Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Matemáticas y Física

Maestría en Ciencia de Datos

Comparación de modelos predictivos aplicados a las cuartas oportunidades en la NFL

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
Maestro en Ciencia de Datos

Presenta:
Rafael Cañedo

Director:
Dr. Jaime Emmanuel Alcalá Temores

Tlaquepaque, Jalisco, 5 de febrero de 2025

Comparación de modelos predictivos aplicados a las cuartas oportunidades en la NFL

Rafael Cañedo

Resumen

El presente trabajo presenta los resultados derivados de la implementación y evaluación de distintos modelos de clasificación, diseñados para anticipar el desenlace de las cuartas oportunidades en juegos de la NFL. Estos modelos se contrastan con un enfoque de clasificación estándar, representado por una regresión logística simple.

Durante el proceso de experimentación, se seleccionaron modelos de *Machine Learning* que se ajustaran a la naturaleza del problema y la estructura de los datos, apoyados por un análisis estadístico. Además, se implementaron los modelos, y se ofrece una comparación entre ellos mediante diversas pruebas y manipulaciones que respaldan la robustez de los resultados.

Si bien se lograron mejorar categóricamente los resultados obtenidos por la regresión logística, los modelos propuestos encontraron limitaciones debido a la naturaleza del problema, que posee un fuerte componente aleatorio. En el apartado de conclusiones y trabajo futuro, se proporcionan reflexiones generales sobre el grado de incertidumbre epistémica de los modelos, y se propone como trabajo futuro realizar análisis de naturaleza bayesiana para comprender de manera integral la incertidumbre total de los modelos (epistémica o del modelo y aleatoria o de los datos).

Keywords: Sports, Data, Machine Learning, Predictive Modeling, NFL.

Tabla de Contenidos

Índice de figuras

Página

Índice de tablas

Página

Dedicado a mis papás, Luz María y Rafael así como a Fernando Donadt quien más allá de apoyarme en este camino siempre me demostró que creía en mí incluso cuando yo no creía tanto.

1 Introducción

En este capítulo se presenta el contexto del análisis de los modelos predictivos aplicados a la NFL, la justificación del objeto de estudio, la definición del problema y los objetivos generales y específicos.

1.1 Contexto

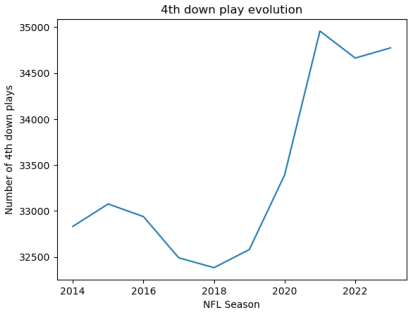

Figura 1.1: Número de 4tas oportunidades jugadas por temproada

La *National Football League*, a la cual nos referiremos en lo sucesivo como NFL por sus siglas en inglés, es la principal liga de fútbol americano en el mundo.

En el fútbol americano, la cuarta oportunidad se vuelve crucial, ya que generalmente representa la última oportunidad para evitar entregar el balón al rival. Como se puede observar en la Figura 1.1 a partir de la temporada 2020 (año en que coincidentemente empieza el auge de la IA), el número de cuartas oportunidades por temporada aumentó categóricamente.

En el pasado la toma de decisiones y el diseño del plan de juego se hacía de manera empírica, por algún tiempo para muchas personas la analítica parecía una idea peleada con la visión tradicional de lo que era el futbol americano, hoy en día es una idea adoptada por todos los equipos dentro de la NFL y los deportes en general.

En una liga donde los equipos son cada vez más competitivos, los métodos estadísticos para apoyar la toma de decisiones durante el juego han cobrado cada vez más relevancia en las agendas de los equipos,

siendo estos métodos y modelos la piedra angular en la elaboración de la estrategia previa al partido.

Existen muchos casos de éxitos de equipos que han logrado mejorar sus resultados y que han sido los mayores adoptadores de los métodos estadísticos en sus esquemas de juego. Uno de los ejemplos más claros es el de las Águilas de Filadelfia [1] quienes a voz de su entrenador Doug Pederson su estrategia de juego es mayoritariamente basada en data.

El uso de datos en la toma de decisión ha cambiado de manera radical el porvenir del futbol americano. La última temporada (2022-2023) vio un incremento radical dejando en máximos históricos los intentos de cuarta oportunidad, conversión de dos puntos, en los puntos extras y otras decisiones.

El uso de *big data* y analítica avanzada se ha expandido de manera tal que hoy en día existe un concurso anual organizado por la NFL llamado el *Big Data Bowl* [2]. Este concurso presenta a los participantes datos provistos por *AWS* y *Next Gen Stats* para que los mismos los analicen y presenten modelos y conclusiones sobre ciertos temas de interés en el fútbol americano, los cuales son de acceso público. Hasta 2023, más de 50 participantes del *Big Data Bowl* han sido contratados en roles de analítica deportiva.

El creciente interés por desarrollar modelos de *machine learning* y aplicar técnicas estadísticas en el deporte es lo que motiva y justifica el presente trabajo.

1.2 *Justificación*

Existe evidencia del creciente interés de la estadística aplicada y el uso de *Machine Learning* en la NFL y en los deportes en general.

El presenta trabajo resulta pertinente para todos aquellos directivos, entrenadores de jefe y personas involucradas o interesadas en entender desde una persepectiva estadística el impacto del contexto de una jugada en el devenir de la misma. El trabajo provee conclusiones útiles para que se optimicen las estrategias y decisiones en un partido de NFL.

1.3 *Problema*

Predecir el éxito o fracaso de una jugada $\mathbf{y}_i \in \{0, 1\}$, en función de un espacio de características denotado como $\mathbf{X}_i$ en un momento $\mathbf{t}$.

