pontIA

INFORME PONTIA WORLD

PROYECTO JÚPITER RECONOCIMIENTO DE EMOCIONES

Anamaría Turda - William Ganem - Inés Benito - Iñigo Ugidos

EQUIPO DE TRABAJO Y OBJETIVOS DEL PROYECTO

En este proyecto se analizan los datos de la empresa **PontIA World**, la empresa detrás de un parque de atracciones con el mismo nombre. Para el desarrollo de este proyecto se ha construido un equipo de analistas de datos, los cuales han desarrollado múltiples tareas durante sus 8 meses de trabajo aproximadamente.

- **Anamaría Turda**: Project Manager del proyecto, ha liderado la organización del mismo desde su inicio. Además, ha contribuido con su trabajo en el proceso de limpieza de datos con Python, la construcción e implantación del modelo relacional en SQL, el desarrollo del mejor modelo de ML conseguido y la contribución con una propuesta de IA Generativa
- **William Ganem**: Miembro activo del proyecto, ha contribuido con su trabajo en el proceso de limpieza de datos con Python, la construcción e implantación del modelo relacional en SQL y el desarrollo de varios de los mejores modelos de ML que hemos logrado.
- **Inés Benito**: Miembro activo del proyecto, ha contribuido con su trabajo en el proceso de limpieza de datos con Python, ha sido una de las principales artífices del modelo relacional en SQL, también ha desarrollado algún modelo de ML, y ha contribuido con otra propuesta de IA Generativa.
- **Iñigo Ugidos**: Miembro activo del proyecto, ha contribuido con su trabajo en el proceso de limpieza de datos con Python, la construcción e implantación del modelo relacional en SQL, apoyo y testeo con algún modelo de ML, ha contribuido con una propuesta de IA Generativa y elaborado este informe.

El **objetivo general** que persigue el proyecto es analizar los datos de PontIA World y detectar patrones en las emociones registradas de los visitantes del parque, usando esa información para realizar propuestas estratégicas con la intención de mejorar la experiencia de los visitantes y la rentabilidad del negocio.
Los **objetivos principales** a lo largo del proyecto para lograr el objetivo general han sido:

- **Objetivo 1**: Realizar un proceso ETL consistente y de calidad, utilizando para ello los diferentes archivos JSON, Python y librerías como Pandas y Numpy.

- **Objetivo 2**: Diseñar e implementar un modelo de base de datos relacional en SQL para organizar la información y facilitar la ejecución de consultas de negocio.
- **Objetivo 3**: Montar, entrenar y comparar diferentes modelos de ML y DL para clasificar las emociones y poder predecirlas, acompañándolos del desarrollo de una propuesta de IA Generativa.
- **Objetivo 4**: Generar un dashboard en Power BI para mostrar los insights obtenidos de todo el análisis de manera gráfica e interactiva.

Cabe mencionar que para la organización del trabajo conjunto del equipo se ha utilizado metodología agile SCRUM, basada en sprints y organizado vía **Trello**, herramienta que hemos utilizado para llevar el seguimiento de tareas. Además el equipo ha mostrado gran coordinación e implicación desde el primer momento, organizando un total de **12 reuniones** periódicas vía Google Meet, con el objetivo de poner en común avances y previsiones, más allá de la comunicación continua por **Discord**.

PROCESO DE LIMPIEZA DE DATOS

Para comenzar con la limpieza de datos, partimos de varios archivos en formato **JSON** y comenzamos a explorarlos y ordenarlos a través de **Google Colab** utilizando **Python**. En esta exploración inicial, logramos identificar varios problemas de calidad en los datos y tratamos de solucionarlos:

