gráficos y métricas generados
P13	Diseñar la lógica del formulario de iteración
P14	Estudio de alternativas de proveedores de servidores
P15	Configuración del servidor
P16	Configuración del proyecto en el servidor

P17	Configuración de automatizaciones del proyecto
-----	--

Tabla 9. Product Backlog

- Al inicio de cada *sprint* se realiza un **Sprint Planning**, en el que se construye el *Sprint Backlog* a partir del objetivo especificado para el *sprint*.
- Para simular los **Daily Scrums**, se reservan 5 minutos antes de empezar a trabajar en el proyecto cada día, para realizar los ajustes necesarios en la planificación.
- En el último día de cada *sprint*, se realizan **Sprint Retrospective y Sprint Review**, donde se revisa el trabajo realizado a lo largo del *sprint* y se generan los resultados finales del *sprint*.

Tras esta primera etapa de planificación e investigación, **se procede a desglosar el tiempo de desarrollo de la herramienta de los siguientes sprints**.

4.2.1. Sprint 1: Obtención y procesamiento de datos

El objetivo de este *sprint* fue **obtener todos los datos necesarios para la generación de predicciones de estados de rías**. Este *sprint* se llevó a cabo desde el día 1 de abril hasta el día 14 de abril.

Durante la reunión de planificación del *sprint*, se seleccionaron los ítem del *Product Backlog* que se completarían en el **Sprint 1**, y se le asignó un trabajo estimado en horas. También se midió el tiempo dedicado a cada tarea para mostrar las desviaciones temporales del **Sprint 1**.

Ítem	Horas estimadas	Horas reales
P1	6	5
P2	2	1
P3	2	1
P4	15	20
P5	2	2
Total	27	29

Tabla 10. Sprint Backlog del Sprint 1

Como se puede observar en la **tabla 10**, no existe una gran desviación en las horas estimadas y reales. El número de horas necesarias para completar las actividades de este **Sprint 1** fue bastante ajustado a la realidad, debido a la experiencia previa con las tecnologías y métodos utilizados en *sprint*.

Al finalizar el *Sprint 1*, obtuvimos todos los datos históricos del estado de las rías necesarios para las realizar las predicciones y un método fiable y efectivo de obtener los datos actuales del estado de las rías.

4.2.2. Sprint 2: Estudio de modelos de predicción

El objetivo del *Sprint 2* fue realizar un estudio extenso de los métodos y algoritmos de aprendizaje automático. Este *sprint* se llevó a cabo desde el día 15 de abril hasta el día 6 de mayo.

En la reunión de planificación del *Sprint 2*, se seleccionaron los ítems del *Product Backlog* para ser completados durante ese *sprint* y se estimaron las horas necesarias para cada uno. También se registró el tiempo real invertido en cada tarea para analizar las diferencias con las estimaciones iniciales.

Ítem	Horas estimadas	Horas reales
P9	15	20
P10	20	30
Total	35	50

Tabla 11. Sprint Backlog del Sprint 2

La desviación entre las horas estimadas y reales mostrada en la **tabla 11** se debe mayormente a **problemas técnicos y complejidad no anticipada**. Durante la comparación de algoritmos y modelos de aprendizaje surgieron problemas en las pruebas iniciales, que derivaron en comportamientos y resultados no esperados. También se vió influenciado por la falta de conocimiento previo en algoritmos y modelos de aprendizaje para series temporales, provocando una subestimación de la carga de trabajo a realizar en este *Sprint 2*.

Tras sobrepasar las horas estimadas para la realización de este *sprint*, causadas por un bajo rendimiento, se decidió extender el *Sprint 2* y completar los ítems asignados previamente, debido a que el objetivo inicial del *sprint* y las definiciones de los ítems seguían alineadas.

Una vez terminado el *Sprint 2*, se definió el algoritmo concreto y la forma de uso ideal para generar predicciones futuras de estados de rías.

4.2.3. Sprint 3: Generación de predicciones

El *Sprint 3* tenía como objetivo realizar las predicciones futuras, para generar gráficos y métricas relevantes que se mostrarían en la interfaz web. Este *sprint* se llevó a cabo desde el día 7 de mayo hasta el día 21 de mayo.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Durante la reunión de planificación del **Sprint 3**, se incorporaron las tareas pendientes del *Product Backlog* al *Sprint Backlog* correspondiente, asignándoles una estimación de horas requeridas.

Ítem	Horas estimadas	Horas reales
P6	8	10
P7	12	15
P8	8	6
Total	28	31

Tabla 12. Sprint Backlog del Sprint 3

En esta ocasión, como se puede observar en la **tabla 12**, tampoco se generó una gran desviación del trabajo estimado al real, gracias a que en el *Sprint 2*, a pesar de haber consumido más horas de las previstas, el resultado fue muy beneficioso para el desarrollo de este *Sprint 3*, provocando una desviación en la planificación del trabajo a realizar mínima.

Al terminar el desarrollo de los ítems del *Sprint 3*, obtuvimos un método de generación de predicciones, así como de gráficos y métricas representativas de predicciones generadas.

4.2.4. Sprint 4: Creación de la interfaz web

El *Sprint 4* tenía como objetivo crear la interfaz que mostraría los datos generados en el *sprint anterior*. Este *sprint* se llevó a cabo desde el día 22 de mayo hasta el día 27 de mayo.

Durante la reunión de planificación del *sprint*, se seleccionaron los ítems del *Product Backlog* que se completarían en el *Sprint 4*, y se le asignó un trabajo estimado en horas. También se midió el tiempo dedicado a cada tarea para mostrar las desviaciones temporales del *Sprint 4*.

Ítem	Horas estimadas	Horas reales
P11	15	10
P12	7	4
P13	5	8
Total	27	22

Tabla 13. Sprint Backlog del Sprint 4

Tal y como se observa en la **tabla 13**, el rendimiento en este *sprint* fue mejor de lo esperado. Esto fue debido a la experiencia previa adquirida en diseño web y al buen

resultado generado en el anterior *Sprint 3*, evitando tener que modificar el formato de los gráficos y las métricas a mostrar.

El resultado de este *sprint* fue la creación de una interfaz web totalmente funcional en un equipo local, implementando las funcionalidades especificadas. Al término de este *sprint*, la herramienta estaba totalmente completa para el funcionamiento en un equipo local. Sin embargo, para facilitar el uso por parte del usuario final, se procede a implementar Mytilus en un servidor.

4.2.5. Sprint 5: Implementación en servidor

El objetivo del *Sprint 5* fue implementar y alojar la herramienta en un servidor que permitiera el acceso a la interfaz y el funcionamiento de la herramienta sin necesidad de ejecución por parte del usuario final. Este *sprint* se llevó a cabo desde el día 27 de mayo hasta el día 17 de junio.

En la reunión de planificación del *Sprint 5*, se seleccionaron los ítems restantes del *Product Backlog* y se estimaron las horas necesarias para cada uno. También se registró el tiempo real invertido en cada tarea para analizar las diferencias con las estimaciones iniciales.

Ítem	Horas estimadas	Horas reales
P14	5	3
P15	10	12
P16	10	14
P17	8	12
Total	33	41

Tabla 14. Sprint Backlog del Sprint 5

Como se puede observar en la **tabla 14**, volvemos a tener una desviación considerable en las horas estimadas y reales. En este caso, también se debe a la subestimación del trabajo a realizar y a una falta de conocimiento previo en configuración de servidores y scripts, aumentando las horas requeridas para el desarrollo de los ítems de configuración P14, P15, P16 y P17.

Una vez terminado el *Sprint 5*, la herramienta está totalmente implementada en un servidor que permite el acceso desde cualquier equipo, liberando al usuario final de la ejecución de los diferentes scripts que componen la herramienta.

4.7. Revisión y evaluación de la planificación

Al revisar la planificación desde una perspectiva general, se puede notar que el principal motivo de la extensión del tiempo del proyecto fueron los *Sprints 2 y 5*, debido a que en

ambos casos a la **inexperiencia en los temas y tecnologías utilizadas** en el desarrollo de cada uno de ellos.

Por otro lado, los **Sprint 2 y 3** se completaron al **ritmo previsto**, lo cual ayudó a reducir la discrepancia entre las estimaciones de estos *sprints*.

El **Sprint 4** se desarrolló a un **ritmo más rápido del esperado**, contribuyendo así a una disminución del tiempo de desarrollo global del proyecto.

Por último, observamos en la **Imagen 4** que en la distribución de tiempo entre los diferentes *sprints*, el tiempo dedicado a las **versiones 1 y 2** consumió casi la mitad del tiempo total dedicado al proyecto.

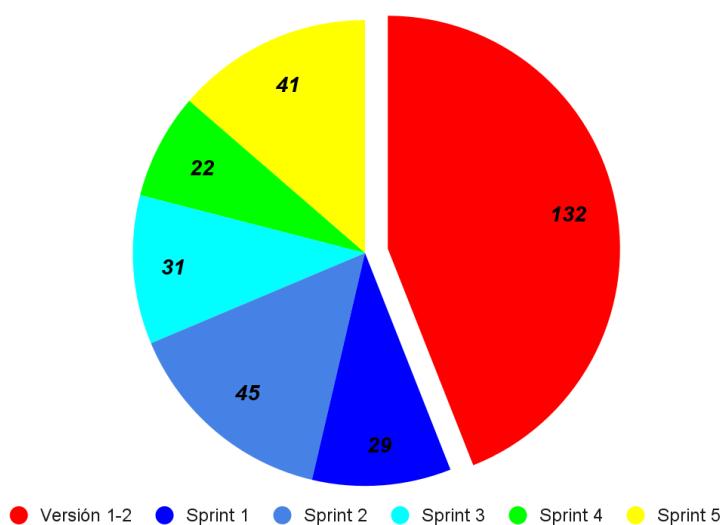


Imagen 18. Reparto del tiempo dedicado a cada sprint

5. Arquitectura

Para diseñar la arquitectura del sistema, se han tomado en cuenta los **objetivos específicos de la herramienta**. Estos objetivos demandan **asegurar la modularidad** para facilitar expansiones y mejoras futuras, así como **garantizar la robustez y seguridad del sistema**.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

