De la telemetría a la estrategia: aprendiendo tiempos de vuelta y espacios latentes en Fórmula 1

Francisco Carruthers, Lucas Pini
Departamento de Ingeniería en Inteligencia Artificial
Universidad de San Andrés
Email: {fcarruthers, lpini}@udesa.edu.ar

***Resumen*—En este trabajo estudiamos el problema de predecir tiempos de vuelta, aprender representaciones de baja dimensión y simular estrategias de neumáticos en Fórmula 1 a partir de datos telemétricos provistos por la librería FastF1. Nos enfocamos en el Gran Premio de Mónaco 2025, utilizando vueltas de Franco Colapinto para entrenar modelos de regresión y evaluar estrategias, y vueltas de todos los pilotos para analizar el espacio latente. Empleamos *Gradient Boosting* para predicción de tiempo de vuelta, PCA y Autoencoder para reducción dimensional, *k-means* para clustering, y un simulador que evalúa seis estrategias distintas. Encontramos que el espacio latente se organiza principalmente por compuesto de neumático, con diferencias entre pilotos como variaciones sutiles. El simulador identificó que una estrategia H-S habría ahorrado $116{,}993\,\mathrm{s}$ respecto a la estrategia real H-M-M, ganando cinco posiciones en la clasificación final.**

I. Introducción

La estrategia de carrera en el automovilismo depende fuertemente de los tiempos de vuelta y de la gestión del neumático. Contar con modelos capaces de predecir el tiempo de vuelta permite comparar estrategias alternativas de parada en boxes y evaluar el impacto de decisiones tácticas, las cuales pueden definir los resultados de las carreras.

Este trabajo esta compuesto por tres partes:

- **Representación de baja dimensión y clustering**: Con el fin de interpretar de forma intuitiva cómo se organiza cada vuelta, aplicamos técnicas de reducción dimensional (PCA y Autoencoder) para proyectar las vueltas en un espacio latente 2D, lo que permite visualizar y agrupar vueltas con características similares. Analizamos si existen regiones del espacio latente que se correspondan con factores relevantes como el compuesto de neumático, el piloto o el ritmo (vueltas rápidas vs lentas). Luego utilizamos K-Means clustering para identificar patrones en los datos y evaluamos la estructura mediante métricas de homogeneidad y silhouette.
- **Predicción de tiempo de vuelta**: dada la información de contexto de una vuelta (número de vuelta, stint, compuesto, vueltas hechas con ese neumático, tipo de sesión, etc.), predecimos el tiempo de vuelta en segundos.
- **Simulación de Carrera**: Dada una carrera y distintas estrategias posibles, simula las vueltas usando el predictor de tiempo de vuelta y, comparando tiempos finales de carrera para cada estrategia, devuelve la optima (menor tiempo final de carrera).

Utilizamos Gradient Boosting con preprocesamiento (estandarización y One-Hot Encoding), entrenado con datos de prácticas libres y carreras de Colapinto en el circuito de Mónaco. Con estos modelos, evaluamos si una estrategia alternativa de neumáticos podría haber mejorado su resultado final.

II. Conjunto de datos y características

II-A. *Fuente de datos*

Utilizamos la librería FastF1 [1] para descargar los datos de la carrera de Fórmula 1 en Mónaco 2025. Para la parte supervisada nos enfocamos en las vueltas de Franco Colapinto en las prácticas libres (FP1-FP2-FP3) y en la carrera, mientras que para el análisis de representaciones usamos todas las vueltas válidas de todos los pilotos en carrera.

A nivel de vuelta, FastF1 provee información como:

- Tiempos: `LapTime`, `SectorTime`;
- Contexto: `LapNumber`, `Stint`, `TyreLife`, `Compound`, `TrackStatus`;
- Piloto/Auto: `Driver`, `Team`.

