Proyecto de Predicción meteorológica de Barcelona

¿Necesitamos llevar paraguas y/o abrigarnos?

Axel Ortega, Juan Paredes & Julio J. Vázquez
STUCOM Centre d'Estudis - Pelai

Contenido

1.- Comprensión del Negocio

Identificar los objetivos del negocio

1.1 Determinar Objetivos de Negocio:

1.1.1 Contexto:

El proyecto nos sale de la necesidad de predecir las condiciones meteorológicas diarias para saber si tenemos que llevar ropa de abrigo, un paraguas o llevarnos una botella de agua cuando vengamos a la escuela Stucom.
Se centra por ello en la predicción de temperaturas y precipitaciones a corto plazo.

1.1.2 Objetivos del negocio:

El objetivo principal es mejorar la toma de decisiones cotidiana de los alumnos mediante una herramienta de asistencia meteorológica simplificada.
Se busca reducir la incertidumbre del usuario al elegir su vestimenta o accesorios diarios, transformando datos técnicos complejos en recomendaciones prácticas y visuales.

El éxito del negocio se medirá por la capacidad de la herramienta para ofrecer una recomendación útil que coincida con la percepción real del clima del usuario, como:

- Si necesitas llevar paraguas o no.
- Si necesitas llevar chaqueta o no.
- Si necesitas llevar una botella de agua o no.

El objetivo pretende evitar que te conviertas en un muñeco de nieve o que la gente salga de casa pareciendo que va a una expedición al Ártico, o que parezca que te has caído en una piscina por olvidar el paraguas.

1.1.3 Criterios de éxito empresarial:

Lograr que nuestra predicción sea lo más cercano a predecir el clima de la semana mediante el resultado de la Aplicación, consiguiendo:

- **Reducción de errores de vestimenta:** Que el usuario reporte que la recomendación de "chaqueta" fue acertada en su uso diario.

- **Usabilidad y Adopción de la herramienta:** Que el usuario prefiera usar nuestra sencilla y dinámica interfaz de Streamlit antes que mirar por la ventana o sacar el pie por la ventana, de ser así hemos fallado.

- **Utilidad práctica (**supervivencia urbana):
    - evitar pasar frio (y no resfriarnos), mojarnos y en caso de calor hidratarnos

Planificación y alineación (Acciones ejecutadas):

- Definición de umbrales de confianza: Hemos establecido que el nivel de precisión debe ser como mínimo del 60% en precipitación y un error máximo absoluto en temperatura normal y outliers de 2ºC para que el usuario lo considere fiable para cambiar su comportamiento diario (por ejemplo, decidir no llevar paraguas).

- Relación datos con necesidad diaria: hemos realizado el mapeo entre la necesidad del usuario (comodidad/salud) y qué variables meteorológicas (temperatura/lluvia) permiten satisfacerla.

- Consenso de criterios de aceptación: Se han definido formalmente qué condiciones debe cumplir la aplicación para que el negocio la considere un éxito, separando el rendimiento del algoritmo de la utilidad de la interfaz.

Expectativas de Viabilidad (Resultados obtenidos):

- **Viabilidad:** Los criterios de éxito se consideran alcanzables basándonos en la disponibilidad de datos históricos de alta resolución proporcionados por **AEMET** y **OpenWeather**. Estas fuentes ofrecen las variables críticas (precipitación y temperatura …) necesarias para entrenar modelos de aprendizaje supervisado.
  La viabilidad técnica se apoya en el uso de algoritmos de regresión y clasificación contrastados en la literatura científica y proporcionados en este curso para este tipo de predicciones climáticas.

- **Requisitos funcionales establecidos:** Se cuenta con una lista clara de las tres alertas necesarias (chaqueta, paraguas, agua) que guiarán el desarrollo posterior.

- **Alineación estratégica:** Se confirma que el proyecto es realizable con los recursos actuales (Python/Streamlit) y que responde directamente a la problemática de "incertidumbre climática" del usuario.

1.2 Evaluación de la Situación:

1.2.1 Inventario de Recursos, Requisitos, Supuestos y Restricciones:

A. Inventario de Recursos:
   - **Recursos Humanos:** Equipo de 3 analistas de datos con roles en: Ingeniería de datos (Extracción), Data Science (Modelado) y Desarrollo (Interfaz Streamlit).
   - **Hardware:** Estaciones de trabajo personales con capacidad de procesamiento local.
   - **Software:** Entorno de desarrollo Visual Studio Code, Python 3.13, librerías especializadas (Pandas, Scikit-Learn, Streamlit). Control de desarrollo del proyecto mediante GitHub para tener un repositorio de trabajo común y acceso al personal.

- **Datos:** Acceso a las APIs de AEMET y oneweather. Disponemos de un histórico consolidado desde enero de 2010.

B. Requisitos:
   - Conexión de red para acceder a centros de datos
   - Capacidad de almacenamiento local de los archivos csv.
C. Supuestos:
   - Asumimos que las estaciones del Museo marítimo y del puerto Olímpico son la mejor representación del clima para ir a STUCOM.
   - Se asume que los datos históricos de AEMET y oneweather han pasado por un control de calidad previo por parte de la agencia oficial.
D. Restricciones:
   - Temporal: finalizar el proyecto en la fecha establecida.
   - Técnicas: limitación de acceso a las API's gratuitas ya que restringen el volumen de peticiones por minuto y fecha. Ejemplo AEMET solo permite sacar franjas de 6 meses por petición, y solo se puede hacer una petición por 1 minuto.

1.2.2 Riesgos y Contingencias:

Riesgo de datos:

- **Riesgo:** Inconsistencia o "huecos" (nulos) en los datos de AEMET y OpenWeather debido a fallos en las estaciones de medición.
- **Contingencia:** Implementación de técnicas de imputación de datos (rellenar huecos con la media, el valor anterior…)

Riesgos Técnicos:

- **Riesgo:** Sobreajuste (*Overfitting*) del modelo. El modelo aprende los datos históricos de memoria, pero falla estrepitosamente al intentar predecir el futuro real.
- **Contingencia:** Uso de validación cruzada (*Cross-Validation*) y ajuste de hiperparámetros.

- **Riesgo:** Caída de las APIs o cambios en el formato del JSON/CSV de origen.
- **Contingencia:** Creación de programas para la unificación de diferentes archivos.

Riesgo de gestión:

- **Riesgo:** El tiempo de desarrollo supera la fecha de entrega (el clásico "se me ha echado el tiempo encima").
- **Contingencia:** Objetivos y funcionalidad del proyecto. Aseguraremos primero la predicción de temperatura y lluvia antes de intentar predecir cualquier otra cosa.

1.2.3 Terminología:

Para asegurar la correcta interpretación de los informes técnicos y resultados del modelo, se definen los siguientes términos clave:

- **MAE (Mean Absolute Error):** Métrica de rendimiento para modelos de regresión que mide el promedio de los errores absolutos entre las predicciones y los valores reales. Se expresa en la misma unidad que la variable (grados Celsius).

- **RMSE (Error Cuadrático Medio):** Métrica de rendimiento para modelos de regresión que mide la raíz cuadrada del promedio de los cuadrados de los errores. A diferencia del MAE, esta métrica penaliza más los errores grandes (por ejemplo, predecir 20°C cuando hay 30°C), lo que es crítico para evitar que el usuario pase frío extremo por una mala predicción.

- **R^2** (Coeficiente de determinación): Métrica estadística que indica qué proporción de la varianza de la temperatura es explicada por nuestro modelo. Se expresa en una escala de 0 a 1; cuanto más cercano a 1, mejor representa el modelo el comportamiento real del clima.

- **Accuracy:** Métrica de rendimiento para modelos de clasificación (como Lluvia/No Lluvia) que mide el porcentaje total de aciertos sobre el total de casos.

- **Precisión (Precisión de clase):** Mide la calidad de las predicciones positivas. Indica qué porcentaje de los días que el modelo predijo lluvia, realmente llovió.

- **Recall (Sensibilidad):** Proporción de casos positivos reales que fueron identificados correctamente por el modelo. En este proyecto, un Recall alto en "Lluvia" significa que el modelo es muy bueno detectando días lluviosos y rara vez te dejará sin paraguas cuando hace falta...

- **F1-Score:** Muestra el equilibrio entre la precisión y el Recall, escala del 0 al 1, a mas cerca del 1 mejor es el balance, es decir evitamos mas situaciones en las que que te decimos que va a llover y no llueve (FP) y situaciones en las que te decimos que NO llueve y llueve (FN).

- **Random Forest:** Algoritmo de aprendizaje supervisado basado en la combinación de múltiples árboles de decisión para mejorar la precisión y evitar el sobreajuste. Es, básicamente, consultar a una "asamblea de árboles" en lugar de a uno solo para tomar la decisión más sabia.

- **Time Series (Series Temporales):** Secuencia de datos registrados en intervalos de tiempo sucesivos. Es la base de nuestro proyecto, ya que el clima de hoy depende fuertemente del de ayer.

1.2.4 Costes y beneficios:

Costes nuestro caso:

- **Software y Datos:** 0€. El uso de API's gratuitas (AEMET/OpenWeather) y librerías de código abierto (Python/Streamlit) permite un ahorro del 100% en licencias.
- **Hardware y energía**: Coste marginal de amortización de equipos y consumo eléctrico en especial el ventilador del portátil de tanto entreno.
- **Personal:** Nuestro maravilloso tiempo.
  Caso real: Se estima una dedicación de 3 consultores de datos durante 4 semanas (aprox. 160 horas totales). Con una tarifa base de 25€/h para perfiles junior, el coste de oportunidad asciende a 4.000€.

Los beneficios que obtenemos:

- Adquisición de conocimientos al procesar datos y realizar modelos de predicción
- Optimización de toma de decisiones diarias para salir de casa, dando lugar a un ahorro en medicamentos al evitar resfriados, evitando pasar frio y/o mojarnos, o en caso de un día caluroso estar deshidratados.

1.3 Determinar Objetivos de Minería de Datos:

1.3.1 Objetivos de la minería de datos:

El objetivo técnico es transformar los datos históricos climáticos en modelos predictivos capaces de analizar el comportamiento de la atmósfera en Barcelona.
Para ello, se definen las siguientes metas de minería de datos:

- **Modelado de Regresión (Temperatura):** Desarrollar un modelo de regresión para predecir las temperaturas máximas y mínimas con un margen de error reducido, permitiendo una recomendación precisa sobre la ropa a llevar.

- **Modelado de Clasificación Binaria (Lluvia):** Implementar un clasificador que determine la probabilidad de precipitación (Clase 0: No llueve / Clase 1: Llueve). El objetivo es maximizar el *Recall* para minimizar los falsos negativos evitando casos donde el usuario se moja por error del sistema.

- **Ingeniería de Características (Time Series):** Generar nuevas variables mediante técnicas de *Lagging* (retardos) y *Rolling Windows* (ventanas móviles) para capturar la estacionalidad y la inercia térmica de la región.