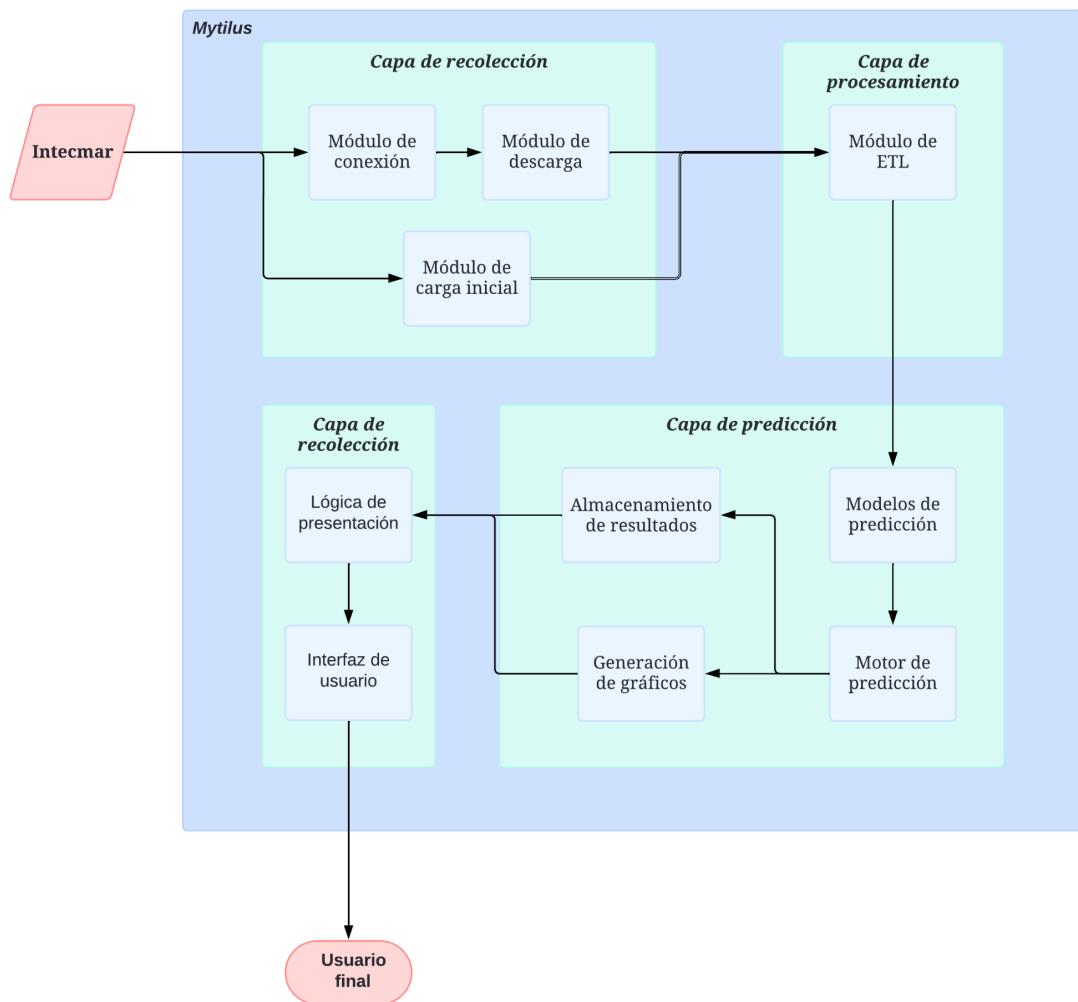


Imagen 19. Diagrama de la arquitectura del sistema

Debido a esto, se opta por la **arquitectura basada en capas** [15] mostrada en la **imagen 5**. Esta elección arquitectónica permite **separar claramente las preocupaciones y responsabilidades** dentro del sistema, facilitando la gestión de cada componente de manera independiente y promoviendo la reutilización de código y la flexibilidad en el desarrollo.

La arquitectura basada en capas se organiza en niveles o **capas lógicas**, cada una con funciones específicas y bien definidas. A continuación se detallan los componentes principales de cada capa:

5.1. Capa de recolección

Se encarga de **recopilar datos de Intecmar**. Tal como se puede observar en la **imagen 5**, esta capa está formada por dos componentes:

- **Módulo de carga inicial**: se encarga de crear una primera carga inicial de los datos recolectados manualmente, que serán datos históricos de cierre de rías.

- **Módulo de conexión:** se encarga de establecer la conexión con Intecmar. En este caso, ya que Intecmar no dispone de API o servicio web por el que obtener los datos necesarios para la predicción, se desarrolla un script en python. Este script se conecta al sitio web donde Intecmar actualiza los datos de cierre de rías a diario.
- **Módulo de descarga:** responsable de la descarga de los datos desde la fuente de Intecmar. Una vez establecida la conexión, el script descarga los datos de la tabla de la web mencionada anteriormente.

5.2. Capa de procesamiento

- **Módulo de ETL:** se encarga de gestionar el proceso completo de ETL (*extract, transform and load*). Primero, los datos brutos descargados son extraídos para su transformación. Luego, los datos extraídos son transformados, de forma que se obtengan los estados de las rías por día. También, cada cierto tiempo, se incluyen a los datos transformados del estado de rías más actual, los datos del módulo de carga inicial, para mantener el conjunto de datos histórico actualizado. Por último, estos datos son almacenados en formato *csv*, quedando así listos para su uso en predicciones en la siguiente capa.

5.3. Capa de predicción

Se encarga de **generar predicciones** al alimentar el motor de predicción con los datos obtenidos de las capas anteriores. Además, **crea gráficos** para visualizar estas predicciones y **almacena otros resultados relevantes** que se mostrarán en la interfaz web.

Como se observa en la **Imagen 5**, esta capa está formada por cuatro componentes:

- **Modelos de predicción:** son el algoritmo/modelo de predicción entrenado para generar predicciones basadas en los datos procesados. En este módulo se realizan entrenamientos del modelo, así como las evaluaciones y validaciones del mismo.
- **Motor de predicción:** infraestructura que ejecuta el modelo de predicción y gestiona las solicitudes de predicción. Se encarga de cargar el modelo entrenado y realizar las predicciones.
- **Generación de gráficos:** módulo encargado de transformar los datos predichos en gráficos y exportarlos en formatos adecuados para ser presentados en la capa de presentación.
- **Almacenamiento de resultados:** sistema de almacenamiento donde se guardan datos relacionados con las predicciones para su posterior uso y análisis.

5.4. Capa de presentación

Se encarga de **mostrar las predicciones generadas** en la capa anterior. Como se muestra en la **Imagen 5**, esta capa está formada por dos componentes:

- **Interfaz de usuario:** es la parte visible con la que interactúa el usuario final. Presenta los datos de predicción y gráficos generados de una manera clara y accesible. También permite navegar por diferentes rías, permitiendo así visualizar gráficos específicos para cada una de ellas.
- **Lógica de presentación:** es el puente entre la interfaz de usuario y los datos de predicciones procesados. Maneja las solicitudes del usuario, permitiendo que se muestre la información que el usuario desea.

La **modularidad** permite una actualización y mantenimiento sencillos, ya que cada módulo trabaja de manera independiente del anterior y posterior. Por ejemplo, si Intecmar implementa una API para poder solicitar datos de una forma más sencilla, tan solo habría que modificar el módulo de recolección, ya que el resto sólo depende de que existan unos datos en un formato concreto. Otro ejemplo sería cambiar el algoritmo de predicción, ya que lo único que cambiaría es solamente eso, ya que la recolección de datos, su procesamiento y presentación serían exactamente igual.

En resumen, la arquitectura de esta herramienta está diseñada para ser **robusta, escalable y segura**, facilitando la recolección, procesamiento, predicción y presentación de los datos.

6. Tecnologías e integración de productos de terceros

6.1 Tecnologías y librerías

- **Python (y librerías) [16]:** Mytilus se basa en la generación de predicciones de mejillón. Para desarrollar el código relacionado con predicciones y desarrollar parte de la app web, se utilizó *Python*. Actualmente, *Python* es uno de los lenguajes de programación más usados y extendidos del mundo. Esta decisión asegura la sostenibilidad y mantenimiento del software a largo plazo.

Python debe su popularidad a su sencillez de sintaxis y aprendizaje, versatilidad, comunidad activa y amplio ecosistema de librerías.

En el contexto específico del proyecto, la predicción de cosechas, *Python* ofrece una extensa **ecosistema de librerías** para desarrollar modelos predictivos. En el desarrollo de este proyecto se utilizan:

- **flask [17]:** es un framework ligero diseñado para construir apps web en Python. Facilita la creación de servidores web, manejo de rutas, procesamiento de solicitudes HTTP y respuestas, entre otros.
- **csv [18]:** se encarga de la manipulación de archivos de valores separados por comas (csv). Proporciona herramientas para leer y escribir archivos csv de forma sencilla.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

- **os [19]:** permite interactuar con el sistema operativo. Acceder a variables de entorno y manejar archivos y directorios son algunas de las operaciones que permite.
- **pandas [20]:** se usa para manipular y analizar datos. Su uso se debe a que proporciona estructuras de datos flexibles como *DataFrame*¹ y *Series*², así como herramientas para la manipulación, limpieza y análisis.
- **datetime [21]:** se encarga de manipular variables en formato fecha y hora.
- **prophet [22]:** es una herramienta desarrollada por Facebook que facilita el pronóstico de series temporales con estacionalidad y tendencias.
- **sklearn [23]:** ofrece algoritmos de predicción y herramientas para tareas de clasificación, preprocesamiento de datos, entre otros.
- **matplotlib [24]:** permite crear gráficos y visualizaciones.
- **numpy [25]:** ofrece soporte para matrices y *arrays* multidimensionales, junto a funciones matemáticas para operar datos.
- **request [26]:** se utiliza para realizar solicitudes *HTTP*.
- **bs4 [27]:** facilita la extracción de datos en archivos *HTML* mediante técnicas de *scrapping*.
- **Git [28]:** *Git* es un sistema de control de versiones distribuido. Esto significa que cada copia del proyecto es un repositorio de control de versiones completo.

Git se ha convertido en el estándar mundial para el control de versiones. Esto se debe a su flexibilidad y facilidad de uso.

El uso de *git* en el desarrollo de Mytilus se debe a diferentes factores:

- **Urgía controlar el desarrollo de código por día.** Debido a la necesidad de tener un seguimiento detallado de las horas dedicadas a cada ítem del Product Backlog, surge la necesidad de llevar un estricto control sobre el trabajo realizado, más concretamente en el desarrollo de código.
- **Integración con productos de terceros.** Otros *software* que se usan en el desarrollo de Mytilus tienen integración por defecto con *git*, como puede ser Visual Studio Code, por lo que el uso de *git* es más favorable que otros sistemas de control de versiones.

¹ Dataframe es una serie de Series Pandas indexadas por un valor.

² Series es un vector con datos indexados.

- **Mantener un historial detallado de cambios.** Como el propósito de la herramienta ha ido variando según avanzaba su desarrollo, también ha sido necesario ver cambios y versiones pasadas, para así comprobar que se podía utilizar y que había que descartar.
- **Python Notebook [29]:** Python Notebook es una herramienta popular en el desarrollo de proyectos relacionados con ciencia de datos y análisis, utilizada para desarrollar código interactivo, visualizaciones de datos y documentación.

Gracias a su formato de celdas de código y texto, permite crear documentos ejecutables por porciones, lo que posibilita y facilita la visualización de variables y gráficos en diferentes estados de la ejecución.

Dentro del desarrollo de Mytilus, *Python Notebook* se ha usado principalmente en el desarrollo de primeros bocetos y ejecución de pruebas, proporcionando un espacio de pruebas y desarrollo eficaz.

- **HTML [30]:** HTML es el lenguaje estándar para crear páginas web. Define la estructura básica de una web utilizando elementos y etiquetas.

En Mytilus, se utilizar para definir la estructura de la página web donde se mostrarán las predicciones, incluyendo formularios para navegar entre rías, imágenes donde mostrar los gráficos y establecer la base para aplicar estilos mediante *CSS*.