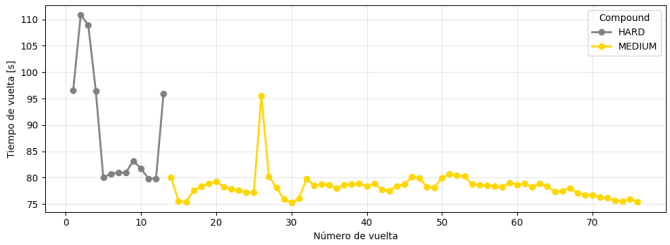


Figura 1: Evolución del tiempo de vuelta - Colapinto (Monaco 2025)

Convertimos los tiempos a segundos (`LapTime_s`) y filtramos vueltas con tiempos faltantes o vueltas borradas. Además, excluimos vueltas bajo banderas de seguridad u otras condiciones anómalas restringiendo a `TrackStatus == 1` (bandera “verde”).

II-B. *Limpieza de vueltas rápidas/lentas usando K-Means*

En prácticas, los pilotos alternan vueltas de instalación, vueltas lentas y vueltas de empuje. Para evitar entrenar con vueltas no representativas, definimos un esquema de filtrado basado en los tiempos de vuelta.

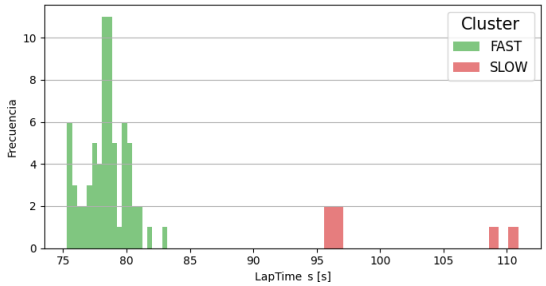


Figura 2: Histograma de vueltas de carrera - Colapinto: vueltas FAST vs SLOW (K=2)

En la carrera aplicamos *k-means* con $K = 2$ sobre el `LapTime_s`. De esta forma dividimos los datos en dos clusters C_{fast} y C_{slow} , y definimos C_{fast} como aquel con menor tiempo medio. Luego definimos el umbral de corte t y el desplazamiento respecto de la mejor vuelta de carrera $t_{\text{mín}}$ como:

$$\Delta_{\text{race}} = t_{\text{cut}} - t_{\text{mín}}.$$

Después utilizamos este Δ_{race} como parámetro de limpieza en todas las sesiones (FP1-FP2-FP3 y carrera), conservando sólo vueltas que satisfacen

$$t_i \leq t_{\text{mín}}^{(\text{sesión})} + \Delta_{\text{race}}.$$

Esto no solo nos dio mejores resultados a nivel métricas sino que también permitió entender mejor la distribución de los datos en base a `LapTime_s`. Podemos ver como quedan distribuidos los datos en el Histograma de la Figura 2.

II-C. Features para regresión

A partir del DataFrame resultante, construimos un vector de características x por vuelta. Para el modelo de tiempo de vuelta utilizamos un conjunto de *features* de alto nivel, evitando data leakage (no se usan variables futuras ni el propio `LapTime`). Las principales son:

- numéricas: `LapNumber`, `Stint`, `TyreLife`, una normalización de la fase de la sesión `lap_norm_session`, un indicador binario `is_race` y un índice ordinal `compound_order` que codifica "blando vs duro" de forma ordenada.;
- categóricas: `Session` (FP1/FP2/FP3/RACE) y `Compound`.

Estas *features* se normalizan con *StandardScaler* (numéricas) y *OneHotEncoder* (categóricas) usando `scikit-learn` [2].

II-D. Features para representaciones latentes

Para el estudio de representaciones latentes utilizamos el mismo conjunto de características, pero incluyendo vueltas de todos los pilotos en carrera. Importante: `Driver` y `Team` se usan sólo como etiquetas para colorear los gráficos y evaluar los clusters, pero no como *features* de entrada al PCA o al autoencoder.

III. Metodología

III-A. Regresión de tiempo de vuelta

Sea $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ el conjunto de vueltas de Colapinto, donde $x_i \in \mathbb{R}^d$ representa las características de la vuelta y $y_i \in \mathbb{R}$ el tiempo de vuelta en segundos. Buscamos una función f_θ tal que $\hat{y}_i = f_\theta(x_i)$ minimice el error de predicción.

Modelamos f_θ mediante un *Gradient Boosting Regressor* [**?**]:

$$f_\theta(x) = \sum_{m=1}^{M} \gamma_m h_m(x),$$

donde cada h_m es un árbol de decisión de baja profundidad entrenado secuencialmente para corregir los residuos del ensamble anterior. Utilizamos la implementación de `sklearn` con pérdida cuadrática y ajustamos hiperparámetros como número de árboles M, profundidad máxima y tasa de aprendizaje η mediante *Optuna* [**?**].