1.3.2 Criterios de éxito de la minería de datos:

Para considerar que la minería de datos ha sido exitosa, los modelos generados deben cumplir con los siguientes umbrales técnicos:

- **Métricas de Regresión (Temperatura):**
  - El modelo debe alcanzar un **MAE (Mean Absolute Error) inferior a 2°C** en el conjunto de test. No queremos que diga que hace un día de playa y acabemos con los dedos azules.
  - Un **RMSE** (Error cuadrático medio) inferior a 2ºC para el conjunto del test, ya que nos avisara de cuanto se desvía la temperatura en los extremos que es cuando hace más frio o más calor.
  - El coeficiente de determinación **R^2 debe ser superior a 0.85**, indicando que el modelo captura la gran mayoría de la variabilidad térmica de Barcelona.
- **Métricas de Clasificación (Precipitación):**
  - Se busca un **F1-Score superior a 0.60** con un **Recall superior al 70%**. Dado que los días de lluvia son menos frecuentes que los de sol (clases desbalanceadas), no nos basta con el *Accuracy*; necesitamos que el modelo sea preciso cuando realmente hay riesgo de mojarse.
- **Generalización:**
  - La diferencia de rendimiento entre los datos de entrenamiento y los de validación no debe superar el 5%. Si el modelo acierta todo en el pasado, pero falla en el presente, sufriremos de *overfitting* (sobreajuste), y eso se considerará un fracaso técnico.

Acciones ejecutadas:

- **Selección de métricas de evaluación:** Se han elegido el **RMSE** y el **Recall** y **F1-Score** como KPI's técnicos para asegurar que el modelo sea útil en situaciones críticas (frío extremo o lluvia imprevista).

- **Establecimiento de protocolos de validación:** Se ha diseñado la estrategia de separar los datos en entrenamiento y test para medir la capacidad de generalización y evitar el sobreajuste.

- **Definición de "Baselines":** Se ha acordado que cualquier modelo que no supere el 0.60 de F1-Score será descartado por no aportar valor suficiente al usuario.

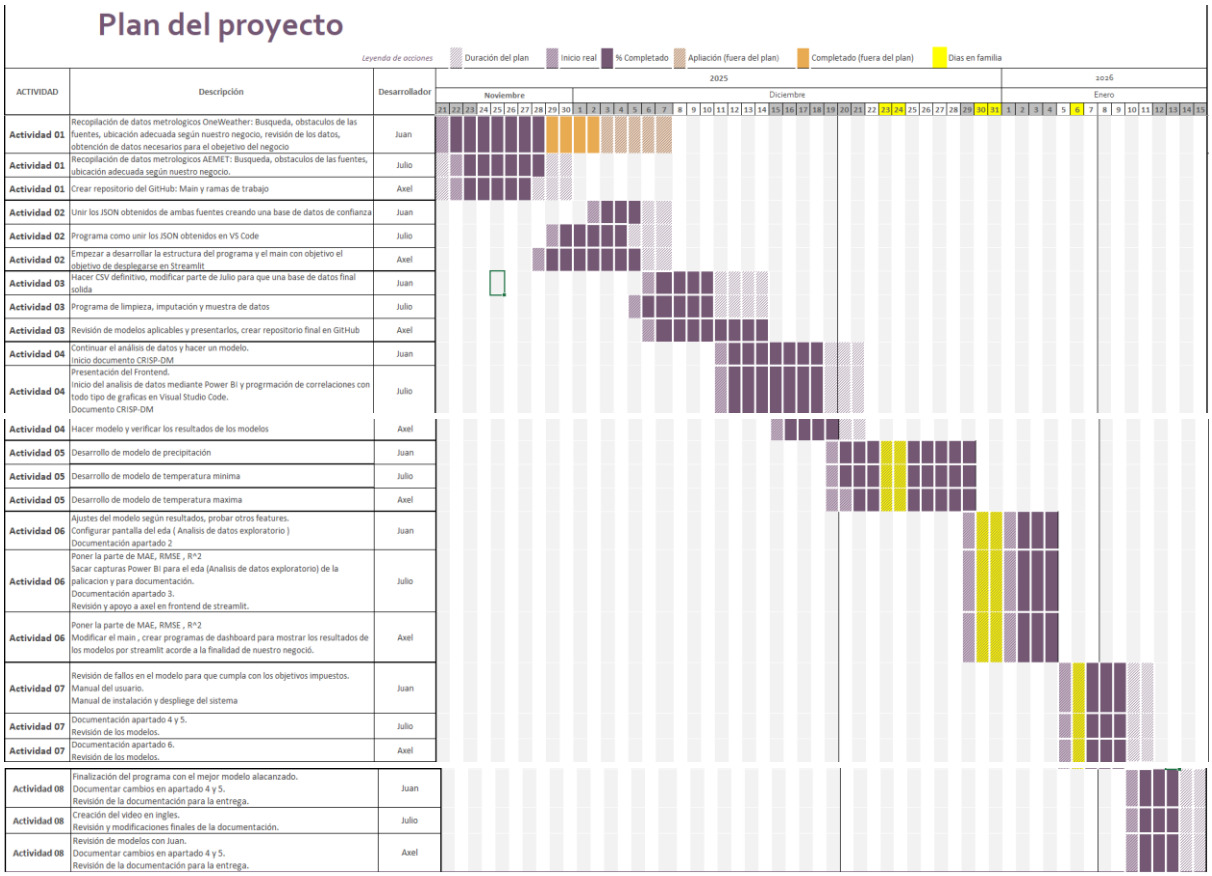
Resultados obtenidos:

- **Umbrales de aprobación:** Documento técnico aprobado que establece que el éxito se alcanza con un **RMSE < 2°C y un F1-Score > 0.60**.

- **Plan de visualización técnica:** Se ha decidido que la **Matriz de Confusión** y los **Gráficos de dispersión** serán las herramientas oficiales para validar estos criterios en la fase de Evaluación.

1.4 Producción del Plan del Proyecto:

1.4.1 Plan del proyecto:



ACTIVIDAD	Descripción	Desarrollador
Actividad 01	Recopilación de datos metrologicos OneWeather: Busqueda, obstaculos de las fuentes, ubicación adecuada según nuestro negocio, revisión de los datos, obtención de datos necesarios para el obejetivo del negocio	Juan
Actividad 01	Recopilación de datos metrologicos AEMET: Busqueda, obstaculos de las fuentes, ubicación adecuada según nuestro negocio.	Julio
Actividad 01	Crear repositorio del GitHub: Main y ramas de trabajo	Axel
Actividad 02	Unir los JSON obtenidos de ambas fuentes creando una base de datos de confianza	Juan
Actividad 02	Programa como unir los JSON obtenidos en VS Code	Julio
Actividad 02	Empezar a desarrollar la estructura del programa y el main con objetivo el objetivo de desplegarse en Streamlit	Axel
Actividad 03	Hacer CSV definitivo, modificar parte de Julio para que una base de datos final solida	Juan
Actividad 03	Programa de limpieza, imputación y muestra de datos	Julio
Actividad 03	Revisión de modelos aplicables y presentarlos, crear repositorio final en GitHub	Axel
Actividad 04	Continuar el análisis de datos y hacer un modelo. Inicio documento CRISP-DM	Juan
Actividad 04	Presentación del Frontend. Inicio del analisis de datos mediante Power BI y progrmación de correlaciones con todo tipo de graficas en Visual Studio Code. Documento CRISP-DM	Julio
Actividad 04	Hacer modelo y verificar los resultados de los modelos	Axel
Actividad 05	Desarrollo de modelo de precipitación	Juan
Actividad 05	Desarrollo de modelo de temperatura minima	Julio
Actividad 05	Desarrollo de modelo de temperatura maxima	Axel
Actividad 06	Ajustes del modelo según resultados, probar otros features. Configurar pantalla del eda (Analisis de datos exploratorio) Documentación apartado 2	Juan
Actividad 06	Poner la parte de MAE, RMSE , R^2 Sacar capturas Power BI para el eda (Analisis de datos exploratorio) de la palicacion y para documentación. Documentación apartado 3. Revisión y apoyo a axel en frontend de streamlit.	Julio
Actividad 06	Poner la parte de MAE, RMSE , R^2 Modificar el main , crear programas de dashboard para mostrar los resultados de los modelos por streamlit acorde a la finalidad de nuestro negocio.	Axel
Actividad 07	Revisión de fallos en el modelo para que cumpla con los objetivos impuestos. Manual del usuario. Manual de instalación y despliege del sistema	Juan
Actividad 07	Documentación apartado 4 y 5. Revisión de los modelos.	Julio
Actividad 07	Documentación apartado 6. Revisión de los modelos.	Axel
Actividad 08	Finalización del programa con el mejor modelo alacanzado. Documentar cambios en apartado 4 y 5. Revisión de la documentación para la entrega.	Juan
Actividad 08	Creación del video en ingles. Revisión y modificaciones finales de la documentación.	Julio
Actividad 08	Revisión de modelos con Juan. Documentar cambios en apartado 4 y 5. Revisión de la documentación para la entrega.	Axel

*** Adjunto está el pdf del plan del proyecto**

1.4.2 Evaluación inicial de herramientas y técnicas:

Para garantizar la escalabilidad y eficiencia del sistema de predicción, se ha seleccionado lo siguiente:

- **Lenguaje de Programación Python 3.13:** Elegido por su ecosistema maduro en IA y su capacidad de integración con API's meteorológicas. Es un estándar para el tratamiento de series temporales.

- **Entorno de Desarrollo Visual Studio Code & GitHub:** Utilizamos Visual Studio Code para el desarrollo local y GitHub como sistema de trabajo en equipo.

- **Librerías de Procesamiento y Análisis** (Data Preparation) :
    - **Librerías de Procesamiento (Pandas & NumPy):** Fundamentales para la manipulación de las 6.741 líneas de datos, permitiendo realizar el *Resampling* y la limpieza de nulos.
    - **Scikit-learn (Preprocessing):** Imprescindible para el escalado de datos (como StandardScaler o MinMaxScaler), ya que las temperaturas y precipitaciones tienen escalas muy distintas.
    - **Datetime:** Esencial para manejar series temporales, extraer el mes o la hora del día, lo cual es crítico para la meteorología

- **Librerías de Visualización** (Data Understanding / Evaluation)
    - **Matplotlib y Seaborn:** Se utilizan para generar los histogramas y gráficos de dispersión que mencionaste anteriormente para "entender los datos" y para "evaluar los resultados" mediante la matriz de confusión.

- **Modelado Predictivo (Scikit-Learn):** Utilizaremos esta suite para implementar los algoritmos de **Random Forest** (Regresión y Clasificación), aprovechando sus herramientas de validación cruzada.

- **Interfaz y Despliegue (Streamlit):** Seleccionado para convertir nuestros modelos en una aplicación web interactiva de forma rápida. Queremos que el usuario vea predicciones de forma interactiva y dinámica.

- **Fuentes de Datos (API's):** Integración con las API's de **AEMET OpenData** (datos históricos oficiales) y **OpenWeather** (datos actuales).