- **CSS [31]:** CSS es un lenguaje de estilo utilizado para describir la presentación de documentos *HTML*. Definir el diseño, formato y apariencia visual de la web son algunas de sus funciones.

En Mytilus, se utiliza para dar formato a los elementos *HTML*, definir colores, fuentes, tamaños de los elementos y adaptar la web a diferentes tamaños de pantalla.

- **JavaScript [32]:** Es un lenguaje de programación utilizado para añadir interactividad a las páginas web.

Permite realizar acciones en respuesta a eventos del usuario y modificar el contenido de la web dinámicamente.

En Mytilus, se encarga de validar formularios, realizar peticiones al servidor y manipular la página para mostrar elementos dinámicamente sin necesidad de recarga.

6.1 Productos de terceros

- **Notion [33]:** Es una herramienta todo en uno que combina notas, bases de datos, calendarios y tareas en una sola interfaz. Además de ser muy flexible, permite la colaboración en tiempo real, lo que la hace ideal para equipos que necesitan

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

gestionar proyectos de manera integrada y mantener una documentación organizada.

Se ha optado por Notion en el desarrollo de Mytilus, además de por la experiencia previa y conocimiento de sus posibilidades, por la sencillez que ofrece para crear documentación, así como llevar un seguimiento de la pila de trabajo del *Sprint Backlog*.

- **Google Docs [34]:** Es una herramienta de procesamiento de texto en la nube que permite la creación y edición de documentos en tiempo real.

Se ha utilizado principalmente para desarrollar la documentación del TFG, principalmente porque permite el acceso desde cualquier lugar y dispositivo con acceso a internet

- **Github [35]:** Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones de *git*.

En este caso, se ha optado por Github y no otros proveedores de alojamiento para proyectos *git* por la integración con VS Code, ser un estándar en el alojamiento de proyectos y por la posibilidad de colaboración mediante revisión de código y solicitudes de extracción.

- **LucidChart [36]:** Es una herramienta en línea para la creación de diagramas y visualizaciones.

En Mytilus, se utiliza para diseñar los gráficos y visualizaciones presentes en esta documentación. Se ha optado por LucidChart por la facilidad de uso y por ofrecer un plan gratuito suficiente para diseñar gráficos y visualizaciones explicativas.

- **Firefox [37]:** Es un navegador popular de código abierto conocido por su enfoque de privacidad y personalización.

Es el navegador usado para acceder a todas las herramientas en línea mencionadas previamente, búsqueda de información, descarga de datos, entre otros.

- **Visual Studio Code [38]:** Visual Studio Code es un editor de código desarrollado por Microsoft. Se ha convertido en la herramienta más popular entre desarrolladores de software.

En este caso, se ha optado por el uso de VS Code en vez de otros editores por ser gratuito y de código abierto, multiplataforma, tener una gama amplia de extensiones que ayudan al desarrollo, integración perfecta con *git* y soporte para lenguajes como *Python*, *HTML*, *JS* y *CSS*, utilizados en el código del proyecto.

7. Especificación y análisis de requisitos

7.1. Requisitos funcionales

Los requisitos funcionales [39] representan funcionalidades o características que describen las **interacciones entre el sistema y su entorno entorno**, que pueden ser usuarios u otros sistemas, sin tener en cuenta su implementación. En términos simples, estos requisitos especifican lo que debe hacer el sistema.

Los requisitos funcionales capturados para el desarrollo de Mytilus son los siguientes:

- ***Scraping de datos:*** la herramienta debe ser capaz de extraer datos relevantes de la web de INTECMAR. La entrada será la URL de la web de INTECMAR que contiene la tabla de días de cierre por ría, y la salida será un conjunto de datos de estado de rías por días en formato *csv*.
- ***Actualización del histórico de datos:*** la herramienta debe incorporar cada cierto tiempo los nuevos datos scrapeados al conjunto de datos histórico. La entrada será los datos nuevos extraídos del scraping de datos, y la salida será un conjunto de datos históricos actualizados.
- ***Predicción del estado de las rías:*** la herramienta debe utilizar los datos históricos para predecir si las rías estarán abiertas o cerradas. La entrada será el conjunto de datos históricos actualizado, y la salida, la predicción del estado de las rías.
- ***Generación de gráficos:*** la herramienta debe generar gráficos a partir de los datos de predicciones. La entrada será los datos de predicción de cada ría, y la salida, gráficos claros y explicativos de las predicciones del estado para cada ría.
- ***Interfaz web:*** la herramienta debe proveer una interfaz web para que los usuarios puedan visualizar los gráficos de las predicciones. La entrada serán las diferentes solicitudes de visualización de rías por parte de los usuarios, y la salida, la visualización de los gráficos asociados a cada ría.

7.2. Requisitos no funcionales

Los requisitos no funcionales [39] complementan a los requisitos funcionales al especificar cómo el sistema debe realizar ciertas funciones. Imponen restricciones en implementación y diseño. En esencia, los requisitos no funcionales establecen los estándares para el rendimiento, la seguridad y la usabilidad del sistema.

Los requisitos no funcionales recogidos para el desarrollo de Mytilus son los siguientes:

- ***Simplicidad de interfaz gráfica:*** la interfaz gráfica debe ser simple a la vez que explicativa, con la finalidad de ser accesible para cualquier usuario, independientemente de su experiencia con estas tecnologías.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

- **Interfaz web:** para que no se precise instalación de ningún tipo, la interfaz debe ser accesible a través de una página web.
- **Rendimiento:** la herramienta debe ser capaz de procesar los datos y realizar las predicciones en un tiempo razonable.
- **Mantenibilidad:** el código de la herramienta debe ser fácil de mantener y actualizar, facilitando así futuras expansiones.
- **Confiabilidad:** la herramienta debe ofrecer predicciones precisas y consistentes, ofreciendo así un mejor producto final a los usuarios, mejorando su toma de decisiones.

8. Diseño de software

En el diseño de software, es fundamental comprender y planificar tanto la estructura como el comportamiento de un sistema.

Mytilus está compuesto por los siguientes componentes, relacionados según se muestra en la **Imagen 29:**

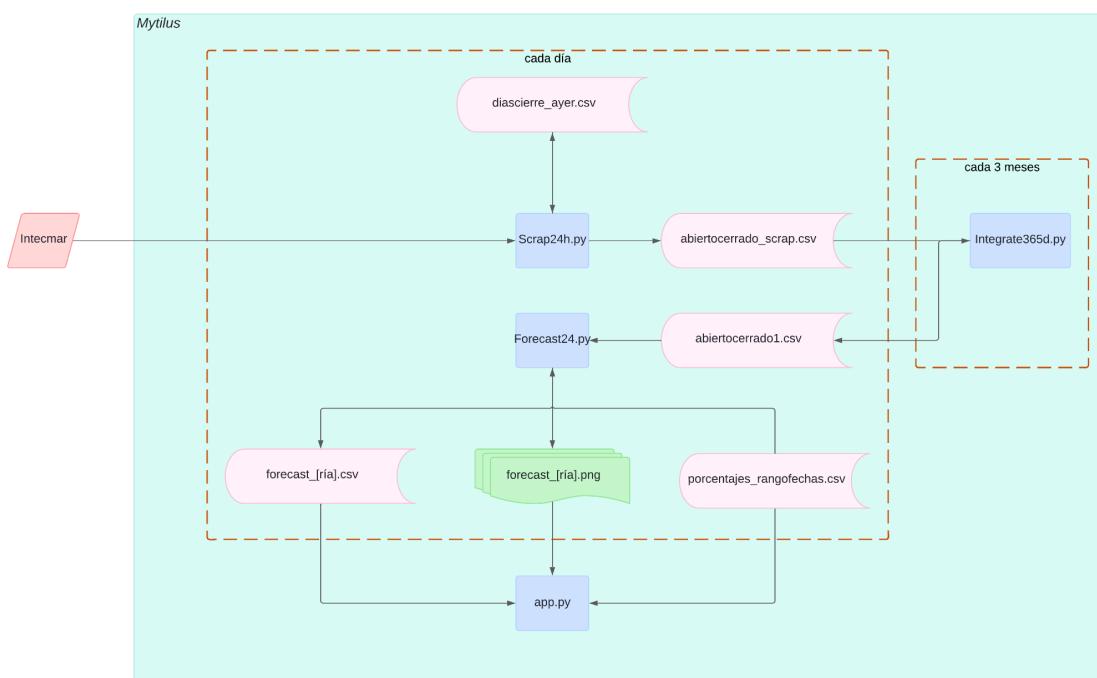


Imagen 20. Interacción entre los componentes de Mytilus

Ciclo diario:

- **daily.sh:** se encarga de ejecutar los scripts de *Python* que se ejecutan diariamente. Además, se encarga de actualizar el repositorio de *git*.

A continuación, se detalla el proceso de ejecución diario de la herramienta:

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

1. *Scrap24h.py* establece conexión con INTECMAR.
2. *Scrap24h.py* accede a la tabla que contiene los días de cierre por ría, y la compara con el archivo *diascierre_ayer.csv*, que contiene los días de cierre por mes del día anterior para cada ría, tal y como se presenta en **Imagen 21**.

Polígono	1	2	3	4	5	6	7	8	9	10	11	12	Total
Vilagarcía B1	0	0	0	0	0	0	0	0	0	0	0	0	0
Vilagarcía B2	0	0	0	0	0	0	0	0	0	0	0	0	0
A Pobra E.1	0	0	0	0	0	0	0	0	0	0	0	0	0
A Pobra E.2	0	0	0	0	0	0	0	0	0	0	0	0	0
Camariñas A	20.0	29.0	31.0	12.0	21.0	5.0	16.0	0.0	0.0	0.0	0.0	134.0	
Sada 1	0	0	0	0	28.0	19.0	0.0	0	0	0	0	47.0	
Sada 2	0	0	0	15.0	31.0	21.0	0	0	0	0	0	67.0	
Muros B	0	0	0	0	16.0	4.0	0	0	0	0	0	20.0	
Muros A	0	0	0	0	16.0	0	0	0	0	0	0	16.0	
Noia A	0	0	0	0	7.0	0	0	0	0	0	0	0	7.0

Imagen 21. Ejemplo archivo diascierre_ayer.csv

3. *Scrap24h.py* construye un archivo *abiertocerrado_scrap.csv* (**Imagen 22**) que contiene los estados de ría actuales, comparando la tabla a la que se accede en el paso anterior y el archivo *diascierre_ayer.csv*. Es decir, si es día 16 de julio y *abiertocerrado_scrap.csv* tiene 1 fila que contiene los estados de rías del 14 de julio sólo, al ejecutar *scrap24h.py*, realizará una comparación con de *diascierre_ayer.csv* con el contenido actual de la tabla de INTECMAR que muestra los días de cierre por rías, y generará otra fila en *abiertocerrado_scrap.csv* con día 15 de julio y estados de rías. Entonces *abiertocerrado_scrap.csv* tendrá 2 filas, del 14 y 15 de julio respectivamente.

dia	Baiona A	Vigo A	Redondela E	Redondela D	Redondela C	Redondela B	Redondela A	Cangas B	A Pobra A	Baiona A	Vigo A	Redondela E	Redondela D	Redondela C	Redondela B	Redondela A
2024-07-14	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2024-07-15	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Imagen 22. Ejemplo archivo abiertocerrado_scrap.csv

4. *Forecast24h.py* lee el archivo *abiertocerrado1.csv* (**Imagen 23**), que contiene el histórico de estados de rías.

dia	Cangas B	A Pobra A	Baiona A	Vigo A	Redondela E	Redondela D	Redondela C	Redondela B	Redondela A
16/1/2014	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17/1/2014	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18/1/2014	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Imagen 23. Ejemplo archivo abiertocerrado1.csv

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

5. **Forecast24h.py** generá predicciones para todas las rías que tengas datos históricos en todas las fechas de *abiertocerrado1.csv*. Almacena las predicciones en diferentes archivos para cada ría, *forecast_[ría].csv*, siendo *[ría]* la ría predicha en cuestión.

ds,yhat
2024-07-17,0.2217384808260282
2024-07-18,0.21482694812094522
2024-07-19,0.1976027674792269
2024-07-20,0.18931696840940826
2024-07-21,0.18052076064955289
2024-07-22,0.1713574928477345

Imagen 24. Ejemplo archivo forecast_A Pobra A.csv

6. **Forecast24h.py** generá gráficos representativos de las predicciones futuras para cada una de las rías en archivos *forecast_[ría].png*, siendo *[ría]* la ría predicha en cuestión.