Entrenamos y evaluamos el modelo usando validación cruzada de K folds (con $K = 5$), manteniendo consistente el preprocesamiento dentro de un `Pipeline`.

III-B. Representaciones latentes con PCA

Como método lineal de reducción de dimensionalidad aplicamos Análisis de Componentes Principales (PCA). Dado el conjunto de vectores preprocesados $X \in \mathbb{R}^{N \times d}$, PCA busca una proyección $Z = XW$ tal que las columnas de Z maximicen la varianza explicada, con $W \in \mathbb{R}^{d \times k}$ ortogonal.

Seleccionamos $k = 2$ y $k = 3$ componentes principales, que explican aproximadamente el $76\,\%$ y el $94\,\%$ de la varianza, respectivamente. Utilizamos Z_{2D} (PC1 vs PC2) para visualizaciones coloreadas por piloto y por compuesto.

III-C. Autoencoder para reducción no lineal

Para capturar estructuras no lineales entrenamos un Autoencoder fully connected. El encoder es una red $z = g_\phi(x)$ que mapea $x \in \mathbb{R}^d$ a un espacio latente de dimensión $k = 2$, mientras que el decoder $\hat{x} = h_\psi(z)$ reconstruye la entrada. Entrenamos el modelo minimizando el error cuadrático medio

$$\mathcal{L}(\phi, \psi) = \frac{1}{N} \sum_{i=1}^{N} \|x_i - h_\psi(g_\phi(x_i))\|^2$$

mediante descenso por gradiente con Adam, utilizando la librería PyTorch [**?**]. Una vez entrenado, usamos z_i como representación latente de cada vuelta.

III-D. Clustering en el espacio latente

Aplicamos *k-means* sobre distintas representaciones:

- PCA 2D (Z_{PCA}),
- Autoencoder 2D (Z_{AE}),
- El escalar `LapTime_s` para $K = 2$ (vueltas rápidas vs lentas).

Evaluamos la calidad de los clusters usando *silhouette score* y medimos su alineación con etiquetas conocidas (compuesto, piloto) mediante *homogeneity score*.

III-E. Simulador de estrategias de carrera

Una estrategia de neumáticos S se define como una secuencia de stints $S = \{(c_j, n_j)\}_{j=1}^{J}$, donde $c_j \in \{\text{SOFT}, \text{MEDIUM}, \text{HARD}\}$ es el compuesto del stint j y $n_j \in \mathbb{N}$ el número de vueltas con ese neumático. La estrategia debe cubrir el total de vueltas de la carrera: $\sum_{j=1}^{J} n_j = L$, donde L es el número total de vueltas.

Para simular una estrategia S en una carrera específica, utilizamos el modelo de predicción de tiempo de vuelta f_θ entrenado previamente. Para cada vuelta $\ell \in \{1, \ldots, L\}$, construimos el vector de características x_ℓ según la estrategia S, incluyendo número de vuelta, stint actual, vida del neumático y compuesto. Predecimos el tiempo de vuelta "limpio"mediante $\hat{t}_\ell = f_\theta(x_\ell)$ y aplicamos penalizaciones a la variable p_ℓ según el contexto específico de la vuelta: se impone una penalización de $15\,\text{s}$ si $\ell = 1$ (correspondiente a la largada de la carrera), y una penalización de $20\,\text{s}$ si ℓ indica la vuelta en la que se realiza un *pit stop*.

El tiempo total de vuelta considerando penalizaciones es $t_\ell = \hat{t}_\ell + p_\ell$, y el tiempo total de carrera para la estrategia S resulta:

$$T(S) = \sum_{\ell=1}^{L} t_\ell = \sum_{\ell=1}^{L} \left(f_\theta(x_\ell) + p_\ell\right).$$

Para encontrar la estrategia óptima S^*, evaluamos un conjunto de estrategias candidatas $\mathcal{S} = \{S_1, S_2, \ldots, S_K\}$ y seleccionamos aquella con menor tiempo total:

$$S^* = \arg\min_{S \in \mathcal{S}} T(S).$$

IV. Experimentos, resultados y discusión

IV-A. Evaluación de la regresión de tiempo de vuelta

Para evaluar el modelo de tiempo de vuelta consideramos tres métricas estándar de regresión: error absoluto medio (MAE), raíz del error cuadrático medio (RMSE) y coeficiente de determinación (R^2).