1.4 Objetivos

1.4.1 Objetivo general

Comparar modelos de *Machine Learning* aplicados a la predicción de éxtito/fracaso de las cuartas oportunidades en la NFL.

1.4.2 Objetivos específicos

1. Realizar un análisis detallado de las cuartas oportunidades
2. Seleccionar modelos de Machine Learning adecuados para el problema.
3. Implementar los modelos predictivos.
4. Comparar y analizar los modelos de acuerdo a métricas apropiadas.
5. Plantear posible trabajo futuro devenido de los resultados.

2 *Metodología*

En este capítulo se presenta en detalle el desarrollo metodológico que incluye la selección y el análisis de la base de datos, la selección de los modelos de *Machine Learning*, la implementación de los modelos, la interpretación de los resultados y el posible trabajo futuro del proyecto.

El estudio comparará las métricas de diferentes modelos predictivos también aplicará diferentes pruebas y validaciones para asegurar que los resultados obtenidos son robustos y consistentes.

A partir de los resultados obtenidos, se derivan una serie de conclusiones y recomendaciones generales sobre el modelo y su relación con el estado del juego.

2.1 *Descripción de los datos*

Los datos se obtuvieron de *NFLfastR* [3], la base de datos que se uso fue *play-by-play data*. La base de datos contiene un registro jugada por jugada de cada partido de la NFL desde 1999.

Para efectos del proyecto se generó una consulta que devolviera los registros que fueron cuartas oportunidades, que fueron pase o carrera y que no resultaron castigo, gol de campo, o reto de las temporadas 2019, 2020, 2021, 2022 y 2023.

Las tablas contienen más de 300 variables. De esas 300 variables sólamente algunas eran relevantes para el objetivo del proyecto.

Las variables preseleccionadas fueron:

- **yardline_100:** Distancia en número de yardas desde la zona de anotación del oponente para el equipo en posesión.
- **posteam_type:** Cadena que indica si el equipo que juega después está de local o de visitante.
- **half_seconds_remaining:** Segundos restantes en la mitad.
- **game_half:** Cadena que indica en qué mitad se encuentra el juego, ya sea Mitad 1, Mitad 2 o Tiempo extra.
- **goal_to_go:** Indicador binario que indica si el posteam se encuentra o no en situación de gol.
- **ydstogo:** Distnacia en yardas desde el marcador de primer intento o la zona de anotación en situaciones de gol.
- **ydsnet:** Valor numérico del total de yardas ganadas en la serie dada.
- **play_type:** Cadena que indica el tipo de jugada: pase, carrera, despeje, gol de campo, patada inicial, punto extra, qb kneel, qb spike, sin jugada (tiempos fuera y penalizaciones).
- **shotgun:** Indicador binario de si la jugada fue en formación de escopeta o no.
- **no_huddle:** Indicador binario que indica si la jugada se realizó en formación no huddle o no.
- **qb_dropback:** Indicador binario que indica si el QB retrocedió o no en la jugada.
- **pass_length:** Indicador de cadena para longitud de pase: corto o profundo.
- **pass_location:** Indicador de cadena para la ubicación del pase: izquierda, medio o derecha.

- **run_location:** Indicador de cadena para la ubicación de la carrera: izquierda, medio o derecha.
- **run_gap:** Indicador de cadena para el espacio entre líneas de carrera: final, guardia o tackle
- **score_differential:** Distancia en puntos entre equipo en posesión y equipo defendiendo previo al inicio de la jugada
- **fourth_down_converted:** Indicador binario que indica si la cuarta oportunidad fue exitosa o no.

2.2 *Análisis exploratorio*

2.2.1 *Análisis del problema*

Se trabajó con un dataset de 15 características y 2,764 observaciones. Existen tres tipos de variables, continuas, categóricas y binarias. La variable objetivo es *fourth_down_converted* que es binaria por lo que estamos ante un problema de clasificación.

Es importante antes de pasar al análisis estadístico entender los datos como elementos del juego para pdoer hacer inferencias correctas.

En el siguiente espacio se tratan de resolver 4 preguntas fundamentales. ¿Qué son las 4tas oportunidades?, ¿Por qué son importantes estas oportunidades?, ¿Qué contexto existe alrededor de estas oportunidades?, ¿Cómo juegan las probabilidades (desde el enfoque frecuentista) y el «premio» en la elección de «jugarse» una cuarta oportunidad?

Las cuartas oportunidades en números

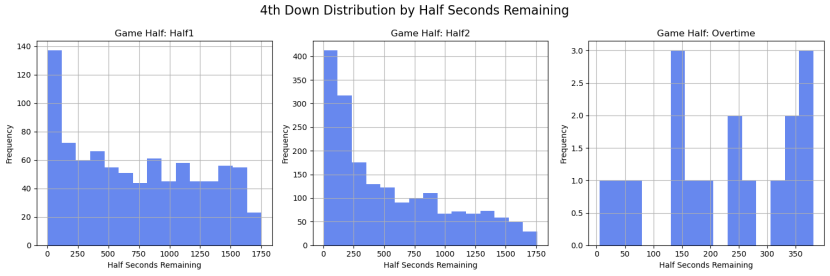

Figura 2.1: Distribución de las 4tas oportunidades a través del tiempo (Segundos restantes en la mitad)

En la Figura 2.1 se observa la correlación entre el tiempo y la frecuencia con la que se juegan las cuartas oportunidades. Durante la primera mitad la frecuencia en función del tiempo es uniforme (cuasi) esto es porque durante la primera mitad de los partidos las decisiones son más estrategícas. Durante la segunda mitad, donde se observa que hay más 4tas oportunidades conforme se acerca el final del partido,

esto indica que la decisión de jugarse la cuarta oportunidad se vuelve más situacional que estratégica.