Imagen 25. Ejemplo archivo forecast_A Pobra A.png

7. **Forecast24h.py** generá porcentajes de acierto históricos y rangos de entrenamiento para cada ría, que almacena en *porcentaje_rangofechas.csv*.

zona,porcentaje,rango
Cangas B,69.3,2014-01-16 - 2023-12-31
A Pobra A,95.4,2014-01-16 - 2023-12-31
Vilagarcía A,97.3,2014-01-16 - 2023-12-31
Sada 1,71.2,2014-01-16 - 2023-12-31

Imagen 26. Ejemplo archivo porcentaje_rangofechas.csv

Ciclo trimestral:

- **3monthly.sh:** se encarga de ejecutar los scripts de Python que se ejecutan trimestralmente.

1. *integrate365y.py* lee los archivos *abiertocerrado_scrap.csv* y *abiertocerrado1.csv* y los combina, sobreescribiendo *abiertocerrado1.csv*, para mantener el histórico actualizado.

Siempre activo:

1. *app.py* inicia el servidor y configura la lógica del formulario, que muestra imágenes *forecast_[ría].png*, siendo ría el ítem seleccionado en el formulario, y muestra el porcentaje de acierto y rango de fechas de entrenamiento accediendo a *porcentaje_rangofechas.csv* y filtrando por ría.

9. Gestión de datos e información

En el contexto de desarrollo de Mytilus, la gestión efectiva de los datos disponibles es esencial, ya que son la base de las predicciones realizadas. Por ende, garantizar la calidad y coherencia de estos datos es un aspecto fundamental de la herramienta.

El formato de los archivos utilizados en todos los puntos de la herramienta son **CSV**³, como se muestra en el ejemplo de la **Imagen 27**. Esto nos permite **almacenar datos tabulares de manera estructurada**. Es ampliamente utilizado por su simplicidad y facilidad de manipulación. La elección de este modelo de datos también se basa en las herramientas de las que dispone *Python* para el procesamiento y manejo de archivos *csv*, como importación, exportación y manipulación de manera eficiente.

zona,porcentaje,rango
Cangas B,69.3,2014-01-16 - 2023-12-31
A Pobra A,95.4,2014-01-16 - 2023-12-31
Vilagarcía A,97.3,2014-01-16 - 2023-12-31
Sada 1,71.2,2014-01-16 - 2023-12-31

Imagen 27. Ejemplo de archivo CSV de la herramienta

Los archivos *csv* se **almacenan localmente en el equipo o servidor que aloja la herramienta**. Se implementa un sistema de gestión de archivos que asegura la organización y la disponibilidad de los archivos, facilitando así accesos y manipulaciones. Esta elección permite mantener el control de los datos, adaptándose a las necesidades específicas de nuestro proyecto sin depender de servicios de almacenamiento en la nube. En este caso, *mytilus2/app/data* es el directorio encargado de almacenar todos los archivos de datos, según lo indicado en **Imagen 28**.

³ CSV es la nomenclatura de archivos separados por comas.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

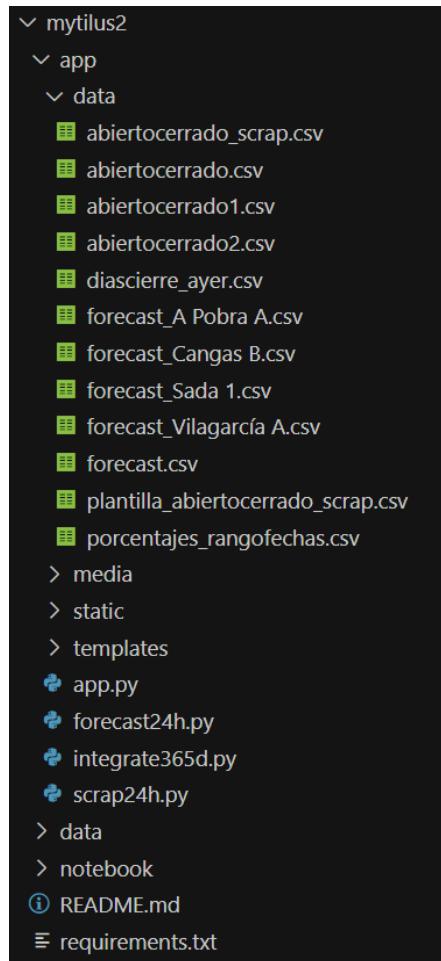


Imagen 28. Estructura de directorios de la herramienta

Para asegurar la integridad y calidad, se implementan **validaciones automáticas** en el proceso de carga y almacenamiento de *csv*. Se utilizan técnicas de validaciones de formato, así como procesos en el código de la herramienta que aseguran que no existen datos faltantes o erróneos.

El acceso a los datos se gestiona a través de accesos a los archivos mediante las **rutas locales relativas**. Esto facilita el desarrollo en mecanismos de lectura y exportación de archivos *csv*, así como la manipulación de los archivos.

La estrategia de backup y recuperación de datos se basa en las **actualizaciones recurrentes del repositorio git** donde se ubica el proyecto. Al estar los archivos de datos almacenados dentro del mismo directorio que el proyecto, la recuperación de datos se puede realizar mediante **mecanismos proporcionados por git**, permitiendo ver el historial de cambios del archivo (**Imagen 29**) y si se precisara, la recuperación de un estado anterior del archivo.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

The screenshot shows a terminal window displaying a CSV file named 'diascierre_ayer.csv'. The terminal interface includes a top bar with file navigation icons and a status bar indicating the file path. The main area is a table with columns for revision number, date, author, and commit message. The commits are color-coded: blue for additions, red for deletions, and green for modifications. The commit history shows several changes made to various locations (Vilagarcía, A Pobra, Sada, Muros, Cambados, Grove, Redondela, Vigo, Baiona) across different dates (e.g., 2024-01-03, 2024-01-04, 2024-01-05, 2024-01-06, 2024-01-07, 2024-01-08, 2024-01-09, 2024-01-28, 2024-01-29, 2024-01-30, 2024-01-31, 2024-02-01, 2024-02-02, 2024-02-03, 2024-02-04).

app/data/diascierre_ayer.csv			
...	6	@@ -3,7 +3,7 @@	Vilagarcía B1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
3	3	- Vilagarcía B2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
4	4	+ A Pobra E.1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
5	5	- A Pobra E.2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
6	6	- Camariñas A,20.0,29.0,31.0,12.0,21.0,5.0,0,0,0,0,0,0,118.0	
	6	+ Camariñas A,20.0,29.0,31.0,12.0,21.0,5.0,1.0,0,0,0,0,0,119.0	
7	7	Sada 1,0,0,0,0,28.0,19.0,0,0,0,0,0,0,0,0,47.0	
8	8	Sada 2,0,0,0,0,15.0,31.0,21.0,0,0,0,0,0,0,0,67.0	
9	9	Muros B,0,0,0,0,16.0,4.0,0,0,0,0,0,0,0,0,20.0	
...	...	@@ -28,7 +28,7 @@ Cambados D,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
28	28	Cambados D (Ostra),0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
29	29	Grove A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
30	30	Grove A (Ostra),0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
31	31	- Grove B (Ostra),31.0,29.0,31.0,30.0,31.0,30.0,0,0,0,0,0,0,182.0	
	31	+ Grove B (Ostra),31.0,29.0,31.0,30.0,31.0,30.0,1.0,0,0,0,0,0,0,183.0	
32	32	Grove C1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
33	33	Grove C2,0,0,0,0,10.0,0,0,0,0,0,0,0,0,10.0	
34	34	Grove C3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
...	...	@@ -54,4 +54,4 @@ Redondela C,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
54	54	Redondela D,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
55	55	Redondela E,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
56	56	Vigo A,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	
57	57	- Baiona A,0,14.0,31.0,30.0,31.0,30.0,0,0,0,0,0,0,136.0	
	57	+ Baiona A,0,14.0,31.0,30.0,31.0,30.0,1.0,0,0,0,0,0,0,137.0	

Imagen 29. Historial de cambios de archivo CSV

En resumen, la gestión de datos e información en Mytilus, basado en el manejo de archivos csv, abarca desde el procesamiento de datos tabulares hasta la implementación de accesos controlados, validaciones y estrategias de recuperación. Estas medidas aseguran la integridad y disponibilidad de los datos en cualquier momento y al mismo tiempo permiten un procesamiento y almacenamiento sin necesidad de contar con muchos recursos.

10. Pruebas llevadas a cabo

Para garantizar que Mytilus cumple con todas las especificaciones requeridas y asegurar su calidad, se llevan a cabo diversas **pruebas exhaustivas sobre la herramienta**.

Las pruebas se llevaron a cabo al final de cada *Sprint* que involucra directamente desarrollo de código, es decir, del *sprint* 1 al 5. Estas pruebas incluyeron:

- **Pruebas unitarias:** para verificar el correcto funcionamiento de las unidades individuales del código.
- **Pruebas de integración:** Para asegurarse de que los diferentes módulos de Mytilus funcionan correctamente cuando se integran.
- **Pruebas de sistema:** Para evaluar el sistema completo y asegurarse de que cumple con los requisitos especificados.

10.1. Pruebas unitarias

Estas pruebas se realizaron a nivel de capa de arquitectura **de forma individual**, en otras palabras, se realizaron pruebas para obtención, procesamiento, predicción y presentación, pero sin relacionarse unas con las otras.