- Formato de los datos: Desglose de datos encapsulados en diccionarios y homogeneización en tablas.
- Valores negativos: Se detectaron en los campos “tiempo de espera”, “costes”, “duración”, y se transformaron en valores positivos.
- ID visitante inconsistente: El campo “id_visitante” no representa a una persona única ya que aparecían diferentes visitantes (con distinta “procedencia” y “foto”), bajo el mismo ID. Se descartó su uso como identificador principal y se creó un nuevo campo llamado “id_visitante” a partir de “procedencia”, “duración” y “día de visita”.
- Dos imágenes faltantes: En los archivos “emociones” y “valoraciones”. Se crearon nuevos valores según duplicados existentes en otros archivos para mantener la consistencia de filas, añadiendo a estas imágenes la terminación “doble.jpg”.
- Valores extremos en la duración: Detectamos visitas superiores a 9 horas (540 min), lo que transgrede las reglas del negocio. Se propuso imputar estos valores, tras el análisis, con la duración media de 350 min para no perder así datos (etiquetando, estudiando el error y concluyendo que no se cumple la regla de negocio asociada)
- Valores atípicos en costes: Algunos tickets con valores muy bajos (0, 1, 2 o 3 euros) sin una lógica. Se consideró revisar por tipo de entrada y analizar si deben excluirse o considerarse como entradas gratuitas/promocionales.
- Errores en el Fast Pass: Este tipo de entradas no cumplen su regla de <3/día. Se etiquetaron las entradas como erróneas para posteriores conclusiones.

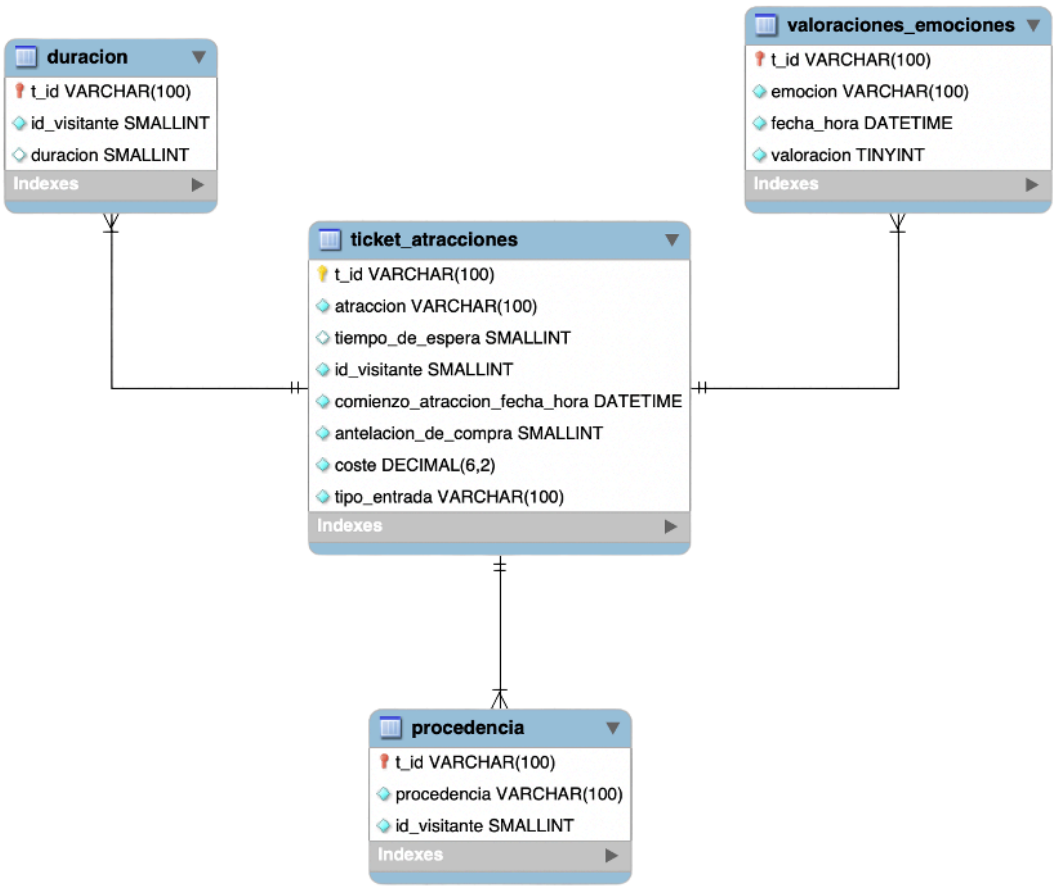
Además de la detección de estos errores, se realizó un análisis exploratorio y diferentes visualizaciones **EDA**:

- Mapamundi con la distribución geográfica de visitantes por país de procedencia.
- Gráfico de número de visitantes por atracción
- Gráfico de porcentajes de distribución de las entradas.
- Datos de las duraciones medias distribuidas.