Del análisis de las cuartas oportunidades en función del tiempo restante de partido se demostro la (a priori) importancia de variables de tiempo para el modelo. Sin embargo, se asume que no es un modelo líneal por lo que más variables deberían influir en las cuartas oportunidades.

En la Figura 2.2 se observa que también hay patrones marcados de acuerdo a la ubicación del terreno de juego. Se observa concentración en la zona central del campo (franja entre las yardas 30 y 60) y las yardas cercanas a la zona de anotación rival (franja entre la yarda 10 a la 0)

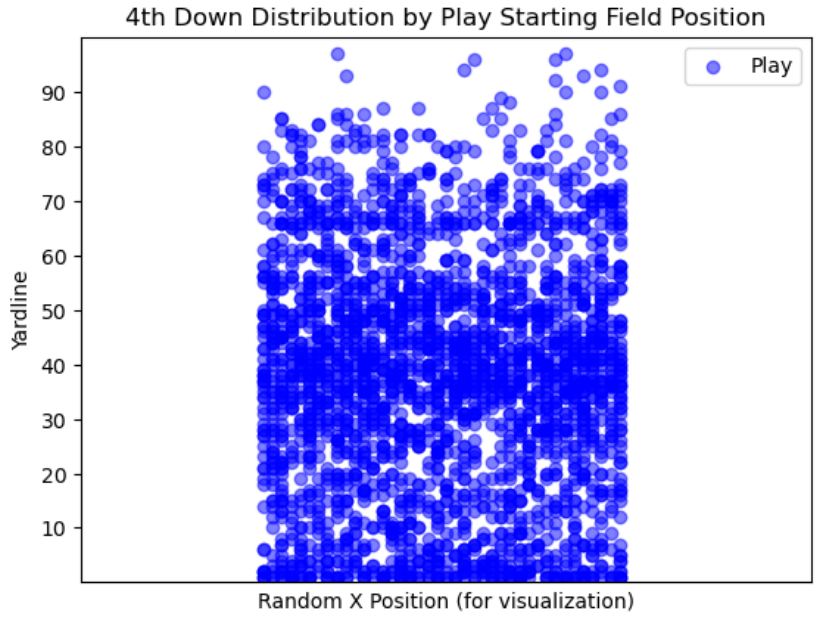

Figura 2.2: Distribución de las 4tas oportunidades según la posición del terreno.)

Los patrones respecto de la ubicación en el campo se explican con el balance riesgo-premio. En las yardas cercanas a la zona de anotación del equipo en posesión prefiere despejar, en las zonas medias se concetra la mayor parte de las observaciones porque las jugadas son aún lejos del rango promedio de gol de campo y lo suficientemente lejos de tu propia zona de anotación para despejar, en las yardas 30-10 es zona de gol de campo por lo que se prefiere patear gol de campo en lugar de tomar el riesgo de la 4ta oportunidad, mientras que en la zona más próxima a anotar el premio de convertir la 4ta son 7 puntos y el riesgo de no convertirla bastante bajo por eso existe de nuevo concentración de observaciones.

Cuando se combinan las variables de tiempo como de posisión, tal

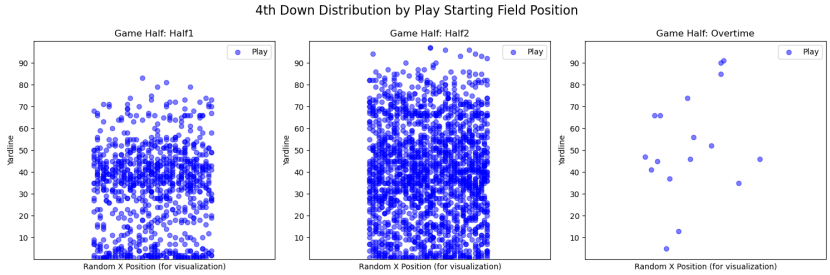

Figura 2.3: Distribución de las 4tas oportunidades según la posición del terreno y el tiempo de juego restante.)

como se puede visualizar en la Figura 2.3, se pueden observan patrones de tendencia aún más claros que abonan a la hipótesis de riesgo-premio, observamos que las decisiones en la primera mitad del partido parecen ser más estratégicas y que en la segunda mitad se empiezan a convertir en oportunidades más situacionales.

La probabilidad de conversión de una cuarta oportunidad, además de las variables propias del juego, es condicionada por variables relacionadas a la estrategia del equipo en posesión. En la Figura 2.4 se muestran la distribución real de jugadas en las pasadas 5 temporadas a nivel liga.

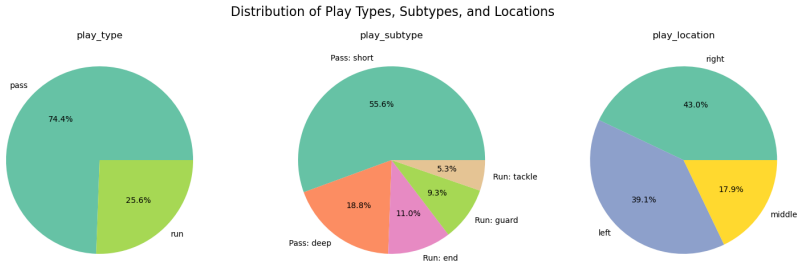

Figura 2.4: Distribución de las observaciones por tipo de jugada, sub tipo y dirección.)