Para la **capa de recolección**, se realizaron las siguientes pruebas:

- **Prueba de conexión:** para garantizar que el archivo que almacena los datos de cierres de rías se actualice diariamente, la conexión debe ser exitosa. Por ello se realiza una prueba simulando que la llamada a la web de INTECMAR devuelve un *status_code* de error. Gracias a ello, se identifica el siguiente error.
- **Error y solución:** no se gestiona si no hay conexión con la web de INTECMAR. Se añade un mensaje que se muestra desde la consola del servidor que informa de que no se puede acceder a la página.
- **Prueba de segundas ejecuciones diarias:** para constatar de que el almacenamiento de archivos es correcto y que no se sobrescribe ningún archivo sin planearlo previamente. Se realiza una prueba ejecutando el script de *scrapeo* dos veces el mismo día. Gracias a ello, se identifica el siguiente error:
 - **Error y solución:** no se gestiona que el script se ejecute dos veces el mismo día, por lo que genera que el archivo que guarda los días de cierre de rías de ayer se sobrescriba. Para solucionarlo, se cambia la lógica del script, haciendo que se ejecutó el script el día actual, no se sobrescriba el archivo que guarda los días de cierre de ayer
- **Gestión de fines de semana:** para asegurarse que los fines de semana el script pueda obtener datos válidos igualmente. Se realiza una prueba ejecutando el script de *scrapeo* diferentes días de semana, incluyendo sábados y domingos. Gracias a ello, se identifica el siguiente error:
 - **Error y solución:** INTECMAR no actualiza la tabla que utilizamos para montar nuestro archivo de estado de rías los fines de semana. Para solucionarlo, se gestionan los fines de semana de forma diferente al resto de días, añadiendo el valor del viernes de la misma semana a sábado y domingo.

Para la **capa de procesamiento**, se realizan las siguientes pruebas:

- **Prueba de rías vacías:** para comprobar que las rías que se van a utilizar en la predicción tienen datos. Se realiza una prueba ejecutando el script que lee los datos de estados de las rías. Gracias a ello, se identifica el siguiente error:
 - **Error y solución:** Como en la carga inicial de datos históricos no se incluyen todas las rías, solo algunas rías tienen valor para todas las fechas. Se añade una comprobación que verifica qué rías tienen valor en

todas las fechas, y si no tienen, no se utilizan para la predicción, hasta que tengan valor en todas las fechas del dataset histórico.

Para la **capa de predicción**, se realizan las siguientes pruebas:

- **Ajuste en datos de entrenamiento:** para comprobar que los datos de entrenamiento son usados de forma correcta. Se realizan pruebas utilizando varios métodos de división de datos. Gracias a ello, se identifica:
 - **Error y solución:** Los métodos de división de datos tradicionales como 80/20 utilizan datos futuros para realizar test en los pasados. Al ser un modelo basado en series temporales, no nos interesa esto. Por ello, se implementa un nuevo modelo de división de datos, *TimeSplitSeries*, aumentando el porcentaje de acierto en un 5% aproximadamente.
- **Ajuste en curva de crecimiento:** para comprobar si la tendencia muestra variación significativa año a año. Se realizan pruebas utilizando varios parámetros de predicción de *Prophet*. Se identifica el siguiente error:
 - **Error y solución:** La curva de crecimiento del gráfico de las predicciones de los estados de las rías ofrece una mayor precisión del acierto cuanto más se acerque a ser plana. Por lo tanto, se ajusta el modelo para que el crecimiento sea plano, aumentando el porcentaje de acierto en más de un 10% histórico.
- **Pruebas de visualización gráficos:** para comprobar que el gráfico es lo más sencillo y accesible posible. Se realizan varias pruebas generando distintos gráficos.
 - **Error y solución:** Un gráfico con mucha información puede llegar a ser confuso para los usuarios con menos experiencia. Se diseña un nuevo gráfico mucho más sencillo, basado en colores con significados asociados y una leyenda explicativa, simplificando el gráfico y, a su vez, hacerlo más claro.

Para la **capa de presentación**, se realizan las siguientes pruebas:

- **Prueba de carga de la página:** para comprobar que en la primera carga de la web se muestra una información inicial. Se realizan pruebas de recargas de la página web y se identifica:
 - **Error y solución:** Al cargar la web por primera vez, la interfaz no mostraba información relativa a ninguna ría porque no se gestionaba. Se modifica el código de forma que al cargar el DOM por primera vez, se muestra la información relacionada con el primer campo del formulario de rías.

Una vez solucionados los errores identificados en las pruebas unitarias, **los resultados obtenidos son los esperados, por lo que cumplen con la especificación**.

10.2. Pruebas de integración

Estas pruebas también se realizaron a nivel de capa de arquitectura, pero esta vez **de forma colectiva**. Se realizaron pruebas para la obtención, procesamiento, predicción y presentación y de las relaciones entre cada una de ellas.

Como cada capa solo se relaciona con su anterior y posterior, se realizan pruebas en el siguiente conjunto de relaciones:

- **Obtención-procesamiento:** para verificar que la integración de las capas de obtención y procesamiento funciona de la forma esperada. Se realiza una prueba que consiste en conectar la salida que otorga la capa de obtención con la entrada de la capa de procesamiento.
- **Procesamiento-predicción:** para comprobar que la integración de las capas de procesamiento y predicción funciona según las expectativas. Se lleva a cabo una prueba donde se conecta la salida de la capa de procesamiento con la entrada de la capa de predicción.
- **Predicción-presentación:** para asegurar que la integración de las capas de presentación y predicción funciona de manera adecuada. Se realiza una prueba en la que se establece una conexión entre lo que la capa de obtención produce y lo que la capa de procesamiento recibe como entrada.

Ninguna de las pruebas identifica ningún error. Esto se debe a la **modularidad de las capas de la herramienta**. Como cada capa produce como salida el resultado esperado, la siguiente capa, va obtener como entrada el resultado esperado. Por lo tanto, los resultados obtenidos de las pruebas de integración **cumplen con la especificación requerida**.

10.3. Pruebas de sistema

Estas pruebas evaluaron el sistema al completo **de forma integral**. Para ello, se simuló un ciclo completo de la herramienta, dividido en 2 secuencias:

- **Secuencia diaria:** para comprobar que el ciclo de ejecución diaria funciona de forma correcta. Se realiza el ciclo de ejecución diario completo durante una semana, que trata de la obtención y procesamiento diaria de datos de cierre de rías, predicción para los próximos 3 meses y presentación.
- **Secuencia trimestral:** para verificar que el ciclo de ejecución trimestral funciona de la manera esperada. Para ello, se simula un ciclo trimestral ejecutando el script de integración de datos obtenidos al histórico, para entrenar y probar las predicciones.

Ninguna de las pruebas identifica ningún error. Debido a las pruebas unitarias y de integración realizadas anteriormente, el sistema funciona de la forma que se espera, **cumpliendo así con la especificación requerida**.

11. Manual de usuario

Para asegurar la correcta instalación y uso de Mytilus, se ha documentado todo el proceso detalladamente en el **anexo 16.2**.

12. Principales aportaciones

La principal aportación generada por este TFG ha sido **la creación de una herramienta innovadora para la predicción del estado de las rías** en el sector mejillonero gallego, Mytilus. Sin embargo, esta herramienta genera también una serie de contribuciones adicionales:

- **Contribuciones al conocimiento:** El TFG ha enriquecido el conocimiento del desarrollo de software y la predicción de estados ambientales, en este caso, del sector del mejillón. Gracias a esta documentación y la herramienta, se ha proporcionado un marco de trabajo para proyectos futuros que busquen un resultado similar, incluso en otros ámbitos sin relación con el mejillón.
- **Innovaciones técnicas:** El desarrollo ha introducido varias innovaciones técnicas, entre ellas destacan la implementación de algoritmos avanzados para la predicción del estado de las rías y el desarrollo de una interfaz web intuitiva. Estas innovaciones mejoran la precisión de predicciones y optimizan el uso de la herramienta.
- **Mejora de soluciones existentes:** Como se explica en el punto de investigación del estado del arte existente, existe una herramienta de predicción de cierre de rías. Sin embargo, esta herramienta predice a muy corto plazo el cierre. Mytilus no se limita a predecir el cierre en un espacio de tiempo corto, sino que trata de dar una solución para los espacios de tiempo medio y largo, permitiendo así gestionar las operaciones del sector de forma más efectiva, reduciendo gastos y optimizando recursos.
- **Aplicaciones prácticas:** las aplicaciones prácticas de Mytilus son variadas. Desde permitir a los trabajadores del sector mejillonero de Galicia ver la predicción del estado de las rías con mayor precisión hasta servir de estudio o modelo para otro tipo de proyectos.
- **Impacto social y económico:** mejorando la capacidad de predecir el cierre de rías, se ayuda a los trabajadores a planificar mejor sus actividades, reduciendo pérdidas y optimizando la producción en todo el ciclo de vida del mejillón. Esto no beneficia solo a los productores, sino que también tiene un efecto positivo en la población y el medio ambiente.
- **Contribución a la formación:** proporcionando un caso práctico de desarrollo de software y gestión de datos, se apoya a estudiantes y profesionales interesados tanto en los modelos de predicción como en los aspectos que rodean al mundo del mejillón.

- **Herramientas y recursos generados:** el proyecto ha generado varias herramientas y recursos útiles, como una interfaz web para la visualización de predicciones, scripts para la recolección y conversión de datos, y archivos de datos con información histórica del cierre de rías. Estos recursos pueden ser reutilizados y adaptados para proyectos diferentes.
- **Efectos a largo plazo:** a largo plazo, las innovaciones introducidas por Mytilus tienen el potencial de transformar la gestión del sector mejillonero y otros sectores similares. La precisión en las predicciones pueden llevar a una mayor sostenibilidad y eficiencia, beneficiando tanto a los productores como al medio ambiente.
- **Originalidad:** la originalidad de este proyecto radica en su propósito y el sector en el que se enfoca. Al ser un sector mayormente familiar, no existen grandes innovaciones tecnológicas a lo largo del tiempo. La propuesta de crear una herramienta completa que ayude a los trabajadores a organizar sus recursos mediante predicciones de estados de rías destaca por su creatividad y aplicabilidad.

13. Conclusiones

Durante el desarrollo de Mytilus, se lograron **importantes avances técnicos** que contribuyeron a mi formación académica y profesional.

13.1. Conclusiones técnicas

En términos técnicos, se han alcanzado **varios hitos significativos**. Se implementaron con éxito la recolección de datos históricos de cierre de rías, el procesamiento de estos datos, la predicción futura de probabilidades de cierre y la presentación mediante una interfaz web. Estas fueron desarrolladas con *python*, con el apoyo de librerías como prophet, flask, sklearn, pandas, numpy, entre otros. También se utilizaron lenguajes utilizados en desarrollo web como *HTML*, *CSS* y *JS*.

A lo largo del desarrollo, aparecieron **desafíos técnicos** como la forma de recolección de datos, para el que se desarrolló un script específico con una lógica específica para obtener los datos de cierre de rías diariamente, la adaptación de prophet a variables binarias, que se solucionó transformando el output en probabilidades, o la creación de gráficos suficientemente fáciles de entender, que se corrigió coloreando zonas con colores descriptivos dentro de rangos específicos, demostrando así la capacidad de resolver problemas técnicos.