CONSTRUCCIÓN DE BASE DE DATOS RELACIONAL Y MODELO SQL

Tras finalizar con la limpieza preliminar de los datos y creación de tablas combinadas, se diseñó un modelo relacional utilizando **MySQL**, con la intención de conectar las fuentes de datos y las **4 tablas principales**. El objetivo perseguido ha sido estructurar los datos para facilitar **consultas analíticas** y **extracción de información.**

Las tablas principales son la de “ticket_atracciones”, que es la central y tiene relación con las otras tres “duracion”, “valoraciones_emociones” y “procedencia”, como se puede observar en el siguiente esquema.



Este establece la base de datos de PontIA World con un almacenamiento eficiente y un acceso rápido a estos mediante **consultas SQL**, facilitando cálculos como; la media diaria de visitantes, la cuantía total de visitas, los días con más visitas, las

atracciones más visitadas, la emoción más frecuente por atracción, la media de valoraciones por atracción o los tiempos de espera por atracción, entre otros KPIs clave para el negocio.

En cuanto a esas consultas, hemos realizado tanto las requeridas por el proyecto inicialmente, como algunas otras que consideramos podrían ser de gran utilidad para dar una visión más completa. Al hacerlo, descubrimos varios **hitos**:

- Existe una fuerte presencia de empates en los datos, por lo que se planteó una optimización del código para poder ver los datos según los requisitos de cada query. Por ejemplo, en querys con muchos datos, se sacaron % de agrupaciones de datos para una mejor visión de conjunto.
- La atracción más visitada tiene nombre “desconocido”, por lo que se implementó un ranking para ver cuáles eran las siguientes. Esta solución también comprende los empates, para evitar pérdida de información.
- Las consultas extra se han utilizado principalmente para la parte de visualización.

En cuanto a los **insights** o **conclusiones** más reveladoras que hemos obtenido, destacamos las siguientes:

- Los visitantes suben como máximo a 7 atracciones, y como mínimo a 1.
- Asimismo, el 90,74% de los visitantes se han subido solo en una atracción durante su estancia en el parque.
- La emoción que predomina es “feliz” en todas las atracciones.
- Todas las atracciones tienen medias de valoración muy similares (5/10).
- El tiempo de espera es muy similar en todas las atracciones (12,5-13 mins).
- Ni la procedencia, ni el día, ni el tiempo de espera parecen ser factores que afecten a la valoración.
- Las entradas tienen el mismo precio independientemente de la antelación de su compra.

Por último, mencionar que este proceso en MySQL ha marcado una diferencia notable a nivel de **optimización de espacio** si la comparamos con los datos en bruto que teníamos anteriormente. Estos han sido las mejoras:

- **Diferencia JSON vs SQL (sin incluir las consultas)**: Ahorramos **8.73 MB (51%)** con SQL respecto al JSON.
- **Diferencia JSON vs SQL (con consultas incluidas)**: Si contamos las consultas, la diferencia es de **0,996 MB (5.4%)** más que con el JSON, tampoco muy elevado.
- **Diferencias entre SQL sin consultas y SQL con consultas**: El SQL con consultas ocupa **9.69 MB** más que la versión sencilla.

Frente a **mejoras futuras de este proceso** en SQL, hemos estimado el **ahorro en tiempos de procesamiento** (CPU) que supone usar 2 tablas estáticas ("atracciones" y "procedencia") en la propia base de datos (35.885 filas en "ticket_atracciones") para 1 año. Simulando un crecimiento x2 (71.770 filas) para un 2º y x3 (107.655 filas) para un 3º.