Es importante demostrar que la elección de jugada puede influir también en las probabilidades de éxito de la jugada y analizar su correlación con otras variables. Por ejemplo, la Figura 2.5 muestra un patrón claro para las jugadas cercanas a zona de anotación en donde las jugadas de carrera son elegidas más frecuentemente mientras que en el resto del terreno de jeugo parecen los pases ser la opción preferida de los entrenadores.

Cuando se calcula la tasa de éxito de las cuartas oportunidades según el subtipo de jugada y la dirección de la jugada se obtiene un mapeo como el de la Figura 2.6 donde se logran identificar patrones claros, por ejemplo, en general las jugadas de carrera tienen una mayor tasa de conversión que los pases.

Con el análisis propuesto se logró ver la influencia de algunas

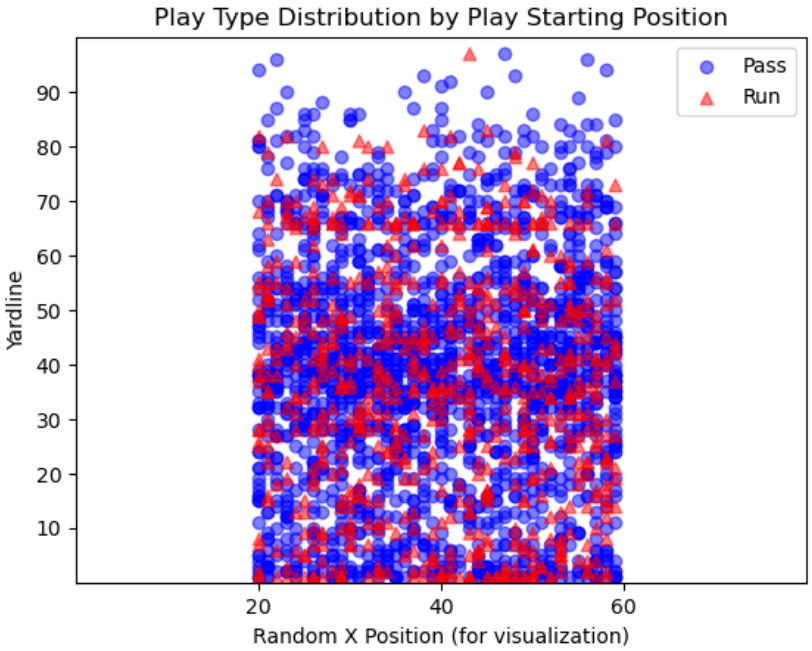

Figura 2.5: Distribución de las observaciones por tipo de jugada dada la ubicación en el terreno.)

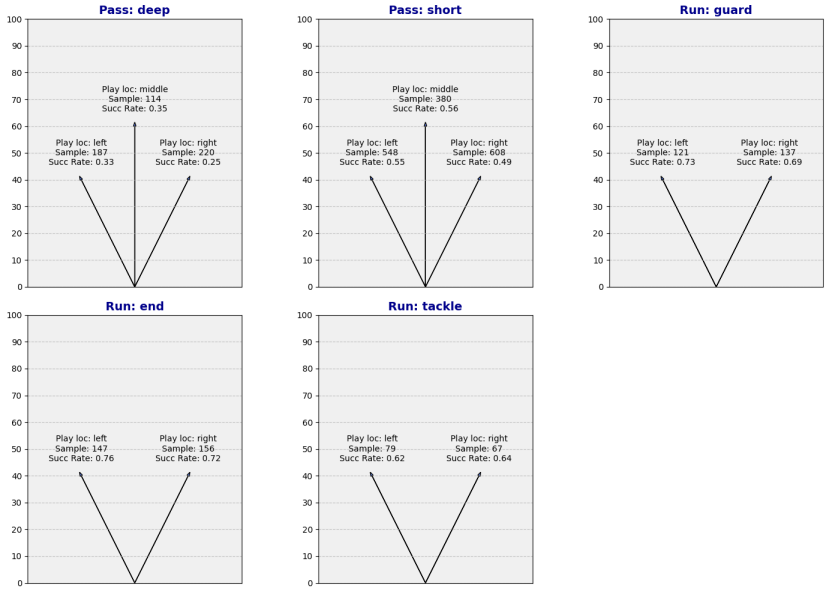

Figura 2.6: Tasa de éxito por subtipo de jugada y dirección.)

variables para confirmar la adecuada preselección de las mismas, se observaron algunos patrones claros en los datos y se permite avanzar al análisis estadístico con nociones e hipótesis más concretas.

2.2.2 *Análisis estadístico*

Ahora, aunado al análisis del juego es necesario un análisis estadístico orientado a validar los supuestos necesarios para el funcionamiento de los modelos con los que se trabajarán, para así definir un *pipeline* de transformación de datos.

Se empezó el análisis estadístico con el informe 2.1, que nos permitió validar ciertas características importantes del dataset con el que se está trabajando.

Names	**Type**	**Missing_values**	**Unique_values**
yardline_100	float32	0	96
posteam_type	object	0	2
half_seconds_remaining	float32	0	1216
game_half	object	0	3
goal_to_go	int32	0	2
ydstogo	float32	0	31
ydsnet	float32	0	118
play_type	object	0	2
shotgun	float32	0	2
no_huddle	float32	0	2
qb_dropback	float32	0	2
score_differential	float32	0	80
fourth_down_converted	object	0	2
play_location	object	0	3
play_subtype	object	0	5
yardline_group	category	0	10

Tabla 2.1: DQR

Con el *DQR* se pudo observar la no presencia de valores nulos, las variables categóricas y el tipo de registro de cada una de las variables.

Además de la calidad de los datos, fue importante ir validando los principales supuestos de los modelos para poder implementar un pipeline de transformación de datos. Los principales supuestos que siguieron en el orden de validación fueron: la distribución de los datos, balance de clases a predecir, valores atípicos, escala de los datos numéricos y colinearidad.