13.2. Conclusiones personales

Desde una perspectiva personal, el desarrollo de Mytilus ha sido una **experiencia profundamente enriquecedora**. He podido fortalecer mis habilidades técnicas en programación, específicamente en *Python*, *HTML*, *CSS* y *JavaScript*, así como en la gestión de proyectos utilizando metodologías ágiles como Scrum. Además, he desarrollado habilidades blandas como la gestión del tiempo y la comunicación efectiva.

Los desafíos personales que he enfrentado, como llevar a cabo todas las tareas por mí mismo, desde la redacción de la documentación hasta la realización de pruebas y la interacción con expertos del sector, me han brindado la oportunidad invaluable de experimentar todo el ciclo de vida de un proyecto. Estas experiencias también han contribuido significativamente al desarrollo de mis habilidades de resolución de problemas y mi capacidad de perseverancia.

En conjunto, esta experiencia ha sido **excepcionalmente valiosa para mí**. Me ha permitido comprender que dedicar tiempo y esfuerzo a un proyecto que me entusiasma y que percibo como útil es tremadamente satisfactorio. Por lo tanto, recomendaría a futuros estudiantes que **enfoquen sus trabajos finales de grado en temas que les apasionen y donde sientan que pueden ofrecer una solución concreta a un problema dentro de un sector que les interese**.

14. Vías de trabajo futuro

En el presente TFG, se han abordado diversas fases del ciclo de vida del desarrollo de software, incluyendo la planificación, el diseño, la implementación y las pruebas. Sin embargo, este proyecto deja abiertas **varias vías de trabajo futuro a explorar**, entre las que destacan:

- **Añadir más rías para mostrar predicciones:** el proceso de recolección de datos histórico es manual. Debido a la limitación de tiempo establecida para la realización de este TFG, no se han podido añadir datos de más rías a la herramienta, debido al coste en horas que ello supone. En futuras actualizaciones, se prevé integrar datos históricos de todas las rías gallegas, lo que permitirá ofrecer predicciones a los trabajadores del sector mejillonero de toda Galicia.
- **Mejoras en la gestión de datos:** tal como se explica en la vía de trabajo futuro anterior, la recolección de datos históricos es manual. Esto se debe a que INTECMAR expone públicamente los datos históricos de cierre de rías en formato pdf, de la siguiente manera, previamente explicada en la **Imagen 1**. En este punto, existen dos posibles soluciones: obtener los datos de otra manera, como *API* o *webservice*, o desarrollar una herramienta que extraiga y convierta la tabla del PDF a formato *CSV*. La opción más viable sería obtener los datos por otros medios. Sin embargo, si esto no es posible, crear una herramienta específica para convertir la tabla del PDF a *CSV* también sería una alternativa adecuada.
- **Añadir más funcionalidades en la interfaz web:** la primera versión de Mytilus tan solo implementa la visualización de predicciones mediante gráficos con el fin de ser fácilmente entendibles. Añadir una versión para las personas que busquen más profundidad en las predicciones también es una de las funciones a incorporar en la herramienta en futuras actualizaciones. Además, explorar la

opción de añadir otras funcionalidades como la visualización a tiempo real del estado de rías también puede ser interesante para los usuarios finales.

- **Mejoras en la predicción:** a pesar de tener un porcentaje óptimo de acierto histórico en el estado de las rías, investigar y añadir nuevos parámetros que influyan positivamente en la predicción puede ser beneficioso para mejorar los resultados.

Es crucial seguir explorando diversas vías de desarrollo identificadas y no identificadas en este TFG para mejorar el conjunto de Mytilus y su aplicabilidad. Estos desarrollos futuros no sólo fortalecerán el impacto de la herramienta, sino que también contribuirán al avance de la investigación del mundo del mejillón.

15. Referencias

1. Xunta de Galicia. (n.d.). La primera potencia mundial en cultivo y. Recuperado de <https://emigracion.xunta.gal/es/conociendo-galicia/galicia-multimedia/galicia-50-prodigios/la-primer-a-potencia-mundial-cultivo-y>
2. Fundación Observatorio Español de Acuicultura. (n.d.). Cuaderno del mejillón. Recuperado de https://www.observatorio-acuicultura.es/sites/default/files/images/adjuntos/libros/cuaderno_mejillon.pdf
3. Instituto Tecnológico para el Control del Medio Marino de Galicia (Intecmar). (n.d.). Recuperado de <http://www.intecmar.gal/>
4. Instituto Tecnológico para el Control del Medio Marino de Galicia (Intecmar). (n.d.). Evolución de biotoxinas. Recuperado de <http://www.intecmar.gal/Informacion/biotoxinas/Evolucion/DiagramaBateas.aspx>
5. Instituto Tecnológico para el Control del Medio Marino de Galicia (Intecmar). (n.d.). Recuperado de <http://www.intecmar.gal/Informacion/biotoxinas/Evolucion/Default.aspx?sm=a6>
6. Atlassian. (n.d.). Scrum. Recuperado de <https://www.atlassian.com/es/agile/scrum>
7. Sommer, U. (1989). Phytoplankton ecology: Structure, function and fluctuation. Recuperado de https://books.google.es/books?hl=es&lr=&id=IL59CAAAQBAJ&oi=fnd&pg=P_A1937&dq=Phytoplankton+Ecology:+Structure,+Function+and+Fluctuation&ots=32voDZdU8Q&sig=suInqmS_Fu53wv5yeKKux6nnRmI#v=onepage&q=Phytoplankton%20Ecology%3A%20Structure%2C%20Function%20and%20Fluctuation&f=false

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

8. Fundación Observatorio Español de Acuicultura. (n.d.). Cuaderno del mejillón. Recuperado de https://www.observatorio-acuicultura.es/sites/default/files/images/adjuntos/libros/cuaderno_mejillon.pdf
9. Instituto Tecnológico para el Control del Medio Marino de Galicia (Intecmar). (n.d.). Recuperado de <http://www.intecmar.gal/Ctd/Default.aspx>
10. Empromar. (n.d.). Recuperado de <https://empromar.com/>
11. Instituto Español de Oceanografía. (n.d.). Índice de afloramiento. Recuperado de <http://www.indicedeafloramiento.ieo.es/afloramiento.html>
12. Instituto Tecnológico para el Control del Medio Marino de Galicia (Intecmar). (n.d.). Recuperado de <http://www.intecmar.gal/Deinteres/MareasVermellas.aspx?sm=b>
13. Xunta de Galicia. (n.d.). Portal de notificaciones. Recuperado de <https://notifica.xunta.gal/notificaciones/portal/portada>
14. Amazon Web Services. (n.d.). Recuperado de <https://aws.amazon.com/es/>
15. Reactive Programming. (n.d.). Estilos arquitectónicos: Capas. Recuperado de <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas>
16. Python Software Foundation. (n.d.). Python. Recuperado de <https://www.python.org/>
17. Pallets Projects. (n.d.). Flask. Recuperado de <https://flask.palletsprojects.com/en/3.0.x/>
18. Python Software Foundation. (n.d.). CSV. Recuperado de <https://docs.python.org/3/library/csv.html>
19. Python Software Foundation. (n.d.). OS. Recuperado de <https://docs.python.org/es/3.10/library/os.html>
20. pandas. (n.d.). Recuperado de <https://pandas.pydata.org/>
21. Python Software Foundation. (n.d.). Datetime. Recuperado de <https://docs.python.org/3/library/datetime.html>
22. Facebook. (n.d.). Prophet: Forecasting at scale. Recuperado de https://facebook.github.io/prophet/docs/quick_start.html
23. Scikit-learn. (n.d.). Recuperado de <https://scikit-learn.org/stable/>
24. Matplotlib. (n.d.). Recuperado de <https://matplotlib.org/>

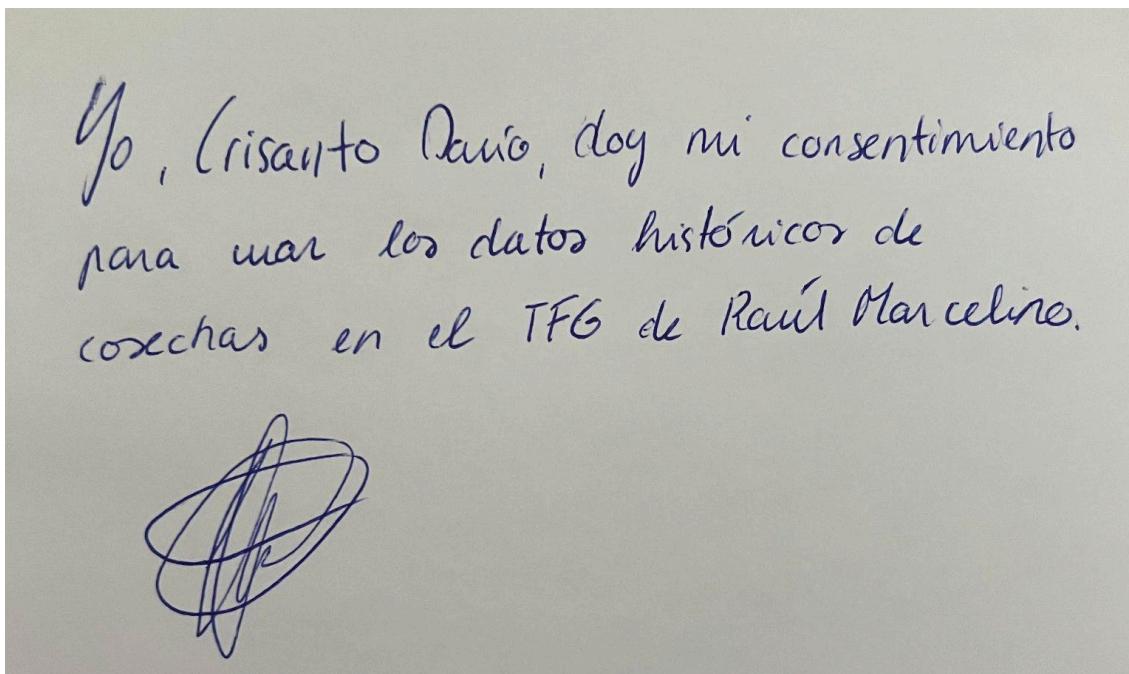
Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

25. NumPy. (n.d.). Recuperado de <https://numpy.org/>
26. PyPI. (n.d.). Requests. Recuperado de <https://pypi.org/project/requests/>
27. PyPI. (n.d.). Beautiful Soup 4. Recuperado de <https://pypi.org/project/beautifulsoup4/>
28. Git. (n.d.). Recuperado de <https://git-scm.com/>
29. Jupyter. (n.d.). Recuperado de <https://jupyter.org/>
30. Mozilla Developer Network. (n.d.). HTML. Recuperado de <https://developer.mozilla.org/es/docs/Web/HTML>
31. Mozilla Developer Network. (n.d.). CSS. Recuperado de <https://developer.mozilla.org/es/docs/Web/CSS>
32. Mozilla Developer Network. (n.d.). JavaScript. Recuperado de <https://developer.mozilla.org/es/docs/Web/JavaScript>
33. Notion. (n.d.). Recuperado de <https://www.notion.so/es-es>
34. Google Docs. (n.d.). Recuperado de <https://www.google.es/intl/es/docs/about/>
35. GitHub. (n.d.). Recuperado de <https://github.com/>
36. Lucidchart. (n.d.). Recuperado de <https://www.lucidchart.com/>
37. Mozilla Firefox. (n.d.). Recuperado de <https://www.mozilla.org/es-ES/firefox/new/>
38. Visual Studio Code. (n.d.). Recuperado de <https://code.visualstudio.com/>
39. Visure Solutions. (n.d.). Functional vs. non-functional requirements. Recuperado de <https://visuresolutions.com/es/requirements-management-traceability-guide/functional-vs-non-functional-requirements>

16. Anexos

16.1. Consentimiento de uso de datos



16.2. Manual de usuario

16.2.1 Requisitos mínimos

16.1.1.1 Software

- **Sistema operativo:** Windows 10 o 11, Linux.
- **Python:** Versión 3.x (preferiblemente 11.9 o 12.4).
- **Control de versiones:** Git.
- **Navegador web:** Se recomienda Firefox 127 o superior.
- **IDE (opcional):** Se recomienda Visual Studio Code 1.91 o superior con extensión de Python instalada.