Las tablas estáticas son las siguientes:

- "atracciones": Almacena IDs y nombres de las atracciones
- "procedencia": Almacena IDs y nombres del país de procedencia de visitantes
- Además, reemplazan textos largos en "ticket:atracciones" (como "Rueda de la fortuna") por IDs más cortos

El ahorro en el procesamiento de datos de la CPU se produce porque:

- Buscar IDs es más rápido que buscar textos largos
- Consultas como las KPIs (Ej. recaudación, valoraciones, etc.) hacen JOINs más rápidos con tablas pequeñas e indexadas, lo que reduce el tiempo que el servidor tarda en procesarlas.

Ahorro total de un **30% menos de tiempo por consulta** (estimación simple). **Queries/día aprox** de 500 (suposición para reportes diarios).

Escenario	Filas	Ahorro (secs/año)	Ahorro (días CPU)
Actual	35.885	1.095.000	13 días
Crecimiento x2	71.770	2.190.000	25 días
Crecimiento x3	107.655	3.285.000	38 días

CONSTRUCCIÓN DEL MODELO AUTOMÁTICO DE DETECCIÓN DE EMOCIONES

Llegados a este punto del proyecto, nuestro objetivo ha sido desarrollar un **sistema de clasificación automático** para **7 emociones** ("*angry*", "*disgust*", "*fear*", "*happy*", "*neutral*", "*sad*", "*surprise*") a partir de **imágenes faciales** en formato 48×48 px y escala de grises, usando el dataset de PontIA World y splits aproximados de **train ≈ 22.968, val ≈ 5.741, test ≈ 7.178**.

PREPARACIÓN DE LOS DATOS

- División: *TRAINING* de un conjunto de 28.709 imágenes etiquetadas con un split de 0.2 : **train ≈ 22.968**, **val ≈ 5.741.** TEST de un conjunto de 7.178 imágenes.
- Aumentación moderada (rotación, traslación, flip, zoom).
- Normalización: Reescalado a [0,1]
- *Class Weights*: Para equilibrar las clases desbalanceadas {0:1.0266, 1:9.4016, 2:1.0010, 3:05685, 4:0.8261, 5:0.8492, 6:1.2933}

MÉTRICAS Y CRITERIOS DE EVALUACIÓN

Cuando comenzamos a realizar las primeras pruebas con modelos pre entrenados pudimos observar que el ***accuracy*** no era la mejor forma de evaluar debido a un desbalance de clases. Por ello, se ha usado como KPI principal el **Macro-F1** y como KPI de referencia el ***accuracy***, por los siguientes motivos:

- **Macro-F1 (principal)**: Promedia el F1 por clase dando el mismo peso a cada emoción, siendo más crítico con los desbalances (Ej. "*disgust*" tiene muy pocas muestras). Se ha usado como métrica de decisión del mejor modelo que hemos logrado.
- ***Accuracy* (principal)**: Media entre acierto y el total, es útil para aportar una visión global, pero enmascara problemas en clases minoritarias o con poca presencia. La elegimos como métrica complementaria a Macro-F1 para entender mejor el modelo y su evolución.

Otras métricas que hemos utilizado:

- Por clase (Precisión/Recall/F1 por emoción): Imprescindibles para conseguir equidad por clase y para saber dónde se han mejorado/empeorado con cada enfoque que se ha realizado.
- Matriz de confusión (análisis de errores): Muestra pares de confusión clave para diseñar mejoras. Esto se ha utilizado para guíar augmentación, *class weights*, *oversampling* focalizado y ajustes de umbrales.
- Función de pérdida (*loss*) — "*sparse_categorical_crossentropy*": Se usa para entrenamiento y control de sobreajuste (junto a *callbacks*), no como métrica de selección final. Ayuda a garantizar generalización; la acompañamos de métricas de clasificación para decidir.
- Soporte por clase y distribución del dataset: Reportamos el soporte (nº de ejemplos por emoción) y el desbalance (Ej. *disgust* ≈ 1,5% del total), porque condiciona el rendimiento y justifica priorizar Macro-F1 y métricas por clase.
- Criterio de selección del mejor modelo: Hemos elegido el modelo con mejor Macro-F1 y mejor *accuracy* en TEST.