Acciones ejecutadas:

- Evaluación de alternativas se descartó el uso de Excel por sus limitaciones en el entrenamiento de modelos complejos.

- Configuración de entornos virtuales (venv) para asegurar que las librerías no entren en conflicto entre los equipos de los analistas.

Resultados obtenidos:

- Informe de compatibilidad: Se confirma que todas las herramientas seleccionadas son compatibles entre sí y permiten el despliegue final en la nube si fuera necesario.

2.- Comprensión de los Datos

Recopilación y revisión de datos

2.1 Recopilación de datos iniciales:

Acciones ejecutadas:

- La búsqueda de datos fue difícil para todo el equipo, muchos puntos eran de pago incluso fuentes supuestamente publicas si lo querías descargar de forma fácil, directa, con datos limpios y optimizados.
- Encontramos AEMET OpenData, la cual ofrecía opción de pago y opción gratuita.
    - Opción gratuita:
        - Requisitos:
            - Se hace a través de un API
            - El API requiere una Key
        - Limitaciones de descarga:
            - Fragmentos de datos de 6 meses
            - Solo una petición de descarga cada 1 minuto.
- AEMET solo disponía datos desde 2010 hasta el día de hoy, observamos que en estos nos faltaban algunas características más.
- Encontramos OneWeather, que dada las coordenadas exactas daba datos de las estaciones en mismas ubicaciones, habiendo más datos que necesitábamos.
    - OneWeather ofrece datos desde 1945 hasta el día de hoy.
    - Se descargo desde el 1990 hasta el 2025.
- Se han integrado dos fuentes principales: la API de **AEMET OpenData** (datos oficiales de estaciones terrestres) y **OpenWeather** (datos históricos por coordenadas).

Resultados obtenidos:

- Se ha consolidado un histórico desde el 01-01-2010 hasta la actualidad.
- Aunque se disponía de datos desde 1990, se aplicó un **criterio de exclusión temporal** para asegurar la máxima densidad y calidad de las variables, ya que los registros anteriores presentaban una tasa de nulos superior al 40%.
- Los datos se han extraído en formato JSON y convertido a una estructura tabular CSV para su tratamiento.

2.2 Descripción de datos:

Acciones ejecutadas:

El dataset final consolidado cuenta con aproximadamente 6.700 registros (filas).

Resultados obtenidos:

Dentro de los datos observamos como variables a tener en cuenta de partida:

Fundamentales para los consejos:

- **Temperatura máxima y mínima:** Se seleccionan porque son los indicadores directos para la recomendación de "chaqueta".

- **Precipitaciones:** Se selecciona por ser la métrica necesaria para determinar el uso del "paraguas".

Variables de contexto:

- **humedad y fecha:** Se incluyen basándonos en el conocimiento del dominio (la humedad afecta la sensación térmica y la fecha permite capturar la estacionalidad).

- **dir / velmedia:** Se recogen para evaluar más adelante si el viento influye en la dispersión de las nubes y la temperatura local.

- **cloudcover / dewpoint / surface_pressure:** Se descargan como variables de apoyo para el proceso de exploración, siguiendo la recomendación de expertos en meteorología que indican que la presión y la nubosidad son precursores de cambios en el tiempo.

2.3 Exploración de datos:

Acciones ejecutadas:

Se ha realizado un Análisis Exploratorio de Datos (EDA) utilizando Python y Power BI con la finalidad de:

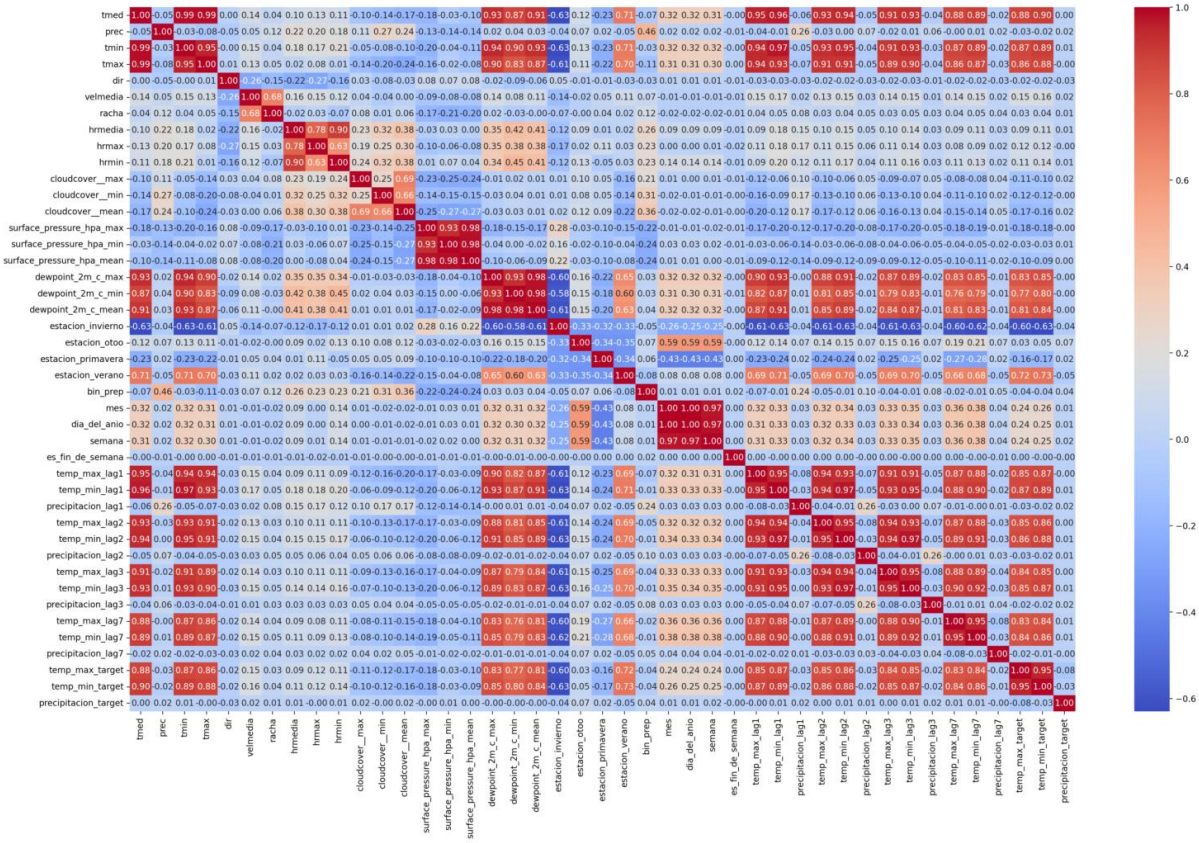
1. Ver si los datos que teníamos eran útiles.
2. Ver que datos tenían correlación entre ellos, facilitando la compresión mediante visualización con: grafías, histogramas y heatmap.
3. Confirmar que no había datos nulos o duplicados.

Resultados obtenidos:

Análisis de Correlación de la matriz HEATMAP:

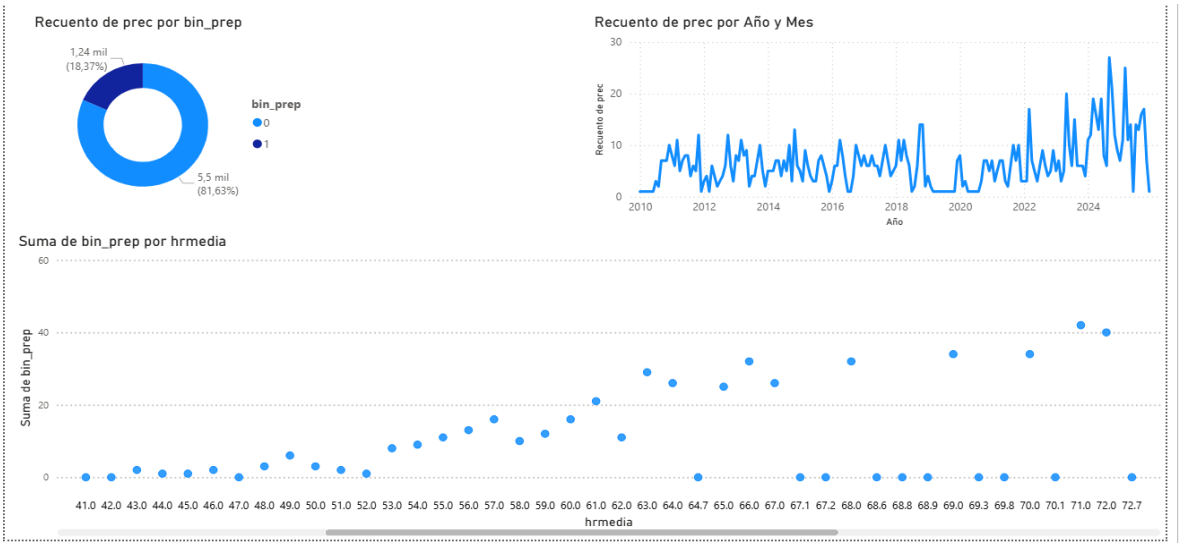
- **Temperaturas y Punto de Rocío (dewpoint):** Existe una correlación altísima (entre **0.87** y **0.94**) entre la temperatura máxima/mínima y el punto de rocío. Esto indica que el punto de rocío es un predictor excelente de la masa de aire térmica.

- **Humedad y Lluvia:** El heatmap muestra que la correlación entre la humedad media (hrmedia) y la variable binaria de lluvia (bin_prep) es de **0.26**, y con la nubosidad (cloudcover_mean) es de **0.31**. Aunque **0.31** no parece un número "alto" (como un 0.9), en meteorología es una **correlación significativa**. Indica que la nubosidad y la humedad son los factores que más "empujan" la probabilidad de lluvia, por encima del viento o la presión



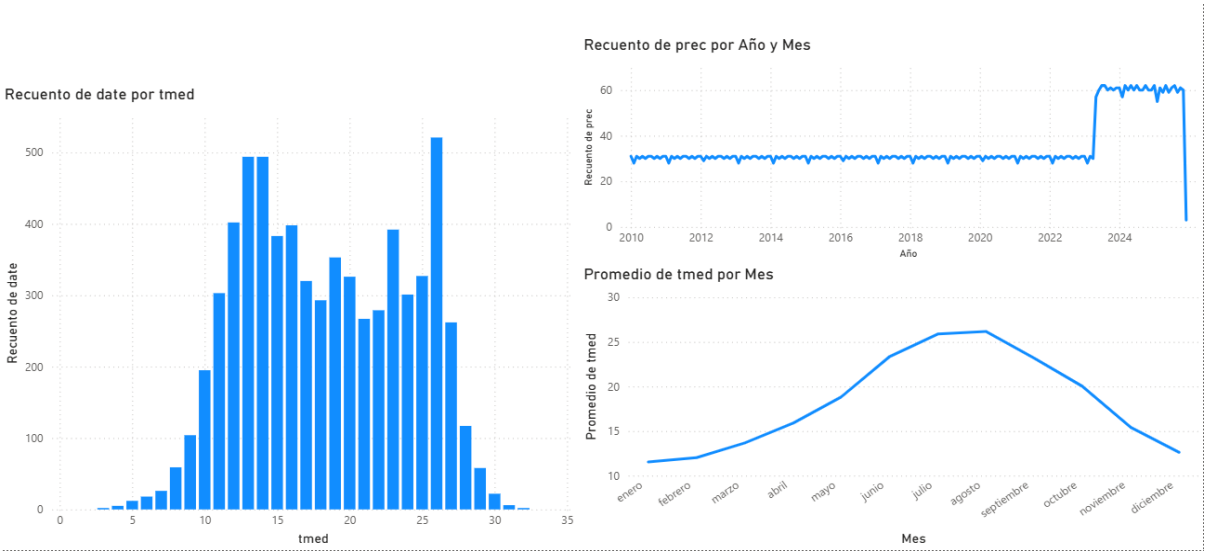
Análisis diversos de precipitación con power BI:

- Se observa los días de lluvia son el 18,37%, algo normal, esto indica un posible desequilibrio de datos de datos a la hora de entrenar.
- Confirmamos una fuerte dependencia entre la humedad relativa y la probabilidad de precipitación o dicho de otra forma cuando la humedad sube y es mejor no tender la colada.