16.1.1.2. Hardware

Los requisitos mínimos de hardware están determinados por los requisitos mínimos de software, particularmente aquellos especificados por los sistemas operativos Windows y Linux.

- **Procesador:** 64 bits, 1GHz.
- **Almacenamiento:** 32GB.
- **Memoria RAM:** 2GB.
- **Conexión a internet:** sí.

16.2.2. Manual de instalación

A continuación se muestran los pasos de instalación de la herramienta para sistemas operativos Windows y Linux.

16.2.2.1 Instalación de requisitos de software mínimos

En caso de necesitar instalar alguno de los requisitos mínimos, se siguen los siguientes tutoriales:

- **git:** <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Instalaci%C3%B3n-de-Git>
- **VSCode:** <https://code.visualstudio.com/download>
- **Firefox:** <https://www.mozilla.org/es-ES/firefox/new/>
- **Python:** <https://kinsta.com/knowledgebase/install-python/>

16.2.2.2 Despliegue en Windows

1. Clonado de proyecto

Una vez se disponen de todas las herramientas necesarias para desplegar el proyecto en un equipo local, se procede a obtener el código fuente del proyecto.

Abrimos una terminal de Símbolos del sistema o Windows Powershell.

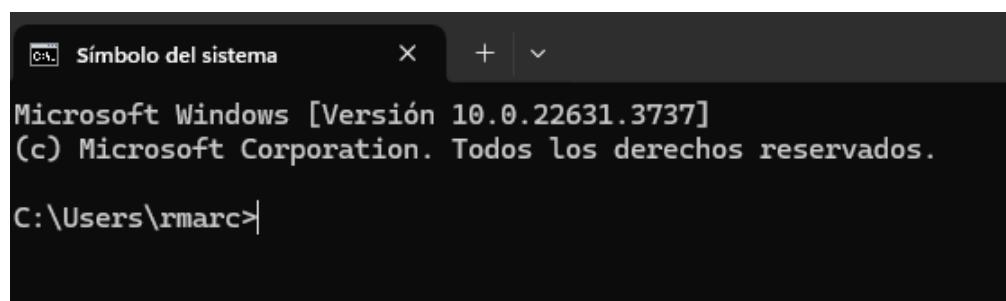


Imagen 30. Consola de Windows

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Y nos situamos en el directorio que guardará nuestro código fuente de la herramienta. En este caso, se crea una carpeta *Mytilus-install* con el comando *mkdir*.

mkdir mytilus-install

Inmediatamente después, nos situamos en la carpeta creada con el comando *cd*.

cd mytilus-install

```
Microsoft Windows [Versión 10.0.22631.3737]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\rmarc>mkdir mytilus-install

C:\Users\rmarc>cd mytilus-install

C:\Users\rmarc\mytilus-install>
```

Imagen 31. Comandos de creación y acceso a carpeta

Y se ejecuta el siguiente comando.

git clone https://github.com/rreiul/mytilus2.git

Esto resultará en la creación de una copia del repositorio que alberga el código fuente de la herramienta, permitiéndonos así obtener dicho código fuente.

```
C:\Users\rmarc\mytilus-install>git clone https://github.com/rreiul/mytilus2.git
Cloning into 'mytilus2'...
remote: Enumerating objects: 323, done.
remote: Counting objects: 100% (323/323), done.
remote: Compressing objects: 100% (207/207), done.
remote: Total 323 (delta 134), reused 290 (delta 104), pack-reused 0
Receiving objects: 100% (323/323), 15.69 MiB | 18.92 MiB/s, done.
Resolving deltas: 100% (134/134), done.

C:\Users\rmarc\mytilus-install>
```

Imagen 32. Clonado de proyecto

2. Creación de entorno virtual

Una buena práctica cuando se trabaja con *Python* es el uso de entornos virtuales. Estos permiten aislar de manera efectiva los entornos de desarrollo de diferentes proyectos. Para que las librerías no entren en conflicto si existen otros proyectos en el equipo, procedemos a crear un entorno virtual.

python3 -m venv venv

Y procedemos a activarlo

.\venv\Scripts\activate

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Si esto produce el siguiente error

```
PS C:\Users\rmarc\mytilus-install\mytilus2> .\venv\Scripts\activate
.\venv\Scripts\activate : No se puede cargar el archivo
C:\Users\rmarc\mytilus-install\mytilus2\venv\Scripts\Activate.ps1 porque la ejecución de scripts está deshabilitada en
este sistema. Para obtener más información, consulta el tema about_Execution_Policies en
https://go.microsoft.com/fwlink/?LinkId=135170.
En linea: 1 Carácter: 1
+ .\venv\Scripts\activate
+ ~~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

Imagen 33. Error de permisos de ejecución de scripts

ejecutamos el siguiente comando

Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass

Y presionamos ‘O’ para seleccionar ‘Sí a todo’.

Una vez hecho esto, volvemos a intentar activarlo

.\venv\Scripts\activate

```
PS C:\Users\rmarc\mytilus-install\mytilus2> .\venv\Scripts\activate
(venv) PS C:\Users\rmarc\mytilus-install\mytilus2>
```

Imagen 34. Entorno virtual activado

El prompt de la consola cambiará, y deberá verse algo parecido a la **imagen 13**.

3. Instalación de requirements.txt

Una vez tenemos el entorno creado y activado, procedemos a instalar las dependencias de nuestro proyecto.

Para ello, ejecutamos el siguiente comando.

pip install -r .\requirements.txt

Esto empezará a instalar paquetes de las dependencias necesarias para que la herramienta funcione.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

```
Using cached pywin32-306-cp311-cp311-win_amd64.whl (9.2 MB)
Using cached wheel-0.43.0-py3-none-any.whl (65 kB)
Installing collected packages: wcwidth, pywin32, pytz, pytweening, pytups, PyRect, pyperclip, PyMsgBox, pure-eval, namex, libclang, flatbuffers, fastjsonschema, XlsxWriter, wrapt, wheel, urllib3, tzdata, typing_extensions, traitlets, tornado, threadpoolctl, termcolor, tensorflow-data-server, tenacity, soupsieve, six, pyzmq, PyYAML, pyparsing, Pygments, PyGetWindow, psutil, protobuf, prompt-toolkit, platformdirs, pillow, parso, packaging, numpy, nest-asyncio, MouseInfo, mdurl, MarkupSafe, Markdown, lxml, kiwisolver, joblib, itsdangerous, importlib_resources, idna, grpcio, greenlet, gast, fontTools, executing, et-xmlfile, decorator, debugpy, Cython, cycler, colorama, charset-normalizer, certifi, blinker, annotated-types, absl-py, Werkzeug, tqdm, stano, SQLAlchemy, scipy, requests, python-dateutil, PySqueeze, pydantic_core, plotly, patcy, optree, opt-einsum, openpyxl, ml-dtypes, matplotlib-inline, markdown-it-py, Mako, jupyter_core, Jinja2, jedi, h5py, google-pasta, contourpy, comm, colorlog, click, beautifulsoup4, attrdict, astunparse, asttokens, wikipedia, tensorflow, stack-data, scikit-learn, rich, pydantic, PyAutoGUI, pandas, matplotlib, jupyter_client, holidays, Flask, alembic, statsmodels, seaborn, pywhatkit, optuna, keras, Ipython, cmdstanpy, arm-mango, scikeras, prophet, ipykernel
Successfully installed Cython-3.0.10 Flask-3.0.3 Jinja2-3.1.4 Mako-1.3.5 Markdown-3.6 MarkupSafe-2.1.5 MouseInfo-0.1.3 PyAutoGUI-0.9.54 PyGetWindow-0.0.9 PyMsgBox-1.0.9 PyRect-0.2.0 PySqueeze-0.1.30 PyYAML-6.0.1 Pygments-2.18.0 SQLAlchemy-2.0.30 Werkzeug-3.0.3 XlsxWriter-3.1.9 absl-py-2.1.0 alembic-1.13.1 annotated-types-0.7.0 arm-mango-1.4.3 asttokens-2.4.1 astunparse-1.6.3 attrdict-2.0.1 beautifulsoup4-4.12.3 blinker-1.8.2 certifi-2023.7.22 charset-normalizer-3.3.0 click-8.1.7 cmdstanpy-1.2.2 colorama-0.4.6 colorlog-6.8.2 comm-0.2.2 contourpy-1.2.1 cycler-0.12.1 debugpy-1.8.1 decorator-5.1.1 et-xmlfile-1.1.0 executing-2.0.1 fastjsonschema-2.18.1 flatbuffers-24.3.25 fonttools-4.51.0 gast-0.5.4 google-pasta-0.2.0 greenlet-0.3.3 grpcio-1.63.0 h5py-3.11.0 holidays-0.49 idna-3.4 importlib_resources-6.4.0 ipykernel-6.29.4 ipython-8.24.0 itsdangerous-2.2.0 jedi-0.19.1 joblib-1.4.6 jupyter_client-8.6.1 keras-3.3.3 kiwisolver-1.4.5 li-bclang-18.1.1 lxml-5.2.2 markdown-it-py-3.0.0 matplotlib-3.8.4 matplotlib-inline-0.1.7 mdurl-0.1.2 ml-dtypes-0.3.2 namex-0.0.8 nest-asyncio-1.6.0 numpy-1.26.1 openpyxl-3.1.2 opt-einsum-3.3.0 optree-0.11.0 optuna-3.6.1 packaging-24.0 pandas-2.2.2 parso-0.8.4 patcy-0.5.6 pillow-10.3.0 platformdirs-4.2.1 plotly-5.22.0 prompt-toolkit-3.0.43 prophet-1.1.5 protobuf-4.25.3 psutil-5.9.8 pure-eval-0.2.2 pydantic-2.4.2 pydantic_core-2.10.1 pyparsing-3.1.2 pyperclip-1.8.2 python-dateutil-1-2.8.2 pytups-0.86.2 pytweening-1.2.0 pytz-2023.3.post1 pywhatkit-5.4 pywin32-306 pyzmq-26.0.3 requests-2.31.0 rich-13.7.1 scikeras-0.13.0 scikit-learn-1.4.2 scipy-1.13.0 seaborn-0.13.2 six-1.16.0 soupsieve-2.5 stack-data-0.6.3 stano-0.5.0 statsmodels-0.14.2 tenacity-8.3.0 tensorflow-2.16.2 tensorflow-data-server-0.7.2 termcolor-2.4.0 threadpoolctl-3.5.0 tornado-6.4 tqdm-4.66.1 traitlets-5.14.3 typing_extensions-4.11.0 tzdata-2024.1 urllib3-2.0.7 wcwidth-0.2.13 wheel-0.43.0 wikipedia-1.4.0 wrapt-1.16.0

[notice] A new release of pip is available: 24.0 -> 24.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip
[venv] PS C:\Users\rmarc\mytilus-install\mytilus2>
```

Imagen 35. Dependencias instaladas

4. Prueba de funcionamiento

Finalmente, probamos que todos los archivos .py del proyecto funcionan de la forma esperada.

Para ello, probamos a ejecutar *app.py* como se muestra en la **Imagen 13**.