IV-A0a. Comparación de modelos: Inicialmente comparamos distintos regresores clásicos sobre el mismo conjunto de *features* y esquema de validación cruzada ($K = 5$): Ridge (modelo lineal regularizado), Random Forest, Gradient Boosting y un perceptrón multicapa (MLP). Posteriormente se entrenaron modelos mas avanzados (XGBoost, LightGBM, HistGB) para comparar contra el mejor modelo obtenido previamente La Tabla I resume los resultados promedio por fold.

Cuadro I: Comparación de modelos (5-fold CV) para predicción de tiempo de vuelta.

Modelo	MAE	RMSE	R^2
Gradient Boosting	0.75	1.04	0.74
Random Forest	0.78	1.09	0.69
Ridge	0.99	1.32	0.60
MLP	2.81	3.40	−1,74
XGBoost	0.76	1.04	0.69
HistGB	1.14	1.43	0.46
LightGBM	1.21	1.48	0.43

Gradient Boosting obtiene el menor MAE y RMSE, y el mayor R^2 promedio, con desviaciones estándar moderadas. Random Forest y XGBoost se acercan en rendimiento pero son sistemáticamente peor en las tres métricas. Ridge, como baseline lineal, tiene un MAE y un RMSE sensiblemente mayores y un R^2 alrededor de $0{,}60$, lo que sugiere que existen interacciones no lineales relevantes entre las *features*. LightGBM y HistGB simplemente obtienen resultados peores que GB. El MLP presenta un desempeño claramente deficiente, con R^2 negativo, probablemente debido al tamaño reducido del conjunto de datos y a la mayor cantidad de parámetros del modelo, que lo hacen más sensible a sobreajuste y a la elección de hiperparámetros.

En base a esta comparación elegimos **Gradient Boosting** como modelo principal: ofrece el mejor compromiso entre precisión, estabilidad y facilidad de ajuste, y se integra de forma natural con el preprocesamiento tabular utilizado.

IV-A0b. Feature engineering y ajuste de hiperparámetros: A partir del modelo de Gradient Boosting, realizamos dos refinamientos:

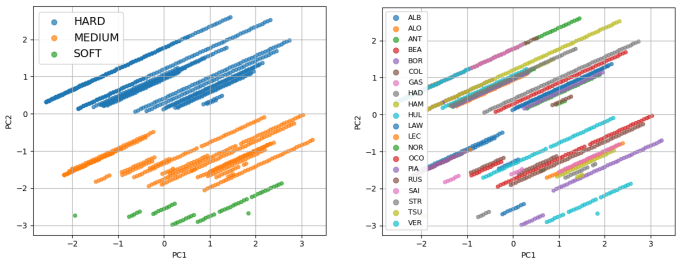
- **Feature engineering de alto nivel**: incorporamos *features* derivadas basadas en conocimiento del dominio, como la fase relativa de la sesión (`lap_norm_session`), un indicador de tipo de sesión (`is_race`) y un índice ordinal de compuesto (`compound_order`). Estas variables son fáciles de interpretar y no introducen fuga de información, por lo que son compatibles con el uso posterior del modelo en un simulador de estrategias. Tambien probamos features mas avanzadas pero simplemente empeoraban el rendimiento del modelo, por lo que decidimos quedarnos con las mencionadas anteriormente.
- **Tuning de hiperparámetros**: ajustamos número de árboles, profundidad máxima, tasa de aprendizaje y parámetros de regularización utilizando búsqueda bayesiana con Optuna sobre el mismo esquema de validación cruzada.

La Tabla II muestra el rendimiento final del modelo de Gradient Boosting tras estos pasos (5-fold CV).

Cuadro II: Rendimiento final de Gradient Boosting con feature engineering y tuning (5-fold CV).