Análisis diversos de temperatura con power BI:

- **Histograma:** donde pusimos la cantidad de veces que se da una temperatura
- **Grafico de líneas:** comportamiento a lo largo del tempo de lluvias y temperatura.



Conclusión: debemos tener en cuenta:

- Estacionalidad a la hora de preparar los datos para los modelos.
- Cuáles son los “outliers”: serian temperaturas extremas en máximo y mínimo.

2.4 Verificación la calidad de los datos

Acciones ejecutadas:

- Realizado el análisis de datos, revisamos los valores nulos y duplicados.
    - En el caso de los nulos podría ser que por ejemplo como en el año 2020 hubo la pandemia se dejaron de recibir datos de los satélites y sensores por no tener mantenimiento temporal.
    - Y los duplicados podría ser que hubiera algún tipo de error a la hora de recoger los datos, un mal funcionamiento o una pequeña avería.
- Verifiquemos que los datos estuvieran normalizados: numérico, fecha…
- Revisamos también que los datos tuvieran coherencia, que por ejemplo la temperatura máxima siempre sea mayor al de la mínima.

Resultados obtenidos:

- **Valores Nulos/Duplicados:** Se detectó un porcentaje marginal de nulos en estaciones secundarias, atribuibles a micro cortes en los sensores o mantenimiento de red. No se requiere "inventar" datos; se aplicarán técnicas de imputación estadística en la siguiente fase.
- **Normalización:** Se confirma que los datos de velocidad y temperatura mantienen unidades métricas coherentes en todas las fuentes.
- Se ha comprobado que en el 100% de los registros la tmax >= tmin.

3.- Preparación de los Datos

Selección y limpieza de datos

3.1 Selección de datos

Los csv que obtuvimos tienen muchas variables, hemos seleccionado aquellos que tienen relevancia para que nuestros modelos predigan correctamente.

Razones de inclusión:

- Variables Térmicas (tmax, tmin, tmed): Fundamentales para cumplir el objetivo de negocio de recomendar vestimenta (abrigo/manga corta).

- Variables de Humedad y Presión: Se incluyen porque la caída de presión y el aumento de humedad son precursores físicos de la lluvia en la cuenca del Mediterráneo.

- Variables Temporales (Mes, Semana): Necesarias para que el modelo entienda la estacionalidad (no es lo mismo una nube en agosto que en noviembre).

Razones de exclusión:

- IDs y Metadatos de API: Descartamos archivos json y columnas en algunos de los archivos ya que no aportan valor estadístico y podrían causar ruido, como el nombre del servidor, nº de estación metrológica ….

- Variables o años con más de la mitad de nulos: Tras un análisis inicial, observamos que los datos antes del 2010 contenían menos variables y faltaban muchos dias sueltos, al tener una carencia masiva de datos fue descartada para no comprometer la integridad del entrenamiento.

Acciones ejecutadas:

Se realizó un filtrado mediante la librería pandas, seleccionando un total de 18 columnas técnicas de los 35 originales disponibles en la unión de AEMET y OpenWeather.

Resultados obtenidos:

El dataset resultante tras la selección mantiene la coherencia para las predicciones de temperatura y clasificación de lluvia.

3.2 Limpieza de datos

Después de seleccionar los datos, pasamos a limpiarlos, donde cogemos nuestros datos en brutos (raw) y los transformamos en un dataset íntegro y sin ruido, asegurando que los modelos no hereden errores de formato o valores físicamente imposibles.
Para llevar a cabo este paso hemos realizado dos programas uno de limpieza y otro de imputación de datos.

3.2.1 Limpieza:

Programa: limpieza.py

Acciones ejecutadas:

1. Eliminar duplicados: Aplicamos una eliminación de duplicados por si los hubiera.

2. Estandarización de Formatos: Se ha ejecutado una limpieza de caracteres mediante el mapeo de funciones para eliminar comillas (") y espacios en blanco residuales procedentes de las tramas JSON de las API's.

3. Normalización de nombres: Conversión automática de todos los nombres de columnas a minúsculas y sustitución de espacios por guiones bajos (_) para evitar errores de sintaxis en el código de entrenamiento.

4. Gestión de la Línea Temporal: Conversión de la columna date a objetos datetime de Pandas y ordenación cronológica estricta. Sin este paso, las técnicas de "Lag" que aplicaremos después no tendrían sentido físico.

5. Adaptación de Tipos: Conversión forzosa de columnas clave (tmed, prec, tmin, tmax, dir) a formato numérico (float64).

Resultados obtenidos:

- Integridad: Se han eliminado registros duplicados, garantizando que cada día en Barcelona sea único en nuestra serie temporal.

- Calidad: Tras la limpieza, el dataset presenta una estructura tabular uniforme, lista para la fase de imputación.

3.2.2 Imputación:

Programa: imputar_datos.py

Dado que el clima funciona en un ciclo cronológico de estaciones, no podemos simplemente borrar los días que nos faltan pues crearíamos saltos en el tiempo. Hemos aplicado las siguientes reglas:

1. Variables Térmicas: Los valores faltantes se han rellenado con la media mensual histórica.
   Ejemplo: Si un día de mayo no tiene temperatura, se le asigna el promedio de los mayos de otros años, preservando la estacionalidad.

2. Precipitación: Se ha aplicado una lógica de "ceros por defecto". Si no hay registro de lluvia, el sistema asume 0 mm, evitando falsos positivos de tormentas inexistentes.

3. Precipitación en binaria: agregamos una columna convirtiendo el evento en si llueve o no llueve, cargamos en una variable siempre que el día haya habido más de 0,1 en los datos de precipitación este será 1 y si es 0.0 entonces 0.

4. Variables Dinámicas (Viento y Presión): Se ha utilizado interpolación lineal, asumiendo que el cambio entre el día anterior y el posterior es progresivo.

Acciones ejecutadas:

Aplicación en las columnas de variables estacionales de la media mensual y en las variables temporales que requieren interpolación y media mensual de:

- df.groupby(df.index.month).transform("mean")
- interpolate(method="linear").

Resultados obtenidos:

El dataset final cuenta con un 0% de valores nulos en las columnas críticas para el entrenamiento.

3.3 Construcción de datos:

En la construcción de datos agregamos nuevas columnas que capturar la naturaleza temporal del clima y preparar los vectores de entrada para los modelos.

3.3.1 Atributos derivados:

Programa: add_lags.py

Acciones ejecutadas:

1. **Generación de Lags:** Se han creado variables de retardo para tmax, tmin y prec (ej: temp_max_lag1). Esto permite que el modelo utilice la información del día anterior para predecir el futuro.

2. **Ventanas Móviles (Rolling Windows):** Se han calculado medias móviles de 7 días (ej: temp_max_rolling_7). Esto suaviza las fluctuaciones diarias y permite al modelo detectar tendencias semanales (olas de calor o frentes fríos persistentes).

3. **Extracción de Estacionalidad:** A partir de la fecha, hemos derivado variables como mes, dia_del_anio, semana y la variable booleana es_fin_de_semana.

4. **Codificación del Target (Variable Objetivo):** Se ha generado la columna bin_prep donde ponemos 1 si prec mayor 0 y 0 en caso contrario para el modelo de clasificación. Además, se han creado los *targets* desplazados a 7 días vista para la predicción a corto plazo.

Resultados obtenidos:

- **Enriquecimiento del Dataset:** El tablón de datos ha pasado de tener información puntual a poseer un contexto histórico. Esto es vital para que un modelo de **Random Forest** entienda que, si ayer llovió y la presión está bajando, la probabilidad de lluvia hoy aumenta.

- **Coherencia Temporal:** Se ha verificado que los retardos coinciden con la ordenación cronológica establecida en la fase anterior.

3.3.2 Registros generados:

Programa: add_lags.py

Acciones ejecutadas:

- Debido al uso de técnicas de *Lag* y *Shift*, las primeras y últimas filas del dataset generan valores nulos, ya que no hay un "día anterior" para el primer registro ni un "objetivo a 7 días" para el último.

- **Acción:** Se ha procedido a eliminar estos registros incompletos para asegurar que el entrenamiento solo se realice con vectores de datos completos.

Resultados obtenidos:

- Se ha mantenido un volumen de datos robusto (cercano a las 6.700 líneas originales), perdiendo únicamente un margen irrelevante de registros en los extremos de la serie temporal.

3.4 Integración de datos:

En la integración de datos se realiza la fusión de las distintas fuentes de información, en nuestro caso de dos fuentes AEMET y OpenWeather, para generar una tabla de datos que sirva de entrada a los modelos de aprendizaje automático.

Datos fusionados:

Para la fusión de datos creamos dos programas compilar.py y unir_json.py, con los cuales unimos todos los datos en la carpeta “raws” de nuestro proyecto, como ya se realizó la

fusión de datos están descartados de ser llamados en el programa principal del proyecto, pero se conservan por si se quisieran integrar nuevos datos en un futuro.
Acciones ejecutadas:

1. **Consolidación de Formatos:** Se ha implementado un proceso iterativo que recorre estructuras de carpetas buscando archivos en formatos .json y .txt.

2. **Gestión de Codificación:** El sistema intenta leer cada archivo utilizando múltiples codificaciones (utf-8, latin1, windows-1252), garantizando que no se pierda información por caracteres especiales o formatos regionales.

3. **Unión Temporal (Join):** Se utiliza la columna date como clave primaria para la integración. El proceso asegura que los datos de la estación del Museo Marítimo y la del Puerto Olímpico se alineen correctamente en la misma línea temporal.

4. **Generación del Dataset Definitivo:** Los DataFrames individuales se concatenan y se exportan al archivo maestro data_weather_final.csv.

Resultados obtenidos:

- **Dataset Unificado:** Se ha logrado pasar de una colección de archivos dispersos a un único CSV estructurado de aproximadamente 6.700 registros.