```
cd .\app\
.\app.py
```

```
[venv] PS C:\Users\rmarc\mytilus-install\mytilus2> cd .\app\
[venv] PS C:\Users\rmarc\mytilus-install\mytilus2\app> .\app.py
[venv] PS C:\Users\rmarc\mytilus-install\mytilus2\app>
```

Imagen 36. Ejecución de app.py

Esto abrirá otra consola, que nos proporciona una IP local con la cual, desde cualquier navegador, acceder a la interfaz web de la herramienta.

```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.11_3.11.3280.0_x64__psd3q1jw6tj3p
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.144:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 351-207-083
```

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Imagen 37. Resultado de la ejecución de app.py

Accedemos a 127.0.0.1:5000 y comprobamos que la herramienta funciona correctamente.



Imagen 38. Interfaz web de Mytilus

16.2.2.3. Despliegue en Linux

Debido a que todo el despliegue en Windows se hace mediante comandos de unix con WSL⁴, el despliegue en Linux se hará de la misma manera que en Windows, exceptuando:

- **Activación del entorno virtual:** se utiliza el comando `source /venv/Scripts/activate` en lugar de `.\venv\Scripts\activate`
- **Prueba de funcionamiento:** se utiliza el comando `python app.py` en lugar de `.\app.py`

16.2.3 Manual de uso

16.2.3.1. Mediante servidor web

Para que la herramienta sea accesible sin necesidad de instalación, se implementa la herramienta al completo en un servidor web.

El servidor se encarga de ejecutar automáticamente los scripts y de mantener activa la interfaz web, eliminando así la necesidad de que el usuario se preocupe por la ejecución manual de los scripts.

Para acceder al servidor web, abrimos una pestaña del navegador y accedemos a la siguiente URL.

⁴ Subsistema de Windows para Linux es una capa de compatibilidad desarrollada por Microsoft para ejecutar binarios de Linux nativamente en Windows

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

<http://15.188.50.47:5000/>

Una vez dentro, se nos muestra la interfaz web de la herramienta. Para interactuar con la interfaz, debemos desplegar el formulario y seleccionar la ría que nos interese.



Imagen 39. Interfaz web

En este caso de ejemplo, vamos a visualizar los datos de *Vilagarcía A*.



Imagen 40. Interfaz web con gráfico de Vilagarcía A

En la parte superior se muestra un aviso con información del gráfico. En él, nos indica que el gráfico está generado a partir de una predicción para la zona Vilagarcía A, y a continuación se nos muestra el porcentaje de acierto histórico y el rango de fechas de los datos usados para generar predicciones.

En la parte de abajo se muestra un gráfico temporal de los próximos 3 meses dividido en zonas de colores, que nos muestra la probabilidad de que la ría este cerrada en esas fechas.

El color rojo representa un 100-80% de probabilidad, el naranja, 80-60%, el amarillo, 60-40%, el verde, 40-20%, y el azul, 20-0%. También se muestra una leyenda más fácil de entender para los usuarios menos experimentados con este tipo de tecnologías.

Para visualizar datos de otra ría diferentes, desplegamos el formulario, como se muestra en la siguiente imagen.

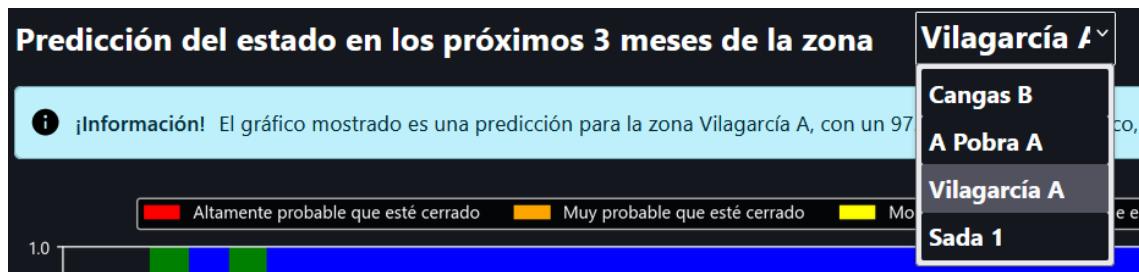


Imagen 41. Formulario de iteración de rías

Este es el **modo de uso recomendado** para Mytilus.

16.2.3.2. Simulando comportamiento del servidor en equipo local

Si queremos simular el comportamiento de la herramienta en nuestro equipo local debemos:

1. Acceder a la ubicación del proyecto

En este apartado, se hará uso de Visual Studio Code para facilitar la ejecución de los scripts.

Abrimos Visual Studio Code y abrimos la carpeta que contiene el código fuente del proyecto.

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

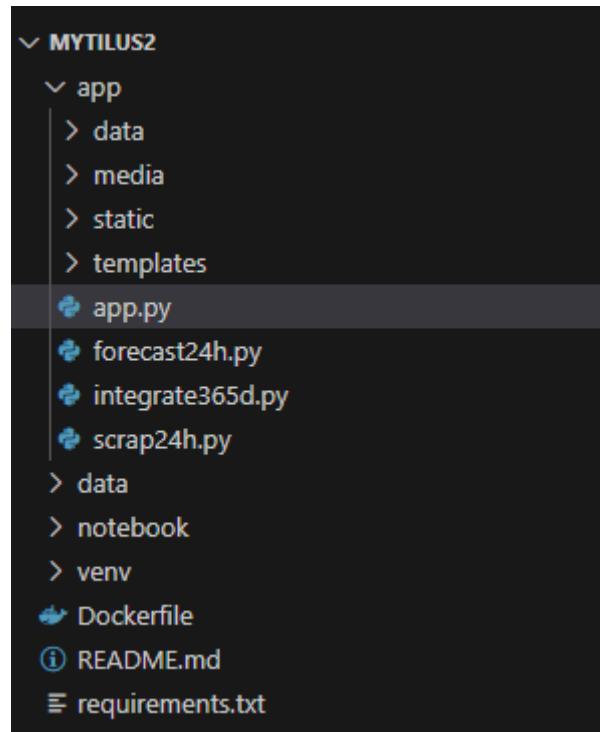


Imagen 42. Carpeta del proyecto

2. Ejecución de scripts

Primero, ejecutamos *scrap24h.py*:

Imagen 43. Ejecución de script scrap24h.py

La ejecución de este script añadirá una fila al archivo *abiertocerrado_scrap.csv*, con la fecha de ayer y el estado para cada una de las rías y reemplazará el archivo *diascierre_ayer.csv* con los datos de hoy.

```
dia,Baiona A,Vigo A,Redondela E,Redondela D,Redondela C,Red  
2024-07-09,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,  
2024-07-10,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
```

Imagen 44. Archivo abiertocerrado_scrap.csv

Una vez hecho esto, ejecutamos el script *forecast24h.py*. Esto generará las predicciones y gráficos representativos para las rías que dispongan de datos históricos.

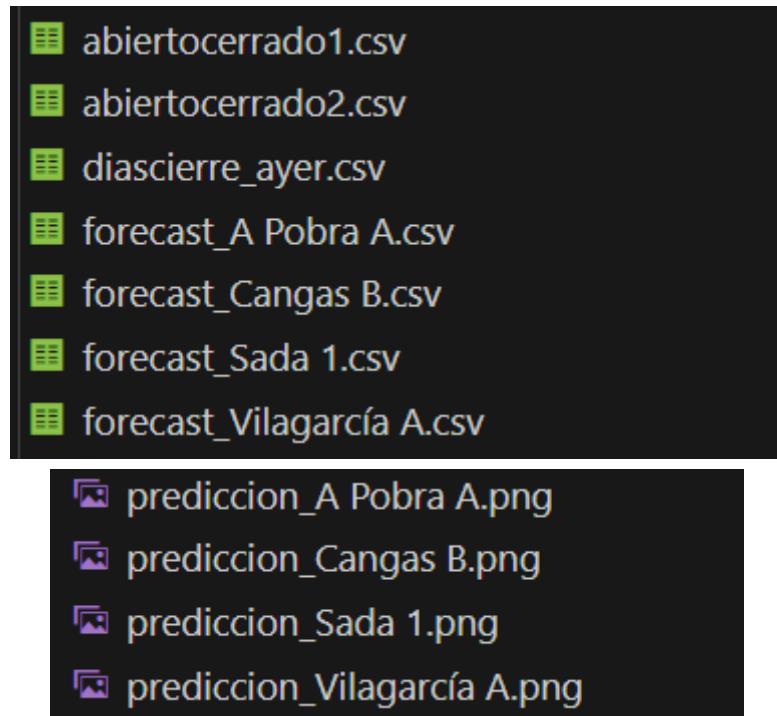


Imagen 45. Resultados de la ejecución de forecast24h.py

Por último, ejecutamos *app.py*, y accedemos a la URL mostrada por el terminal de Visual Studio Code.

```
PS C:\Users\rmarc\mytilus-install\mytilus2> & C:/Users/rmarc/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/rmarc/PycharmProjects/mytilus2/main.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.144:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 351-207-083
```

Imagen 46. Ejecución de app.py

Si queremos simular la integración de los datos *scrapeados* con los datos históricos, debemos ejecutar *integrate3m.py*. Esto combinará los archivos de datos *scrapeados* con los datos históricos en *abiertocerrado1*. (Hay que tener en cuenta que hasta que la ría tenga valor en todos los días del histórico, no se usará para entrenamiento ni predicción)

Imagen 47. Resultado de la ejecución de `integrate3m.py`

3. Acceso a la interfaz web

Mytilus: Herramienta predictiva de cierre de rías gallegas

Raúl Darío Marcelino Docío - 2024

Accedemos a la URL mostrada por el terminal de Visual Studio Code, generalmente será *127.0.0.1:5000*.



Imagen 48. Interfaz web de Mytilus