En cuanto a la distribución de los datos, incluso cuando ninguno de los modelos asume una distribución normal, algunos de estos son sensibles a *outliers* y a las escalas por lo que normalizar datos ayuda a la convergencia y eficiencia del modelo.

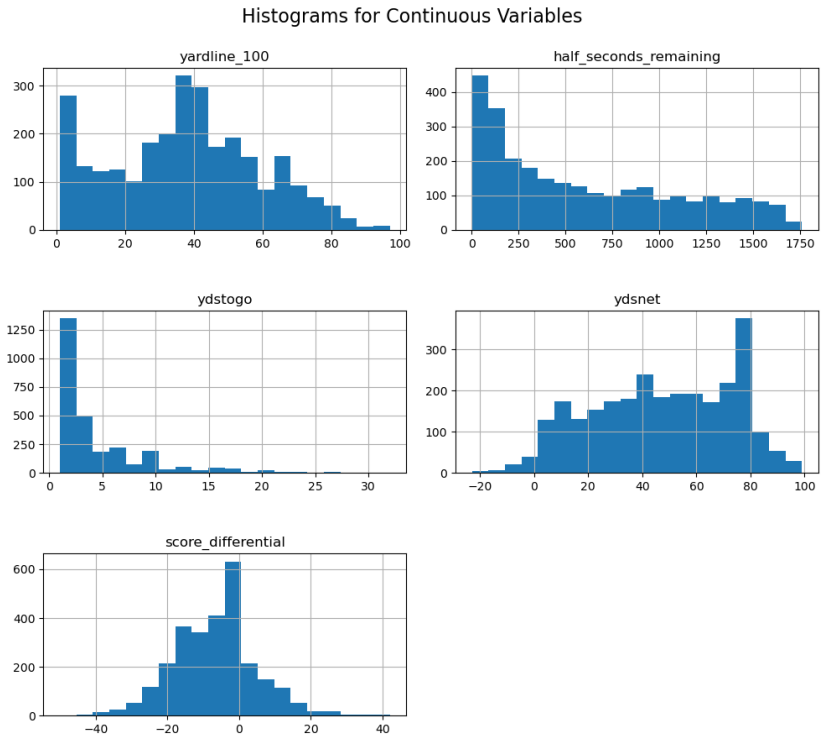

Figura 2.7: Histogramas para cada variable continua.)

En los histogramas de la Figura 2.7 se puede observar la presencia de algunos *outliers* y las diferencias en las escalas derivadas de las diferencias en la naturaleza de las características donde puede haber desde yardas, segundos o puntos.

Cuando se observa la distribución de clases en la Figura 2.8 para la variable a predecir, los resultados muestran una distribución casi perfectamente balanceada por lo que no se es necesario agregar una capa de balanceo de clases al *pipeline* de transformación de datos.

Finalmente, con el análisis del *VIF*, ver Tabla 2.2, se comprueba la no presencia de multicolinearidad.

Variable	**VIF**
yardline_100	2.942
half_seconds_remaining	2.324
ydstogo	2.148
ydsnet	2.568
score_differential	1.442

Tabla 2.2: Análisis del VIF para descartar colinealidad

Se concluyó el análisis estadístico creando un *pipeline* 5.1.1 que contiene los siguientes pasos:

1. Aplicar OneHot Encoding
2. Aplicar Robust Scaling

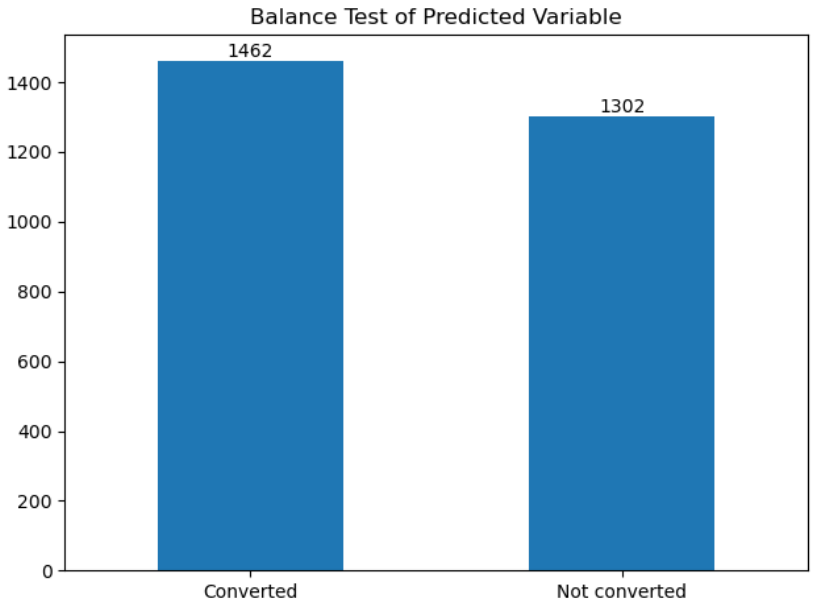

Figura 2.8: Distribución de las clases para la variable a predecir.)

3. Reducción del espacio de características

2.3 *Descripción de los modelos*

En esta sección se incluye una descripción de los modelos utilizados así como el proceso de selección de estos modelos para justificar su pertinencia.