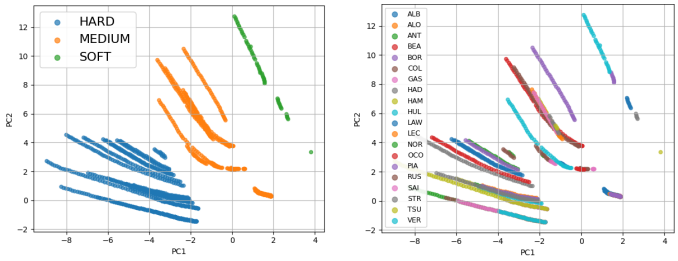
Métrica	Media	Desvío
MAE [s]	0.66	0.16
RMSE [s]	0.942	0.26
R^2	0.76	0.11

Las métricas finales son del mismo orden que las obtenidas en la comparación inicial de modelos: el modelo captura aproximadamente entre el $75\,\%$ y el $77\,\%$ de la varianza de los tiempos de vuelta de Colapinto en Mónaco 2025. El desvío estándar entre folds no es despreciable, lo que podría indicar que, con el tamaño actual del conjunto de datos, las métricas siguen siendo sensibles a qué vueltas quedan en cada partición de entrenamiento y validación. Aun así, en todos los folds el modelo mejora claramente sobre un predictor constante (la media) y sobre el modelo lineal.



(a) PCA por compuesto. (b) AE latente por Piloto.

Figura 3: Visualización de representación con PCA: Coloreado por Compuesto (izquierda) y Coloreado por Piloto (derecha).



(a) PCA por compuesto. (b) AE latente por Piloto.

Figura 4: Visualización de representación con AE: Coloreado por Compuesto (izquierda) y Coloreado por Piloto (derecha).

En términos relativos, el error absoluto medio de $\approx 0{,}66\,\text{s}$ representa menos del 1 % del tiempo típico de vuelta en Mónaco (del orden de 75–80 s). Este nivel de precisión es suficiente para comparar estrategias de parada mediante el simulador de carrera: diferencias de varios segundos en el tiempo total de carrera entre un esquema de dos paradas versus tres paradas, o entre combinaciones de compuestos (HARD–MEDIUM–SOFT, etc.). En cambio, no sería adecuado para decidir microajustes de pocas décimas por vuelta entre estrategias casi equivalentes.

En conjunto, los resultados sugieren que el modelo de Gradient Boosting con *features* de alto nivel capta de forma razonable la relación entre contexto de la vuelta (stint, compuesto, vida del neumático, fase de la sesión) y tiempo de vuelta, manteniendo al mismo tiempo una representación interpretable que puede integrarse en nuestro simulador de estrategias de carrera.

IV-B. PCA: estructura por compuesto y piloto

En la proyección PCA (PC1 vs PC2), que conserva alrededor del 76 % de la varianza, observamos una separación casi perfecta de las vueltas según el compuesto de neumático: las vueltas con HARD, MEDIUM y SOFT ocupan tres bandas bien diferenciadas del plano latente (Figura 3a. Esto indica que, dado el conjunto de *features* elegido, el compuesto es el factor que más estructura introduce en los datos.

Cuando coloreamos el mismo espacio por piloto, los colores se mezclan dentro de cada banda de compuesto: no aparecen clusters limpios por piloto, sino trayectorias que corresponden a los distintos stints dentro de cada compuesto. Esto sugiere que las diferencias de piloto son secundarias frente al efecto del neumático y de la fase de la vuelta/stint.

IV-C. Autoencoder: representaciones no lineales

En el espacio latente 2D del autoencoder, las vueltas vuelven a separarse de manera clara por compuesto, incluso más nítidamente que en PCA. Las regiones asociadas a HARD, MEDIUM y SOFT son prácticamente disjuntas, lo que confirma que el modelo no lineal también identifica al compuesto como variable latente dominante. Véase estos resultados en la Figura 4a

Coloreando por piloto, se observan líneas rectas continuas en el espacio latente que corresponden a la evolución de cada piloto a lo largo de sus stints, pero estas trayectorias se superponen dentro de cada compuesto. Al aplicar *k-means* sobre el espacio del autoencoder, la homogeneidad de los clusters es mayor respecto a la etiqueta de compuesto que respecto a la de piloto, lo que cuantifica esta observación cualitativa.