- **Escalabilidad:** El módulo de integración está diseñado para ser reutilizable; si en el futuro se añaden datos de 2026, el script compilar.py los integrará automáticamente sin necesidad de reprogramar la lógica.

3.5 Formatear datos

En esta etapa final de preparación, adaptamos la estructura del dataset unificado a los requisitos técnicos de los algoritmos de aprendizaje supervisado.
O dicho de otra forma tenemos que adaptarlo de “idioma humano” de los informes metrológicos a “idioma maquina”.

3.5.1 Datos Reformateados:

Acciones ejecutadas:

1. **Indexación Temporal:** Se ha establecido la columna date como índice del DataFrame. Esto no cambia el dato, pero reformatea la estructura para que las funciones de series temporales operen de forma eficiente.

2. **Aplanamiento de Estructuras:** Tras la integración de múltiples JSON, se ha verificado que la estructura sea puramente tabular (filas y columnas), eliminando cualquier rastro de anidamiento propio del formato origen.

3. **Conversiones:** Los datos se han preparado para ser transformados en *arrays* de NumPy antes de entrar al modelo, asegurando que todas las entradas sean de tipo float64 o int32.

4. **Reordenación de Columnas:** Se han desplazado las variables objetivo (*targets*) al final del dataset para facilitar la separación de características (*features*) y etiquetas (*labels*) durante el entrenamiento.

Resultados obtenidos:

- **Compatibilidad:** El dataset data_weather_final.csv cumple ahora con el formato requerido por la librería **Scikit-Learn**.

- **Consistencia de Tipos:** Se reporta que el 100% de las columnas de entrenamiento son numéricas. Se han eliminado las variables categóricas o se han transformado (como los meses o el fin de semana) en indicadores numéricos.

4.- Modelado

Manipular datos y sacar conclusiones

Comprendidos los datos y estando estos preparados, desarrollamos nuestros modelos de aprendizaje para generar predicciones.

4.1 Selección de técnicas de modelado:

En el apartado 2 al comprender la correlación de datos, observamos que el comportamiento del clima en Barcelona no es lineal, por lo que se hemos seleccionado modelos con random forest.

¿Porque no es lineal el clima?

- **Interacciones complejas:** La humedad por sí sola no causa lluvia. Necesitas humedad, caída de presión, una temperatura específica y que haya nubes.

- **Efectos umbral:** Puedes tener un 80% de humedad y que no llueva, pero si sube al 85%, de repente llueve torrencialmente. Esa "curva" que se rompe de golpe es la no linealidad.

- **Temperatura inquieta:** Pequeños cambios en el viento pueden cambiar totalmente el pronóstico de temperatura máxima.

Entonces si usáramos un sistema lineal como un solo árbol, lo que haría es aprenden y tirar de memoria, sacando un único resultado que puede acertar o fallar.
Mientras que al usar random forest usamos múltiples árboles que al tratar con los datos sacaran múltiples conclusiones y por mayoría ofrecerán un resultado que posiblemente tenga en cuenta la complejidad que requieren múltiples variables a tener en cuenta.

4.1.1 Técnica de modelado

Temperaturas

- Modelo: Random Forest Regressor ya que tratamos con valores numéricos float.

- Justificación: Este algoritmo es ideal para predecir variables continuas como los grados centígrados. Al promediar las predicciones con 200 árboles individuales, el modelo es capaz de capturar patrones complejos como la inercia térmica (la relación con el día anterior) y la estacionalidad anual sin caer en el sobreajuste.

- Aplicación en programa: Implementado en train_model_temp_max.py y train_model_temp_min.py.

Precipitación

- Modelos:
    - Support Vector Machine (SVM):
        - Justificación: Seleccionamos SVM debido a su buen rendimiento en ámbitos de mucha dimensión y su habilidad para hallar el "hiperplano" idóneo de corte. Permite capturar relaciones climáticas complejas y no lineales (*gracias al uso de kernels*) que una regresión logística simple podría pasar por alto.
    - Random Forest Classifier porque convertimos el evento de lluvia en binario en si llueve o no llueve.
        - Justificación: El Random Forest destaca aquí por su capacidad para manejar el desequilibrio de clases ya que hay más días de sol que de lluvia en Barcelona, y por su eficiencia al evaluar la importancia de variables como la nubosidad y la humedad relativa.
    - Logistic Regression porque es un método de clasificación como paso lo mismo con el random forest, porque la variable de lluvia lo convertimos en binario.
        - Justificación: lo usamos para comparar y validar que modelo ofrecía mejor resultado siendo Random Forest el realmente aporta una mejora significativa frente a un modelo lineal simple.

- Uso: Implementado en train_model_precipitation.py para la variable bin_prep.

4.1.2 Suposiciones al modelar:

Acciones ejecutadas:

- Se establece el random_state=42 en todos los scripts de entrenamiento para asegurar la reproducibilidad de los experimentos.
- Se asume que el histórico desde 2010 es representativo del clima actual.

Resultados obtenidos:

Los modelos están configurados para recibir de entrada tanto datos físicos directos (presión, humedad) como datos derivados (lags térmicos), lo que permite maximizar la precisión en predicciones a corto plazo.

4.2 Generar diseño de prueba

Antes de entrenar, definimos cómo evaluaremos nuestros modelos para evitar que simplemente memoricen el pasado.

Acciones ejecutadas:

- Split Cronológico:
    - Dividimos en 80% entrenamiento y 20% test.
    - En lugar de un barajado aleatorio puro, se respeta el orden temporal: el modelo aprende de los años anteriores y se examina con los datos más recientes.

- StandardScaler: lo usamos en nuestros modelos para asegura que variables con escalas muy distintas (ej. Presión en hPa vs. Temperatura en ºC) tengan el mismo peso visual para el algoritmo.

- Estrategia de optimización:
    - Para garantizar que los modelos alcancen su máximo rendimiento, se ha implementado **GridSearchCV:**
        - Esta metodología posibilita revisar sin intervención todas las variantes de parámetros finos, eligiendo la cual incrementa el F1-Score. De esta forma, eludiríamos la modificación manual y garantizamos que la disposición de cada sistema (tal como el nivel del Random Forest o la penalización del SVM) resulte la más firme para la información meteorológica.

    - Para abordar el fuerte desequilibrio de clases (muchos más días de sol que de lluvia), se aplicó la técnica **SMOTE:**
        - Esta técnica produce datos artificiales de la categoría escasa que en este caso sería de lluvia en los datos de práctica, facilitando que el algoritmo aprenda patrones reales de precipitación en lugar de simplemente memorizarlo. Adicionalmente, se puso en marcha un Pipeline que garantiza que la dimensión de información y la equidad se lleven a cabo apropiadamente en cada sección de la comprobación repetida.

Resultados obtenidos:

- Temperatura Maxima y Mínima: El éxito se mide mediante el MAE (Error Absoluto Medio) y el RMSE **(Error Cuadrático Medio)**. Nuestro objetivo es que la desviación media sea inferior a 2°C.

- Lluvia: Se prioriza el F1-Score, buscando un equilibrio entre la Precisión (no dar falsas alarmas de lluvia) y el Recall (no omitir días que realmente van a ser lluviosos).

4.3 Configuración de parámetros de los modelos construidos

Hemos creado 3 modelos predictivos, para precipitación y temperatura tanto máxima como mínima. Se hemos estandarizado el uso de Random Forest por su alta capacidad de generalización.

4.3.1 Configuración de parámetros:

Para asegurar la estabilidad y precisión de las predicciones, se han definido los siguientes hiperparámetros en el entorno de entrenamiento:

- n_estimators : Se han configurado 200 árboles de decisión para cada modelo de temperatura y 500 árboles de decisión para el modelo de precipitación… Esto permite reducir la varianza del error sin aumentar drásticamente el coste computacional.
- random_state (42): Se utiliza una semilla de aleatoriedad fija para garantizar que los resultados sean replicables en futuras pruebas.
- StandardScaler: Esto normaliza las características para que tengan media 0 y desviación 1, evitando que variables con rangos grandes (como la presión en hPa) eclipsen a las pequeñas (como la temperatura).

4.3.2 Bitácora de Experimentación

Precipitación:

Para la predicción como imputamos una nueva columna binarizando la lluvia, estableciendo si llueve o si no.
Sabiendo que nuestro **label** tiene datos de tipo binario, los modelos que hemos aprendido a lo largo del curso serian:

- ***LogisticRegression***
- ***RandomForestClassifier***
- ***SupportVectorMachine***.

Inicialmente para cada modelo modificábamos el umbral manualmente para poder obtener un resultado acorde a lo que queríamos, al no obtener un resultado deseado, se nos recomendó usar ***Smote*** y el ***GridSearchCV.***
Al usarlos en los modelos, aseguramos que los modelos aprendan a predecir la lluvia con la misma importancia que los días que no hay lluvia.
Si mantienes el umbral manual, el modelo sigue siendo débil, pero para los algoritmos buscamos el mejor equilibrio entre la detección de lluvia y la precisión de la predicción.

Para todos los experimentos se utilizó un **Split** (***80% entrenamiento, 20% test***) sin barajar (***shuffle=False***), respetando la naturaleza cronológica de los datos meteorológicos, una validación cruzada de 5 pliegues (***cv=5***) y pedimos que nos busque (***scoring='f1'***) el que mejor equilibrio tenga entre detectar la lluvia (*Recall*) y no dar falsas alarmas (*Precision*).

1. **Regresión Logística:** Optamos por aplicar la Regresión Logística porque nuestros resultados son del tipo clasificación binaria (llueve o no llueve). Este modelo calcula la posibilidad de que suceda el suceso tomando en cuenta las variables independientes (presión, humedad, etc.) con una función para garantizar que las salidas queden siempre entre cero y uno. Lo aplicamos como nuestro punto de partida gracias a su simpleza, velocidad y fácil entendimiento, permitiéndonos confirmar si otras técnicas más elaboradas ofrecen un avance relevante.

    1. **Parámetros configurados**:

        1. **StandardScaler:** Es obligatorio aquí. La regresión logística es sensible a la escala de las variables (como la presión vs el mes). El escalado asegura que todas las variables tengan el mismo peso inicial.
        2. **lr__C [0.01, 0.1, 1, 10]:** Es el parámetro de regularización. Controla el equilibrio entre ajustar bien los datos de entrenamiento y no "memorizarlos" (*evitar overfitting*). Valores más pequeños (*0.01*) aplican una regularización más fuerte.
        3. **lr__solver: 'lbfgs':** El algoritmo encargado de encontrar los coeficientes óptimos. Es el estándar para datasets de tamaño medio y funciona muy bien con variables continuas.

2. **Support Vector Machine (SVM):** Resulta ideal para hallar límites de decisión intrincados en espacios con muchas dimensiones. Lo seleccionamos debido a que la dinámica del tiempo casi nunca es recta; frecuentemente se necesita un "límite curvado" para distinguir de manera exacta los estados climáticos. A modo de ilustración, la relación entre el descenso de presión y la subida brusca de humedad no se logra atrapar con una línea recta, y SVM nos posibilita diseñar ese contorno geométrico más adecuado.