2.3.1 *Modelo benchmark: Regresión logística*

Dado que el objetivo de nuestro trabajo es conseguir una clasificación binaria, con mejores métricas, mediante el uso de módelos basados árboles y compararlo contra un modelo lienal de aprendizaje supervisado clásico que comúnmente es usado para este tipo de problemas el cual es la regresión logística[4], la cual básicamente intenta predecir la probabilidad de que una instancia pertenezca a una clase particular. Si la probabilidad estimada es mayor que 50 % entonces el modelo está prediciendo que la observación pertenece a la clase (clase positiva, etiquetada '1'), en otro caso este predice que no está en la clase (i.e. pertenece a la clase negativa, etiquetada '0').

La regresión logística funciona como un modelo de regresión lineal. Un modelo de regresión logística calcula una suma ponderada de las características de entrada, pero en vez de generar el resultado directamente como la regresión lineal, la regresión logística genera la logística de este resultado:

$$P = \sigma(\theta^{\mathrm{T}} x) \tag{2.1}$$

Lo que se entiende como logística es una función sigmoide que genera un valor entre 0 y 1 la cual se define como:

$$\sigma(t) = \frac{1}{1 + \exp(-t)} \tag{2.2}$$

Una vez el modelo de regresión logística ha estimado la probabilidad de que una instancia pertenezca a la clase positiva, esta puede hacer su predicción mediante:

$$y = \begin{cases} 0 & \text{si } P < 0.5 \\ 1 & \text{si } P \geq 0.5 \end{cases} \tag{2.3}$$

Notar que $\sigma(t) < 0$ cuando $t < 0$ y $\sigma(t) \geq 0.5$ si $t \geq 0$, entonces un modelo de regresión logística predice 1 si $\sigma^{\mathrm{T}} x$ es positivo y 0 si es negativo.

Para establecer el vector de parámetros θ tal que el modelo estime probabilidades altas para instancias positivas y bajas probabilidades para instancias negativas. Una función que logra cumpir esta condición es la siguiente:

$$\mathbf{C}(\theta) = \begin{cases} -\log(P) & \text{si } y = 1 \\ -\log(1 - P) & \text{si } y = 0 \end{cases} \tag{2.4}$$

Esta función, que se denominará función de costo, cumple puesto que $-\log(\mathbf{t})$ crece rápidamente cuando $\mathbf{t}$ se aproxima a 0 , entonces el costo será grande si el modelo estima una probabilidad cercana a 0 para una instancia positiva, pero también se obtendrá un costo alto si el modelo estima una probabilidad cercana a 1 para una instancia negativa. Esto explica que los costos son altos si se tienen probabilidades contrarias a lo que buscamos: probabilidades altas para instancias positivas y bajas para instancias negativas. Esto fue para una sola instancia, entonces la función de costo sobre todo el conjunto de entrenamiento será el costo promedio de todas las instancias de entrenamiento, la cual es llamada log loss:

$$J(\theta) = \frac{-1}{m} \sum_{i=1}^{m} [y_i \log(P_i) + (1 - y_i) \log(1 - P_i)] \tag{2.5}$$

Esta función es convexa por lo que se puede encontrar el mínimo global por medio del Gradiente descendente. Recordemos que $P_i = \sigma(\theta^{\mathrm{T}} x_i)$. Por tanto, las derivadas parciales de *log loss* son:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [\sigma(\theta \mathrm{T} x_i^j) - y_i] x_i^j = 1, 2, ..., n \tag{2.6}$$

2.3.2 *Árboles de decisión*

Los árboles de decisión son módelos no paramétricos[1] de aprendizaje no supervisado útiles tanto para clasificación como para regresión.

Un árbol de decisión puede ser visto como una aproximación constante por partes[2]. En el contexto de árboles de decisión la partición refiere a cómo el árbol divide el espacio de características donde se asume que la relación entre características y objetivo es aproximadamente constante, por tanto la predicción será aproximadamente constante [5].

Para entender el funcionamiento de los árboles de decisión se describen tres conceptos básicos a continuación.

- Nodo raíz: Es el primer nodo del árbol
- Nodo: Subespacios de decisión
- Hojas: Predicciones hechas por el árbol
- Impureza: Es el criterio utilizado para tomar decisiones (normalmente puede usar como parámetro la entropía o el índice de Gini).
- Hojas/Nodos impuros: Nodos que contienen una mezcla de clasificaciones, si la impureza del nodo es menor a un umbral definido se convertirá en hoja pero será hoja impura, si la impureza del nodo es mayor al umbral se seguirán agregando nodos para tomar la decisión
- Hojas puras: Los nodos que tienen sólo una clasificación se convierten en hojas y las hojas se denominan hojas puras

El índice de Gini se calcula de la siguiente manera:

$$\mathbf{X}_i GiniImpurity = 1 - P_0^2 - P_1^2 - ... - P_i^2 \tag{2.7}$$

donde p representa la probabilidad asociada a cada clase diferente dentro de una característica, para el caso de clasificación binaria (que atañe al presente trabajo) $P\epsilon(0,1)$. Por lo que, el índice de Gini para el caso de clasificación binaria puede ser representado por:

$$\mathbf{X}_i GiniImpurity = 1 - (\frac{Total_1}{n\_hoja})^2 - (\frac{Total_0}{n\_hoja})^2 \tag{2.8}$$

Donde Total_1 es el número de observaciones en la clase 1 para la variable X_i, Total_0 es el número de observaciones en la clase 0 para la variable X_i y n_hoja es el número total de observaciones del nodo.

Antes que los nodos se conviertan en clases[3] los nodos generan

[1] Se entiende por modelo no parámetricos modelos que no tienen un número fijo de parámetros determinado antes del entrenamiento. Son modelos útiles cuando la relación entre las características y el objetivo es compleja (tal es el caso de nuestro modelo).

[2] Para entender el concepto de 'por partes' se puede visualizar el concepto de una función definida en una secuencia de intervalos.