IV-D. Clustering de ritmo: vueltas rápidas vs lentas

Por último, aplicamos *k-means* con $K = 2$ sobre los tiempos de vuelta `LapTime_s` para separar automatiscamente vueltas rápidas y lentas. El cluster "FAST" agrupa vueltas cercanas al mejor ritmo de carrera, mientras que el cluster "SLOW" concentra vueltas de gestión, tráfico o neumáticos degradados.

Al proyectar estos clusters sobre el espacio PCA o del autoencoder, observamos que las vueltas lentas se concentran en los extremos de las lineas coloreadas por cada piloto. Lo que probablemente representa el principio y el final de cada stint. Esto refuerza la interpretación física del espacio latente aprendido.

IV-E. Simulación de estrategias de carrera

Para evaluar estrategias alternativas de neumáticos en el Gran Premio de Mónaco 2025, definimos seis estrategias candidatas que combinan los tres compuestos disponibles (SOFT, MEDIUM, HARD) en diferentes secuencias de stints, ajustadas para cubrir las 76 vueltas completadas por Colapinto. Las estrategias candidatas incluyen configuraciones de dos stints (M-H, H-M, H-S, S-H) y tres stints (S-M-H, M-M-H), variando tanto el compuesto inicial como los momentos de pit stop. La penalización de largada se fijó en 15 segundos (debido a que largan en parado y todos juntos) y cada pit stop en 20 segundos (tiempo perdido en boxes promedio para Mónaco).

Estrategia	Configuración	Tiempo [mm:ss.mmm]
H-S	35H + 41S	99:33.964
S-H	20S + 56H	99:34.524
M-H	25M + 51H	99:46.013
H-M	30H + 46M	99:59.558
S-M-H	15S + 22M + 39H	100:12.893
M-M-H	20M + 20M + 36H	100:17.715
H-M-M (Real)	13H + 13M + 50M	100:14.391

Cuadro III: Tiempos simulados de carrera para diferentes estrategias (76 vueltas, Mónaco 2025).

La Tabla III resume los tiempos totales simulados para cada estrategia. La estrategia óptima identificada es H-S (35 vueltas con HARD seguidas de 41 vueltas con SOFT), con un tiempo total de 5973,96 s. En contraste, la simulación con la estrategia real utilizada por Colapinto (H-M-M: 13 vueltas HARD, 13 vueltas MEDIUM, 50 vueltas MEDIUM) arroja un tiempo simulado de 6014,39 s, resultando en una diferencia de aproximadamente 40,4 s. El tiempo final de carrera real de Colapinto fue de 6090,957 s, mostrando una diferencia de 76,566 s entre la real y la simulación.

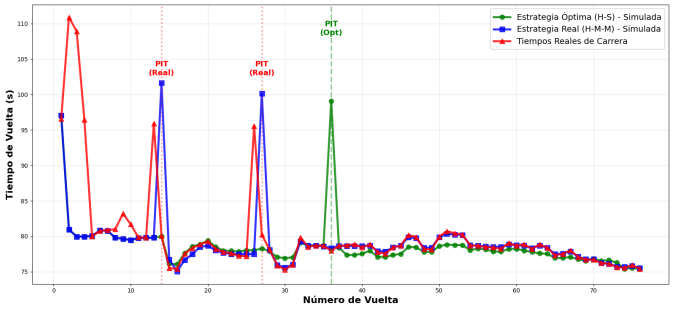


Figura 5: Comparación de tiempos de vuelta: estrategia óptima simulada (H-S), estrategia real simulada (H-M-M), y tiempos reales de carrera. Las líneas verticales indican pit stops.

La Figura 5 muestra la comparación de tiempos de vuelta entre la estrategia óptima simulada, la estrategia real simulada, y los tiempos reales de carrera. Se observa que el modelo captura correctamente las tendencias de degradación del neumático y el efecto de los pit stops. Las diferencias entre la simulación de la estrategia real y los tiempos reales (MAE $\approx$ 2,246 s/vuelta) son consistentes con el error del modelo de predicción base, validando la coherencia del simulador.

Para contextualizar el impacto de esta mejora potencial, insertamos a 'otro Colapinto' con la estrategia óptima en la clasificación final, con el mismo número de vueltas completadas, 76. La Figura 6 muestra que, con la estrategia H-S, Colapinto habría ganado 5 posiciones, pasando de P14 a P9. Si bien el estado de "doblado"limita el impacto absoluto, la mejora de 116,993 s sobre 76 vueltas es significativa en un circuito donde los adelantamientos son escasos y cada décima cuenta para la estrategia de carrera.