Resumiendo, nuestros desarrollos confirman que Macro-F1 es la brújula para un problema emocional multiclase y desbalanceado mientras que *accuracy* complementa la lectura global. Y las métricas por clase y la matriz de confusión nos dicen exactamente qué emociones reforzar y cómo.

EVOLUCIÓN DE LOS MODELOS: DEL PRE ENTRENAMIENTO GENÉRICO AL CNN COMBINADO CON CLASIFICADOR

En las primeras exploraciones vimos que los clasificadores tradicionales (Capas lineales/SVM/RF) no captan bien las emociones y los rasgos que las diferencian, arrojando valores de métricas muy bajos. A raíz de investigar en la literatura de modelos basados en el FER2013 empezamos probando modelos pre entrenados (MobileNet Y MiniXception) y aprendimos sus límites con nuestras imágenes 48×48. Pasamos a una **CNN propia** sencilla, que sirvió de base para iterar y profundizar hasta convertirla en **extractor de *embeddings*** y, finalmente, combinar decisiones

con un **clasificador en *ensamble*** que equilibra precisión global y balancea las clases.

Fase 0 — *Transfer learning* (punto de partida)
William empezó a trabajar con MobileNet como atajo de conocimiento visual general. Resultado: **~48% accuracy**; el **mismatch de escala (48×48)** y la falta de ajuste fino impidieron capitalizar el pre entrenamiento. Decidimos no seguir por esta vía después de probar 9 variaciones.

Fase 1 — CNN propia "simple" (línea base fiable)
Ana empezó con una arquitectura secuencial básica sin nada contra el desbalance. Resultado: **~0,54 accuracy** y **0,50 en Macro-F1 pero resultó mejor que el pre entrenado**. Aporta trazabilidad y control del pipeline, pero aún floja en clases sutiles/minoritarias y con recall bajo.

Fase 2 — CNN "profunda" y estable (salto de calidad)
William le dió más profundidad a la CNN base e incrementamos capacidad y estabilidad con una augmentación moderada, adición de ***class weights*** para balancear y adición de BatchNorm, Dropout y callbacks (EarlyStopping/ReduceLROnPlateau). Resultado: **~0,63 accuracy** y **~0,61 Macro-F1**, con mejora clara en equilibrio entre emociones. Descubrimos que la **calibración y regularización** importan más que añadir capas sin control. Seguimos teniendo problemas para distinguir entre "*fear*"/"*sad*"/"*angry*".

Fase 3 — De clasificador a "*backbone* + *embeddings*" (especialización)
Usamos la CNN profunda como extractor de *embeddings* y añadimos un clasificador lineal (y uno fino para el bloque difícil "*fear*"/"*sad*"/"*angry*"). Observamos que el **router al clasificador fino** apenas impacta por cobertura limitada (**≈11,6% TEST**).

Fase 4 — *Ensamble* suave (mejor compromiso final)
Por último, decidimos hacer un *ensamble* entre nuestra mejor CNN y el clasificador especializado (LogisticRegression) en "*fear*"/"*sad*"/"*angry*" utilizando un **peso α=0,25**. Logramos un **0,6454 *accuracy*** y **0,6237 Macro-F1** en TEST (mejor rendimiento global y más justicia entre clases, sin sacrificar "*happy"/"surprise"*).

LIMITACIONES DE NUESTROS MODELOS Y CONCLUSIONES

- Desbalanceo de clases (*"disgust"* muy escaso, sesga el aprendizaje; aunque mitigado, sigue afectando "*fear*"/"*sad*"/"*neutral*".
- Solapamiento semántico entre expresiones sutiles ("*fear" vs "sad"/"angry"*), visible en las matrices de confusión.
- *Transfer learning* genérico (MobileNet) no se adapta bien a 48×48 en este *setup*; no aporta nada en la versión actual.