[3] De acuerdo con la descrición descrita en el apartado de conceptos básicos

otros nodos cuyas clases a menudo no están balanceadas (existen más observaciones en una clase que en otra), esto hace que las muestras no sean comparables. Es, entonces, el último paso para obtener la impureza de un nodo la de ponderar el índice en función del tamaño de las clases de manera tal que:

$$X_i TotalGini = \frac{n\_hoja\_izqu}{n\_hoja}(Gini\_izq) + \frac{n\_hoja\_der}{n\_hoja}(Gini\_der) \quad (2.9)$$

El proceso para generar un árbol de decisión es, en comparación con otros modelos, simple y sigue el proceso a continuación descrito.

1. **Inicialización del arbol:** Lo primero que se tendrá que responder, en la construcción del árbol de decisión es: ¿Con qué variable debería inicializar el árbol?

   a) Se genera un nodo de decisión para cada característica

   b) Se estima la impureza de cada característica

   c) Se selecciona el nodo más puro (se asume que la característica en cuestión clasifica mejor la variable objetivo)

2. **Creación de nodos:** De manera iterativa se crean nodos de decisión (con el mismo proceso descrito en el punto 1)

3. **Criterio de corte:** Las iteraciones paran hasta que:

   a) El nodo se convierte en hoja alcanzado pureza o o

   b) El nodo converge a un criterio de corte, estos criterios de corte mejoran el rendimiento del modelo y evitan el sobre ajuste

   - Profundidad máxima del árbol. Hiperparámetro que limita el número de niveles que puede tener el árbol
   - Muestras mínimas. Hiperámetro que define un número mínimo de muestras en un nodo para particionarse, en caso de que no alcance un número de muestras el nodo se convierte en hoja impura
   - Impureza mínima de un nodo. Hiperparámetro que define un grado mínimo de impureza que debe tener un nodo antes de particionarse, en caso de que no alcance el grado de impureza deseado el nodo se convierte en hoja impura
   - Reducción mínima de la impureza. Hiperparámetro que define un umbral mínimo para la reducción de impureza, se calcula la impureza antes y después de la partición, si la diferencia en la impureza es menor al umbral no se realiza la partición del nodo

4. **Optimización:** Cuando se trata de clasificar datos con un grado de incertidumbre la poda del árbol optimiza el costo computacional del entrenamiento así como aumenta el poder predictivo del modelo. El proceso de poda elimina particiones con poca validez estadística [6]. Algunos de los algoritmos de poda más comunes son:

   a) Poda por función de costo

   b) Poda por error

   c) Poda por validación cruzada

¿Cómo funcionan los métodos de poda? En caso de encontrar sobreajuste en el modelo, se optará por aplicar poda por función de costo. La idea básica de este algoritmo es calular una función de costo para cada nodo, esta función indica cuánto afectaría la poda de ese nodo a la complejidad del árbol, iterativamente se selecciona el nodo con menor valor en la función, el proceso se repite hasta que sólo quede el nodo raíz [7].

La función de costo está dada por:

$$R_\alpha(T) = R(T) + \alpha \cdot |Hojas(T)| \tag{2.10}$$

donde

$$R(T) = \sum_{t \epsilon Hojas(T)} r(t) \cdot p(t) = \sum_{t \epsilon Hojas(T)} R(t) \tag{2.11}$$

donde $R(T)$ es el error en el entrenamiento, $Hojas(T)$ es el número de hojas del arbol T, $r(t) = 1 - max_k p(C_k)$ es el ratio de error de clasificación y $p(t) = \frac{n_t}{N}$ es el número de muestras en el node n_t respecto del total de las muestras del entrenamiento N. La variación en la complejidad del costo, está dada por $R_\alpha(T - T_t) - R_\alpha(T)$, donde T es el árbol de decisión, T_t es el sub-árbol con nodo raíz en el nodo t y el árbol podado al nodo t sería $T - T_t$. El orden de poda para los nodos internos es estimada igualando la función de costo del sub-árbol $T - T_t$ al de la rama en el nodo t:

$$g(t) = \frac{R(t) - R(T_t)}{|Hojas(T_t)| - 1} \tag{2.12}$$

Por último, se selecciona la relación de poda más debil usando $argming(t)$ y podando ese nodo iterativamente hasta que sólo quede el nodo raíz. El resultado es una secuencia de árboles y sus parámetros asociados a la función de costo.

2.3.3 *Random Forest*

Como se explicó, en la descripción de los modelos basados en árboles, los árboles (incluso después de la poda) pueden no manejar el ruido

en los datos de la mejor manera y ajustar mejor a los conjuntos de entranmiento que a la data futura que tenga que pronosticar. Una de las soluciones a este problema son los métodos de ensamblaje, dentro de ellos se encuentra el modelo *Random Forest*.

El *Random Forest* básicamente agrega y pondera los resultados de múltiples y más pequeños árboles de decisión [8]. Los árboles son 'más pequeños' porque se generan a través de sub-conjuntos de entrenamiento y de características. Mediante esta técnica es posible reducir el impacto del ruido en los datos.

Sin contar el impacto en el poder predictivo otro de los beneficios de usar Random Forest es la mejora en el rendimiento (en términos de tiempo computacional) del modelo sobre todo para sets de datos muy grande.

Habiendo descrito los dos principales beneficios del Random Forest es importante, también, que la interpretabilidad del árbol sobre cóomo y por qué realiza una predicción en particular se vuelve más complicada.

De manera general, el *Random Forest* funciona de la siguiente manera.