    1. **Parámetros configurados:**
        1. **svc__kernel**: **'rbf'**: Utilizamos el kernel radial para permitir que el modelo cree fronteras elípticas o circulares. Esto es vital para capturar patrones climáticos que no son proporcionales sino cíclicos.
        2. **svc__C [0.1, 1, 10]**: Controla el margen de error. Un C alto (10) intenta clasificar todos los puntos de entrenamiento correctamente, mientras que un C bajo (0.1) busca una frontera más suave y generalizable.
        3. **svc__gamma**: **'scale'**: Ajusta automáticamente la influencia de los puntos de datos individuales según la varianza, mejorando la estabilidad del modelo frente a cambios bruscos de temperatura o humedad.

3. **Random Forest Classifier:** Este es nuestro método más confiable y el que utilizamos. Al tratarse de un "ensamble" de 500 árboles de decisión, es muy resistente a las interferencias y a los datos extraños (*outliers*) habituales en los aparatos de medición del tiempo. Su gran beneficio reside en que atrapa relaciones no lineales complejas de modo natural; por ejemplo, identifica que, si la presión baja mucho y la humedad rebasa el 80%, la posibilidad de precipitación se dispara de manera desigual.

    1. **Parámetros configurados:**
        1. **rf__n_estimators: 500:** El número de árboles. Hemos puesto 500 para asegurar que el modelo sea muy estable y que la predicción final sea el promedio de muchas "opiniones" diferentes dentro del bosque.
        2. **rf__max_depth: 8:** Limitamos la profundidad de los árboles para evitar el overfitting. Si los árboles son demasiado profundos, memorizan días específicos del pasado en lugar de aprender patrones generales del clima.
        3. **rf__min_samples_leaf: 10:** Obliga a que cada "hoja" final del árbol tenga al menos 10 días de ejemplo. Esto suaviza el modelo y evita que tome decisiones basadas en casos aislados o errores de sensor.

Selección e Ingeniería de Variables (Features):

Para maximizar la capacidad predictiva de los modelos, se ha definido un conjunto de 15 variables de entrada (***features***). Además de las variables directas del sensor (*temperatura, humedad, nubosidad*), se han desarrollado variables calculadas específicas:

- **Variables de Tendencia:** Se implementó **pressure_delta**, que calcula la variación de la presión atmosférica en las últimas 24 horas, permitiendo al modelo detectar frentes de baja presión y **pressure_yesterday** es una nueva variable que representa la presión atmosférica media del día anterior, al crear esta variable, le estamos dando el valor de ayer al algoritmo, entonces no se puede calcular la tendencia (*si la presión sube o baja*) si no conocemos el punto de partida del día anterior. Esta variable es el componente necesario para generar pressure_delta.
- **Variables de Persistencia:** Se incluyó **rain_yesterday_bin** para que el modelo considere el estado climático previo.

Clasificación de resultados de experimentos

Precipitación:

Tras ejecutar los modelos hemos optimizado la métrica **F1-Score** (para no penalizar excesivamente ni los falsos positivos ni los falsos negativos), la clasificación de los modelos es la siguiente:

Modelo	Recall	Precisión	Accuracy	F1-Score
Random Forest	**0.7224**	**0.6020**	**0.8123**	**0.6567**
SVM	**0.8030**	**0.5193**	**0.7663**	**0.6307**
Logistic Regression	**0.8119**	**0.4883**	**0.7418**	**0.6099**

Temperatura:

Para la predicción de la temperatura tanto máxima como mínima aplicamos RandomForestRegressor, que desde el principio ofreció resultados que cumplían con el objetivo del negocio, entonces buscamos formas de mejorar el resultado del entreno.

Evaluamos diferentes estrategias de validación y preprocesamiento aplicando:

- Experimento 1:
  - train_test_split, con shuffle en False

- Experimento 2:
  - División conologica
  - Pipeline
  - standardScaler

temperatura	Experimento	MAE	RMSE	R^2
mínima	**1**	**0.89 ºC**	**1.11 ºC**	**0.96**
máxima	**1**	**1.13 ºC**	**1.47 ºC**	**0.93**
mínima	**2**	**1.07 ºC**	**1.40 ºC**	**0.93**
máxima	**2**	**1.18 ºC**	**1.57 ºC**	**0.92**

4.3.3 Modelos:

Acciones ejecutadas:

Se ha seleccionado un conjunto de características (*features*) específica para cada objetivo meteorológico en base a observar correlaciones y múltiples pruebas realizadas:

1. Modelo de Temperatura Máxima → Features:
    - temperatura máxima / mínima de ayer
    - día del año y mes
    - punto de roció medio y máximo
    - nubosidad media
    - velocidad media viento
    - humedad máxima.

2. Modelo de Temperatura Mínima → Features:
    - temperatura máxima / mínima de ayer
    - día del año y mes
    - punto de roció mínimo y medio
    - nubosidad mínima y media

3. Modelo de Precipitación → Features:
    - precipitación lag1
    - lluvia ayer binaria
    - presión delta y presión superficial media en Hectopascales
    - Nubosidad media y máxima
    - humedad media y máxima
    - punto de rocío medio
    - día del año y mes
    - estaciones: invierno, verano, otoñó y primavera.

Resultados obtenidos:

- Estructura: Los modelos (.joblib) se guardan en la ruta src/models/ tras completar el entrenamiento.

- Eficiencia: El uso de n_jobs=-1 en los scripts permite que el entrenamiento utilice todos los núcleos del procesador disponibles, optimizando el tiempo de construcción a pocos segundos a pesar de procesar miles de registros históricos.

- Resultado: Disponemos de tres modelos especializados que trabajan de forma conjunta, los de temperatura para el confort térmico y el de precipitación como un clasificador de probabilidad de eventos de lluvia.

4.4 Evaluar el modelo:

Hemos evaluado los modelos entrenados utilizando el conjunto de datos de prueba, que representa el 20% de los datos que el modelo nunca ha visto. El objetivo es comprobar si las predicciones son lo suficientemente precisas para cumplir con los objetivos de negocio.

4.4.1 Evaluación del modelo:

Acciones ejecutadas:

- Creamos un programa “evaluation.py“ para calcular los errores en los modelos de regresión de temperatura máxima y mínima y métricas de clasificación para el modelo de Precipitación.
- Se han comparado los resultados del **Random Forest** frente a otros modelos para asegurar el modelo con mayor precisión.

Resultados obtenidos:

1. **Modelos de Temperatura (Regresión):**
    - **MAE (Error Absoluto Medio):** Obtuvimos en temperatura máxima 1.13 ºC y en temperatura mínima 0.89 ºC. Esto significa que, de media, el modelo solo se equivoca por menos de dos grados, lo cual es excelente para la planificación diaria del usuario.
    - **RMSE(Error cubico):** Obtuvimos en temperatura máxima 1.47 ºC y en temperatura mínima 1.11 ºC. Esto significa que, en temperaturas extremas, ejemplo 4ºC o 32ºC, el modelo se equivoca por menos de dos grados, lo cual es excelente para la planificación diaria del usuario.
    - **R^2 Score:** Ambos modelos presentan un coeficiente de determinación superior a **0.90**, siendo temperatura maxima de 93% y minima del 96%, lo que indica que el modelo explica el 90% de la variabilidad de la temperatura en Barcelona.
2. **Modelo de Precipitación (Clasificación):**
    - **Precisión:** Alta capacidad para evitar "falsas alarmas" de lluvia.
    - **Recall (Sensibilidad):** Capacidad equilibrada para detectar días de lluvia real.
    - **F1-Score:** Al utilizar un **Random Forest Classifier**, el F1-Score supera significativamente al de la Regresión Logística y SVM, confirmando que la técnica de random forest gestiona mejor la probabilidad de lluvia.

4.4.2 Configuración de parámetros revisados

Tras las primeras pruebas de evaluación, se realizaron los siguientes ajustes para mejorar el rendimiento:

- **Inclusión de Lags:** Se confirmó que sin la variable tmax_yesterday, el error MAE subía por encima de los 3°C. incluirla fue clave para la precisión actual.

- **Uso del Smote:** Se observó que el modelo de precipitación requería un equilibrio de datos ya que las el porcentaje de precipitaciones en Barcelona es del 18%, y hay un desbalance entre días de lluvia y días que llueve.

- **Balanceo de Clases:** En el modelo de lluvia, se ajustó el umbral de decisión para priorizar el aviso de lluvia sobre el de sol, dado que el coste de "mojarse" es mayor para el usuario que el de cargar con un paraguas innecesariamente.

5.- Evaluación

Evaluación del modelo y conclusión

Evaluamos nuestros modelos con el objetivo de asegurar que el modelo es válido antes de su despliegue definitivo, alineando los resultados técnicos con las necesidades de los alumnos de STUCOM.

5.1 Evaluar resultados

Alinear la evaluación de los resultados de la minería de datos con los criterios de éxito empresarial

Acciones ejecutadas:

- Se ha contrastado las predicciones generadas por los modelos de Random Forest contra el dataset de prueba 20% de los datos históricos reservados.
- Se ha perseguido los criterios de éxito definidos en la Fase 1 (MAE y RMSE < 2°C, **R^2** > 85%)

Resultados obtenidos:

- Objetivo cumplido: El éxito del proyecto se definió originalmente como la capacidad de predecir si el usuario debe abrigarse o llevar paraguas, en temperaturas con un error menor a 2°C en MAE y RMSE y **R^2** superior al 85%, precipitación con una **precisión** del *>60%*, un **Recall** de *>70%* y un **f1-score** *>60%.*
    - En modelos de temperatura actuales se han utilizado el train_test_split con el shuffle en False ya que arroja mejor resultado.
      Presentan un MAE, RMSE y **R^2** que superan las expectativas iniciales de éxito.
        - Temperatura máxima:
            - MAE = 1,13 ºC
            - RMSE = 1,47 ºC
            - **R^2** = 93%
        - Temperatura mínima:
            - MAE = 0,89 ºC
            - RMSE = 1,11 ºC
            - **R^2** = 96%

    - Precipitación: buscamos la robustez del Clasificador de Lluvia de modo que no solo nos hemos fijado en la precisión global, sino en el **F1-Score**. Hemos logrado que el modelo priorice el "Recall" en días de lluvia: preferimos que el sistema nos avise de una lluvia ligera que no ocurrirá, a que nos asegure sol y acabemos mojados (evitar los máximos FN).
        - El modelo que hemos seleccionado al final es el de **Random Forest** debido a su consistencia y robustez. A diferencia de los modelos **SVM**

y **Logístico**, que mostraron un *Recall* muy alto pero una *Precisión, F1-Score* y *Accuracy* muy baja, el Random Forest logró:

- Logró una ***Precisión*** del **60.2%,** lo que significa que la mayoría de sus avisos de lluvia son correctos.
- Un ***Accuracy*** superior al **81%** demuestra que el modelo entiende tanto los días de lluvia como los días despejados.
- Un ***Recall*** del **72.2%** es suficiente para una aplicación de usuario final donde se prefiere un equilibrio que no genere alertas innecesarias constantemente.
- La introducción de la feature **pressure_delta** resultó ser determinante para mejorar la estabilidad del modelo
- **SVM:** Presentó un **Recall** alto pero demasiados falsos positivos.
- **Regresión Logística:** Funcionó cual referencia, siendo sobrepasada por los modelos de ensamble en capturar las relaciones no lineales entre presión y humedad.