Variante (TEST)	Accuracy	Macro-F1	Key
CNN SIMPLE	0,54	0,50	Softmax del *backbone* sin ayudas
Baseline Complejo	0,63	0,61	CNN con augmentation y *class weight*
Especializado (Todas las clases)	0,6434	0,6233	CNN como extractor + capa lineal → mejora notable en clases difíciles.
Router (General + Fino)	0,6311	0,5874	Impacto global limitado.
Ensamble α=0,25	0,6454	0,6237	Mejor Macro-F1 y mejor accuracy global

FUNCIONAMIENTO DEL MODELO ELEGIDO

1. El modelo recibe la imagen, que entra en tamaño pequeño y en blanco y negro (para quedarse con las formas y gestos importantes).
2. Detecta patrones de la cara. La red CNN actúa como un extractor de rasgos: cejas, ojos, comisuras, arrugas. De ahí saca un resumen numérico (las "pistas" de la expresión).
3. Se combinan dos opiniones independientes:
    a. Generalista: Con esas pistas emite una probabilidad por emoción.
    b. Especialista: Usa las mismas pistas pero está entrenado para los casos confusos ("*fear*"/"sad"/"angry") y aporta su propia probabilidad.
4. Combinamos ambas opiniones en una proporción de 25% especialista / 75% generalista. Es como promediar dos dictámenes, dando un poco más de voz al generalista pero escuchando al especialista donde importa.
5. Elegimos la emoción con **mayor probabilidad** y mostramos también un **nivel de confianza**.

En el análisis comparativo de los modelos se ha comprobado que los datos tabulados no resultan suficientes para discriminar con precisión entre las siete emociones. Incluso técnicas avanzadas, como Random Forest o Gradient Boosting, apenas superan el rendimiento esperado por azar, mostrando una fuerte dependencia hacia la clase mayoritaria (*"happy"*). El bajo valor de las métricas de equilibrio, como el *Macro-F1 score*, evidencia que el desbalance de clases y la limitada capacidad informativa de las variables utilizadas constituyen un obstáculo para obtener un modelo fiable en este ámbito.

Por el contrario, los modelos basados en redes neuronales convolucionales (CNN) aplicados a imágenes presentan un rendimiento más alentador, especialmente en emociones con rasgos visuales claros como "*happy"* y "*surprise"*. No obstante, se mantienen dificultades en clases con menor representación, como *"disgust"*, y en aquellas más sutiles de distinguir ("*sad"*, "*fear"*, "*angry"* y "*neutral"*). Estos resultados apuntan a la necesidad de implementar estrategias de limpieza de datos, técnicas de balanceo (*oversampling* o *data augmentation*) y el uso de otras técnicas o modelos más complejos o combinados.

En este sentido, el último modelo probado analiza la foto de un rostro humano y decide la emoción correspondiente de entre las 7 posibles. Lo hace como un equipo de dos expertos; un **generalista** que suele acertar en la mayoría de casos, y un **especialista** que ayuda justo en las emociones más difíciles de distinguir. Al final, fusionamos sus dos opiniones y nos quedamos con la mejor respuesta.

.

CONCLUSIONES GENERALES FINALES

A partir de este trabajo, que el equipo ha realizado con los datos de PontIA World, se concluye que la **implementación de un sistema integral de análisis de datos** es fundamental para optimizar la experiencia del visitante y la rentabilidad del parque de atracciones. El proyecto **logró con éxito** estructurar y limpiar grandes volúmenes de datos dispares, construyendo una base de datos relacional en SQL que no solo **optimizó el almacenamiento en un 51%** , sino que también **facilitó la extracción de KPIs clave**.

A pesar de que **la emoción predominante en las atracciones es "feliz"** , el análisis reveló **puntos críticos de mejora**, como valoraciones medias consistentemente bajas en todas las atracciones (en torno a 5 sobre 10) y el hecho de que el 90,74% de los visitantes utiliza una sola atracción.

En respuesta a estos desafíos y a la dificultad de los modelos para clasificar con precisión emociones negativas y sutiles , **se ha propuesto una solución de IA Generativa** (encontrado en el documento que lleva el mismo nombre). Esta iniciativa busca abordar directamente las bajas valoraciones, las emociones negativas detectadas y la limitada interacción de los visitantes para, a futuro, mejorar tanto la satisfacción del cliente como los resultados del negocio.