Es importante notar las limitaciones del simulador. Asume condiciones de carrera estables sin considerar tráfico variable, banderas amarillas, u otras eventualidades que podrían alterar significativamente los tiempos reales. La diferencia observada entre tiempos simulados y reales sugiere que factores no capturados por las features disponibles (como la gestión de combustible, dato que es privado de cada equipo) podrían mejorar la precisión del modelo en trabajos futuros.

V. Conclusión y Trabajo Futuro

Presentamos un sistema basado en datos telemétricos de FastF1 para (i) predecir tiempos de vuelta de un piloto específico, (ii) analizar la estructura latente de vueltas de Fórmula 1, y (iii) simular estrategias de neumáticos en carrera. Nuestro

Pos	Driver	Team	Time/Gap	Status
1	NOR	McLaren	100:33.843	Finished
2	LEC	Ferrari	+00:03.131	Finished
3	PIA	McLaren	+00:03.658	Finished
4	VER	Red Bull Racing	+00:20.572	Finished
5	HAM	Ferrari	+00:51.387	Finished
6	HAD	Racing Bulls	1 LAP	Lapped
7	OCO	Haas F1 Team	1 LAP	Lapped
8	LAW	Racing Bulls	1 LAP	Lapped
9	COL*	Alpine	2 LAPS	Lapped
10	ALB	Williams	2 LAPS	Lapped
11	SAI	Williams	2 LAPS	Lapped
12	RUS	Mercedes	2 LAPS	Lapped
13	BEA	Haas F1 Team	2 LAPS	Lapped
14	COL	Alpine	2 LAPS	Lapped
15	BOR	Kick Sauber	2 LAPS	Lapped
16	STR	Aston Martin	2 LAPS	Lapped
17	HUL	Kick Sauber	2 LAPS	Lapped
18	TSU	Red Bull Racing	2 LAPS	Lapped
19	ANT	Mercedes	3 LAPS	Lapped
20	ALO	Aston Martin	DNF	Retired
21	GAS	Alpine	DNF	Retired

Figura 6: Clasificación final del GP de Mónaco 2025. Piloto simulado (COL*) usando estrategia óptima H-S con y resultado real de Colapinto (COL).

modelo de Gradient Boosting, entrenado sobre *features* de alto nivel y cross validation, logra errores medios del orden de 0,7 s y explica aproximadamente el 73 % de la varianza de los tiempos de vuelta. En el espacio latente, tanto PCA como un autoencoder profundo muestran que el compuesto de neumático es el factor principal de organización, mientras que las diferencias entre pilotos aparecen como variaciones locales dentro de cada región de compuesto. El simulador de estrategias aplicado al GP de Mónaco 2025 identificó que una estrategia H-S habría ahorrado 116,993 s respecto a la estrategia real H-M-M, permitiendo ganar cinco posiciones (P14 $\rightarrow$ P9) y sumando dos puntos al campeonato mundial.

El modelo de Gradient Boosting demostró ser el más efectivo para predicción de tiempos de vuelta al capturar relaciones no lineales entre variables como TyreLife y Compound. PCA preservó el 76 % de varianza con alta interpretabilidad, mientras que el autoencoder capturó patrones más sutiles. El simulador permitió evaluar sistemáticamente seis estrategias candidatas bajo condiciones realistas.

Como trabajo futuro, sería interesante incorporar más circuitos y temporadas para mejorar la generalización del modelo, extender la predicción a todos los pilotos de la grilla y explorar arquitecturas más avanzadas que permitan modelar mejor la dinámica temporal de las vueltas. Si tuviésemos acceso a los datos, integrar al simulador condiciones dinámicas como cambios climáticos, degradación diferencial de neumáticos según temperatura de pista, cantidad de combustible, y estrategias reactivas basadas en la posición de competidores.

REFERENCIAS

[1] The FastF1 Project, "FastF1 Documentation," Available at: https://docs.fastf1.dev/, accedido 2025.

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[6] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems*, 2017.

[7] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] S. Lloyd, "Least Squares Quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[10] T. Akiba *et al.*, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.