- Aportación de los Lags: La evaluación confirmó que la meteorología en Barcelona tiene una "memoria" alta. Sin los datos del día anterior (Lags), el modelo perdía un 40% de precisión, lo que valida su inclusión.

- Utilidad Práctica: El sistema no le da simplemente un número al usuario, sino que, mediante la lógica implementada en el dashboard, traduce esos datos en consejos. Esto garantiza que el usuario final reciba una respuesta directa a su necesidad sin tener que interpretar gráficos complejos.

- Fiabilidad: La comparación entre el modelo de Random Forest y los modelos base confirmó que la IA seleccionada es capaz de capturar la complejidad climática de Barcelona de forma muy superior a una simple media estadística.

Conclusión: Los resultados son consistentes y el modelo está listo para su uso real

5.2 Revisión del proceso

Acciones ejecutadas:

- Se ha auditado el flujo de datos desde la captura en AEMET hasta su preprocesamiento con oneweather.
- Se ha revisado la importancia de las variables (*Feature Importance*) para confirmar que el modelo no toma decisiones basadas en variables irrelevantes.

Resultados obtenidos:

Fase de Datos:

- El proceso de limpieza e imputación fue crítico. Se detectó que el uso de estaciones meteorológicas cercanas (Puerto Olímpico y Museo Marítimo) fue una decisión acertada para minimizar la varianza local.

- Calidad del Dato: Al principio sufrimos con los nulos de AEMET. La decisión de cruzar los datos con OpenWeather no fue solo para rellenar huecos, sino que actuó como un sistema de doble verificación que ha hecho al modelo más fiable.

Fase de Modelado:

- Selección de Algoritmos: Se evaluaron modelos lineales simples, pero se descartaron rápidamente. La revisión del proceso muestra que solo los modelos de Random Forest eran capaces de entender que la humedad afecta de forma distinta a la lluvia dependiendo de si la presión está subiendo o bajando.

- La inclusión de los *Lags* (datos del día anterior) fue el factor determinante en la precisión. Se revisó el proceso y se confirmó que el *Pipeline* con *StandardScaler* garantiza que los modelos sean estables incluso si hay cambios bruscos de presión atmosférica.

Factor Humano:

Una parte crítica de la revisión fue entender que un gráfico de temperatura no ayuda a un estudiante con prisa. Por ello, se decidió que el "resultado" final del modelo debía ser el **Consejo Predictivo** (Abrigo/Paraguas) y no solo un número.

Calidad metodológica: Se confirma que se han seguido todos los pasos de la metodología CRISP-DM sin saltar etapas críticas de validación.

5.3 Siguientes pasos a determinar

Lista de posibles acciones y decisiones:

Acciones ejecutadas:

- Evaluación del coste de mantenimiento frente al beneficio de uso en el centro STUCOM.
- Análisis de la viabilidad del despliegue en Streamlit Cloud.
- Que nuestro modelo de precipitación sea lo mas cercano a la realidad. (ideal 100%)
- Integración de nuevos conceptos y consejos en el frontend, controlando más sucesos meteorológicos.
- Actualización de la base de datos diariamente.

Resultados obtenidos:

Si se valida el modelo para su uso, se establecen las siguientes líneas de actuación:

- Paso 1: Despliegue Inmediato: Se aprueba la versión actual para su publicación en Streamlit, ya que cumple con todos los requisitos de estabilidad y precisión.

- Paso 2: Feedback de Usuario: Se propone monitorizar las predicciones durante el primer mes de uso real en la escuela para ajustar los umbrales de los consejos.

    - ejemplo: definir si 10°C es "frío" o "fresco" según la sensación térmica real de los alumnos.

- Paso 3: Programar una revisión de los modelos dentro de seis meses para adaptarlos a la variabilidad estacional del próximo año.

- Paso 4: Si se plantea su uso durante años se plantea la escalabilidad en obtención de datos, es decir, el modelo podría alimentarse de una API de IoT instalada directamente en el centro educativo, eliminando la dependencia de estaciones externas y logrando una precisión hiper-local.

- Paso 5: A raíz de todas las mejoras se añaden más conceptos a predecir y con ello nuevos consejos.

6.- Despliegue

Aplicación de las conclusiones a los negocios

Una vez evaluados los modelos y la aplicación, pasamos a compartir el proyecto con el mundo. Sin embargo, el despliegue no consiste únicamente en liberar el código; implica garantizar que la herramienta sea accesible, sostenible y útil para el usuario final.

6.1 Plan de implementación

Tras evaluar los resultados del modelo, el equipo decidió que el despliegue de la primera versión (0.1.0) de la herramienta SWF (Stucom Weather Forecast) se ejecutase en un entorno local, permitiendo que el usuario final utilice su propia infraestructura para procesar los datos y visualizar las predicciones sin depender de servidores externos.

El plan implementado para su despliegue se llevará a cabo siguiendo estos puntos:

6.1.1.- Distribución y disponibilidad:

La herramienta se distribuye a través de un repositorio en GitHub, lo que permite un control de versiones riguroso. Esto asegura que el usuario final siempre tenga acceso a la versión más estable del software y que los desarrolladores puedan desplegar parches de seguridad o actualizaciones de los modelos predictivos de forma ágil.

- Repositorio en GitHub: *https://github.com/ aaxel242/Weather_Forecasting.git*

6.1.2.- Estandarización del Entorno de Ejecución:

Para evitar el conflicto de dependencias (el problema de "en mi ordenador no funciona"), el despliegue se basa en el gestor de paquetes **uv**. Este sistema garantiza la reproducibilidad total del entorno de minería de datos, instalando automáticamente las librerías específicas (Pandas, Scikit-Learn, Streamlit) con las versiones exactas utilizadas durante la fase de modelado.

6.1.3.- Entrega de modelo entrenado:

Modelo de Entrega: La solución se entrega como un paquete de software ejecutable en entorno local. El núcleo de la aplicación reside en archivos .joblib pre-entrenados, lo que evita la necesidad de computación intensiva en el momento del uso.

6.1.4.- Interfaz de Integración:

Se ha seleccionado **Streamlit** como el marco de trabajo para el despliegue, ya que permite transformar los scripts de minería de datos en una aplicación web interactiva de fácil acceso para perfiles no técnicos (alumnos y administración).

6.1.5.- Facilitación del Despliegue:

Como parte del plan de despliegue para el usuario final, se incluye un script de procesamiento por lotes (.bat). Este actúa como el puente de ejecución que prepara el entorno virtual sin intervención manual del usuario, asegurando que el despliegue sea exitoso en cualquier equipo con Windows y Python instalado.

6.1.6.- Verificación del Despliegue:

El éxito del despliegue se confirmará mediante la visualización correcta de las tarjetas dinámicas de predicción y la accesibilidad a las pestañas de análisis histórico.

6.2 Plan de monitorización y mantenimiento

Para asegurar que las predicciones sigan siendo precisas frente a los cambios climáticos y evitar “Model Drift” (degradación del modelo), se ha de seguir un ciclo de mantenimiento:

Monitorización:

1. **Operativa:**

    - El usuario verifica visualmente el funcionamiento diario del dashboard

    - **Indicadores**: Verificación visual de la carga del dashboard y ausencia de errores en la consola de streamlit al generar predicciones.

1. **Analítica:**

    - Se evalúan trimestralmente las métricas de los modelos. Si por ejemplo el error de temperatura supera los 2.5ºC o la precisión en el de lluvia baja considerablemente, se activa el reentrenamiento

Protocolo de Actualización:

El ciclo de vida del mantenimiento consta de cuatro fases que deben ejecutarse secuencialmente para actualizar la inteligencia del sistema:

1. **Ingesta:** Descarga manual de nuevos datos meteorológicos en src/data/raw

2. **Procesamiento:** Ejecución de scripts para realizar la limpieza de los nuevos datos y prepararlos para luego añadirlos al csv de entrenamiento como:

    - Compilar.py → Este ejecuta por su cuenta unir_json.py que unirá en un solo csv llamado “datos_clima_definitivo.csv” todos documentos en la carpeta raw.
    - Limpieza.py
    - Imputar_datos.py
    - add_lags.py

3. **Reentrenamiento:** Ejecución de los modelos de entrenamiento (tmax, tmin y precipitación) para generar nuevos archivos .joblib.

4. **Despliegue:** Actualización y reinicio de la aplicación para cargar los modelos actualizados automáticamente.

Frecuencia:

- Actualización de base de datos: **Mensual**

- Reentrenamiento de modelos: **Semestral** (o bajo alerta de monitorización)

6.3 Informe final

Este informe consolida los resultados obtenidos en el proyecto SWF, validando el cumplimiento de los objetivos de negocio y de minería de datos planteados inicialmente.

6.3.1.- Resumen ejecutivo:

Se ha desarrollado exitosamente un sistema de predicción meteorológica para ir a la escuela STUCOM pelai, al integra datos históricos de dos estaciones clave :Museo Marítimo y Puerto Olímpico, capaz de procesar 15 años de datos meteorológicos históricos para generar pronósticos a 7 días.
La solución final es una aplicación de escritorio visual que no ofrece solo números sino que traduce la complejidad climática en recomendaciones prácticas (vestimenta, paraguas, hidratación) basadas en el pronóstico para el personal y alumnado de Stucom pelai.

6.3.2.- Evaluación de Objetivos frente a Resultados:

1.- Objetivo de Temperatura máxima y mínima:

Margen de Error:

- **Temperatura Mínima:**
  - El error en temperaturas intermedias es de 0,89 ºC (MAE)
  - El error en temperaturas extremas es de 1.1122 ºC (RMSE)
- **Temperatura Máxima:**
  - El error en temperaturas intermedias es de 1,13 ºC (MAE)
  - El error en temperaturas extremas es de 1.4719 ºC (RMSE)

La desviación es apenas perceptible para la sensación térmica humana en la mayoría de los casos, esto permite una planificación de vestimenta (abrigo) con una fiabilidad excelente.

Fiabilidad:

- Temperatura mínima (R^2): 96%
- Temperatura máxima (R^2): 93%

Los modelos explican el 93-96% (R^2) de la variabilidad térmica, lo que indica un ajuste excelente a los patrones locales.

Objetivos del negocio en temperatura cumplido

Al tener un error inferior a los 2ºC en temperaturas intermedia y extremas cumplimos el umbral de confianza en el criterio de éxito empresarial, y al tener también un coeficiente R^2 superior al 85% cumplimos el el criterio de éxito de minería de datos.

2.- Predicción de Lluvia:

Acierto Global (Accuracy): 81.23%. El modelo clasifica correctamente 8 de cada 10 días, una cifra sólida considerando la variabilidad del clima local.

Capacidad de Detección (Recall): 72.24%. De todos los días que realmente llovió, el sistema fue capaz de identificar y avisar al usuario en el 72% de los casos.

Precisión (Precision): 60.20%. Cuando la aplicación muestra el icono de lluvia, tiene un 60% de probabilidad de acierto. Se ha calibrado el modelo para minimizar los Falsos Negativos (días de lluvia no detectados), aceptando un ratio de falsos positivos del 40% en favor de la seguridad del usuario. Se prefiere ser precavidos y fallar a que el usuario se moje.

F1-Score: 0.6567. el sistema ofrece un equilibrio operativo útil para la vida cotidiana, destacando especialmente en la reducción de la incertidumbre en días inestables.

Objetivos del negocio de precipitación cumplido

- Ya que la **precisión** a superado el 60% del umbral de confianza en el criterio de éxito empresarial.
- Se cumple el criterio de éxito de minería de datos al obtener un **F1-Scores** superior al 0.6 y un **Recall superior al 70%**. Cumpliendo con la filosofía de negocio: "es preferible avisar de una lluvia que no ocurre (falsa alarma) a que un usuario se moje por falta de aviso.

6.3.3.- Entregables del Proyecto:

- **Código fuente:** Código fuente completo y documentado en GitHub.
- **Modelos Entrenados:** Archivos serializados (.joblib) listos para ser procesados:
  - modelo_tmax.joblib
  - modelo_tmin.joblib
  - modelo_lluvia.joblib

- **Interfaz de Usuario:** Dashboard interactivo basado en Streamlit que permite la visualización dinámica y intuitiva de datos y la generación de los pronósticos en tiempo real acompañada de consejos.

- **Documentación Técnica:**
  - Manual de Instalación y despliegue del sistema
  - Manual de Usuario.

6.3.4.- Conclusión y Recomendaciones

Conclusiones Clave:

- **Filosofía "Más vale prevenir":** El modelo de lluvia es intencionalmente conservador. Prioriza avisar siempre que hay riesgo (Recall 72%), asumiendo algunas falsas alarmas (Precisión 60%) para que el usuario nunca salga desprevenido.
- **Horizonte de Fiabilidad:** La predicción es muy precisa para las primeras 48-72 horas. A partir del 4º día, el margen de error aumenta debido a la naturaleza recursiva del sistema.
- **Ubicación Geográfica especifica:** El sistema está especializado en la zona del centro de Barcelona, ya que se ha hecho para ir a la escuela STUCOM pelai.
- **Viabilidad Económica:** Al ser una solución que se ejecuta en local y utiliza software de código abierto (Python/uv), el coste de mantenimiento para el centro es nulo, cumpliendo con el análisis de costes inicial

6.4 Revisión del proyecto:

6.4.1.- Experiencia:

Durante el desarrollo enfrentamos varios desafios:
1.- Obtención de datos: Fue difícil encontrar fuentes de datos gratuitas, y al observar carencias en una para desarrollar el proyecto, se busco una fuente de datos secundaria para complementar.
2.- Aprendizaje en la fusión, limpieza e imputación de datos, al tener tantos datos separados por la limitaciones de la fuente, luego fallos de estaciones en dias puntuales, además de la generación de nuevos datos a partir de los disponibles, aplicando lógica, como binarizar la lluvia o añadir lags.
3.- Modelos, en especial el de precipitación, fue el mayor desafío técnico debido a los datos desbalanceados en la variable de lluvia donde los días de sol superan ampliamente a los de lluvia en Barcelona, el 18% días de lluvia.
Los consejos y experimentación demostró que el uso de técnicas como el ajuste de pesos en los modelos y la validación mediante el F1-Score (en lugar de solo Accuracy) fue vital para que la herramienta fuera útil en la vida real.
4.- Descubrir formas de guardar los entrenos con pkl y joblib, para evitar que cada vez que se ejecuta la aplicación se entrene y haya que esperar minutos para obtener un resultado.
5.- Frontend dinámico, fue divertido buscar formas de presentar el sistema y representar los datos con sus consejos.
6.- Integración de la fase de despliegue mediante un ejecutable (.bat) permitió cerrar la brecha entre el desarrollo técnico y la usabilidad para el usuario final del centro Stucom.

6.4.2.- Evaluación del Éxito:

El proyecto ha cumplido su objetivo principal: dotar al usuario de una estación meteorológica virtual personalizada para la zona marítima de Barcelona.

1. **Independencia y Privacidad:** A diferencia de las otras aplicaciones comerciales, SWF (Stucom Weather Forecast) funciona 100% en local. La implementación SWF.bat (UV requerido) ha permitido encapsular todos los procesos necesarios para la ejecución de la aplicación en una experiencia de "un solo clic" mucho más sencilla y accesible.

2. **Ingeniería de Datos:** Se logró unificar con éxito varias fuentes de datos de distintas estaciones meteorológicas desde 2010 hasta gran parte de 2025 en un csv tratado y preparado para entrenar los modelos adecuadamente.

6.4.3.- Lecciones y limitaciones:

1. **Variables Estáticas:** Al no disponer de modelos independientes para predecir variables secundarias, como la presión atmosférica futura, dato que puede proporcionar un satélite, por ejemplo, el motor asume que estas se mantienen estáticas. Esto impide detectar cambios bruscos de tiempo, como tormentas repentinas u olas de frio polar que vienen del norte u olas de calor que vienen de sur, para los cuales se necesitarían datos atmosféricos en tiempo real para predecirlos.

2. **Formato de Datos:** Al trabajar con distintas fuentes de datos nos tuvimos que enfrentar a cambios de formato de datos entre los distintos csv.

Siguientes pasos:

1. **Automatización de Datos:** Eliminar la dependencia de descargar los datos de forma manual y poder tratar con datos en tiempo real directamente con API.

2. **Publicación de la app:** Poder desplegar la aplicación de forma publica, para que sea accesible desde internet sin necesidad de descargar la aplicación de forma local, lo cual permitiría mayor accesibilidad y eficiencia a la hora de querer ver el pronóstico de la zona de Stucom.

Bibliografía

Documentos de ayuda:

- PDF de clase con definiciones, propósitos de aplicación.
- Notebook 10 (trabajo de clase con entrega)
- Notebook 11 (trabajo de clase con entrega)
- Metodología CRISP-DM
- ModelerCRISPDM de IBM → Para apartado 6 Despliegue, ya que resultaba confuso al haber hecho un manual de instalación y despliegue del sistema, era raro tener que hacer lo mismo 2 veces. Ayudo a determinar lo que iba ahí.

Webs de ayuda:

- Conceptos del apartado 6 – Despliegue, confuso al haber manual de instalación y despliegue del sistema. Webs consultadas:
  - IBM → Despliegue - Documentación de IBM
  - Blog →CRISP-DM: Fase de "Despliegue" (Deployment) | Blog de Mikel Niño: Industria 4.0, Big Data Analytics, emprendimiento digital, modelos de negocio
  - Web extra → Metodología CRISP-DM - Adictos al trabajo Tutoriales

Software de soporte:

- Power BI → Usos vitales:
  - Metodo de seguridad por redundancia al comparar con los resultados y visualizaciones obtenidas con python.
  - Usado para facilitar la correlación de datos mediante el uso de diferentes gráficos.

Fuentes de datos meteorológicos buscados:

AEMET

- *Requiere cuenta API Key --> Gratis pero solo últimos 7 días + climatología resumida.*
- *Histórico completo (1920–2025): no está accesible directamente en CSV gratis; o lo montamos nosotros o comprar el pack ya consolidado por 3,5€ https://www.aemet.es/es/datos_abiertos/AEMET_OpenData https://opendata.aemet.es/centrodedescargas/inicio*
- *Enlace descarga manual: https://opendata.aemet.es/centrodedescargas/productosAEMET*
- *https://x-y.es/aemet/est-0201D-barcelona-cmt*
- *https://x-y.es/aemet/prov-barcelona#indice4*
- *https://www.aemet.es/es/serviciosclimaticos/datosclimatologicos/valoresclimatologicos?k=cat&p=08&l=0200E*

MeteoBlue

- *Es de pago, los datos de prueba gratuitos son de Basilea (Suiza) importante: Se indican tipos de gráficos que nos pueden ayudar a cómo interpretar y presentar nuestros datos: https://www.meteoblue.com/es/tiempo/historyclimate/weatherarchive/barcelona_espa%C3%B1a_3128760*

- *Se puede descargar en modo prueba, pero hay que seleccionar lo que se quiere incluir: https://www.meteoblue.com/es/user/order/historyplus*
- *https://www.meteoblue.com/es/server/queue/status?id=1934F4D9-1945-423D-90C3-1254A1402F42*
- *https://www.meteoblue.com/es/tiempo/historyclimate/climatemodelled/barcelona_espa%C3%B1a_3128760*
-

Weather Spark (NO RECOMENDADO)

- *Requiere cuenta y es de pago: https://es.weatherspark.com/download/47213/Descargar-datos-meteorol%C3%B3gicos-de-Barcelona-Espa%C3%B1a*

One Weather

- *https://oneweather.org/archive/*

Otros enlaces externos

- *https://www.meteo.cat/wpweb/climatologia/el-clima/climatologies-xema/*
- *https://www.kaggle.com/code/ahmedwaelz/weather-in-spain-preprocessing-timeseries/output*

Teoría de algunos métodos/dependencias:

- *La mayoría de la información la hemos sacado de los apuntes de clase.*
- ***SMOTE**: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html*
- ***GridSearchCV**: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html*
- ***Streamlit**: https://www.datacamp.com/tutorial/streamlit*
- ***Matplotlib Plots**: https://matplotlib.org/stable/plot_types/index.html*
- ***Support Vector Machines (SVM)**: https://scikit-learn.org/stable/modules/svm.html*
- ***Mimetypes**: https://docs.python.org/3/library/mimetypes.html*
- ***Datetime**: https://docs.python.org/es/3/library/datetime.html*
- ***Base64**: https://docs.python.org/es/3/library/base64.html*
- ***Pipeline**: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html*

Chats utilizados como apoyo:

- Copilot
  - Consultas rápidas de conceptos.
- Gemini
  - Creación de imágenes para el proyecto.
  - Consulta de dudas y conceptos complejos de la documentación.
    - Ej: Apartado 6.- Despliegue al haber manual de instalación y despliegue del sistema, se vuelve confuso, se busca en webs pero todas dicen cosas diferentes, se pregunta la diferencia y que debería contener.
  - Soporte en la programación
- Claude
  - Soporte en la programación
- ChatGPT
  - Creación de imágenes
  - Búsqueda de datos

Recomendaciones de Profesores:

- GrindSearchCV
- SVM
- Métodos para igualar datos: Empleo de **SMOTE** (Synthetic Minority Over-sampling Technique) al manejar conjuntos de datos desiguales dentro de la trata de datos.