1. **Crear un sub-conjunto de datos.** A partir del conjunto de datos orginal, se crea un sub-conjunto de datos (lo que en inglés se conoce como *Bootstrapped sub-set*) mismo que debe de cumplir con algunas características [9]:
   - El muestro se debe hacer de manera aleatoria
   - El muestro debe ser con reemplazo (una muestra puede aparecer más de una vez en el sub-conjunto de datos)
2. **Crear un árbol de decisión con el sub-conjunto de datos creado en el punto 1.** El proceso es el mismo explicado en la descripción de los árboles de decisión, sin embargo, existe un matiz al proceso que es que el árbol se hará tomando únicamente un número n de variables, mismas que serán aleatoriamente seleccionadas. El número de variables aleatorias es un hiperarámetro del modelo y cómo se seleccionará el mejor hiperparámetro será explicado más adelante.
3. **Se crearán multiples árboles aleatorios.** Mismos que conformarán el bosque aleatorio y se crearán siguiendo los puntos 1 y 2 de este módulo.

Una vez construido el bosque aleatorio, el bosque aleatorio genera una predicción de la siguiente manera.

- Primero, el modelo tomará una observacióndel sub-conjunto de datos
- Después, el modelo evaluará esa observación en cada árbol
- Cada árbol generará su propia predicción

- El bosque aleatorio agrupará las predicciones de cada árbol
- El bosque seleccionará la clase con mayor número de predicciones y utilizará esa clase como la predicción del bosque

Finalmente, existen dos hiperparámetros mínimos para usar *Random Forest*, el número de árboles y el número de variables aleatoria. En el caso del presente estudio se usará un *grid* para evaluar diferentes parámetros, para el caso de las variables aleatorias por árbol el canon propone empezar con el cuadrado de las observaciones y probar con algunos valores arriba y debajo de esa base.

2.3.4 *XG Boost*

El algoritmo *XG Boost* es parte de una familia de algoritmos llamados *Gradient Boosting algorithms*, se puede decir que el boosting del gradiente son técnicas aplicadas a otro algoritmo de aprendizaje automático.

Los algoritmos de *Gradient Boosting* se basan en la idea de mejorar iterativamente un modelo de ensamblaje (en nuestro caso Random Forest) expresados como F_i mediante la adición secuencial de modelos débiles (cada árbol de decisión) expresados como f_i [10].

La escencia del proceso radica en la manera en que se actualiza el modelo de ensamblaje en cada iteración. La ecuación $F_{i+1} = F_i - \alpha \cdot f_i$ representa ese proceso. En la ecuación α controla la magnitud de la contribución del modelo débil f_i al modelo de ensablaje, una α más grande permitirá una corrección más rápida del error en los residuales pero aumenta el reisgo de sobre ajuste.

El proceso de *Gradient Boosting* involucra tres pasos [11].

1. Primero, se debe de identificar una función de costo diferenciable que ajuste bien al problema.
2. Se crea un *weak learner* para hacer predicciones
3. La creación de un modeo aditivo que use las predicciones del *weak learner* para reducir la función de costo

Básicamente, con los métodos de *Gradient Boosting* se entrena, en cada iteración, un nuevo árbol para que se ajuste al error del bosque. Esto es que el árbol f_i se enfoca en las áreas donde el bosque F no es suficientemente preciso. Cuando el árbol f_i se agrega al bosque F el modelo resultante $F_i + 1$ es una versión mejorada de F.

XG Boost, se distingue por aplicar regularización L1 o L2 a la función de costo [12], y por su enfoque de subprocesos múltiples que optimiza la utilización de la CPU, reduciendo el costo computacional [11].

Dado que *XG Boost* se basa en árboles de decisión como clasificadores base, se utiliza una variación de la función de costo para controlar la complejidad de los árboles [13]:

$$L_{xgb} = \sum_{i=1}^{n} L(y_i, F(x_i)) + \sum_{m=1}^{M} \Omega(h_m) \tag{2.13}$$

donde

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{2.14}$$

donde T es el número de hojas del árbol y el vector w son las salidas (las predicciones) de las hojas. Valores altos de γ resultan en árboles más simples. El valor de γ es el parámetro que controla la ganacia mínima (en la función de costo) necesaria para dividir un nodo interno.

Algunos otros hiperparámetros que el algoritmo usa y ayudan a mejorar los resultados del modelo son el *srhinkage* denotado como α en la ecuación del gradiente, los hiperparémtros propios del árbol como los umbrales de impureza, hojas y profundidad. Además de los hiperparámetros mencionados, *XG Boost* utiliza técnicas de aleatorización como submuestreo aleatorio para entrenar árboles individuales y submuestreo de columnas a nivel de árbol y de nodo de árbol que ayudan a reducir el sobreajuste y el tiempo de entrenamiento del modelo.

2.4 *Descripción de las métricas*

En esta sección se incluye una descripción de las métricas utilizadas para la comparación de los modelos y el proceso de selección de estas métricas para justificar su pertinencia.

2.4.1 *Métricas seleccionadas para evaluar el modelo*

- **Accuracy**
  - *¿Qué es?* La proporción de predicciones correctas sobre el total de predicciones.
  - *¿Cuándo es útil?* Cuando la todas las clases son igual de relevantes para el modelo
- **F1 Score**
  - *¿Qué es?* La media armónica entre Precisión y Recall donde el mejor valor del F1 score es 1 y el peor es 0.
  - *¿Cómo se calcula?*

    $$F1Score = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.15}$$

    donde TP son los verdaderos positivos (predicciones positivas cuyo valor real era positivo también), FP son falsos positivos (predicciones positivas cuyo valor real era negativo) y FN son

falsos negativos (predicciones negativas cuyo valor real era positivo).

- *¿Cuándo es útil?* Cuando existe un desbalance de clases (en nuestro caso aunque no es tan grande existe desbalanceo en las clases)

Por la naturaleza del problema no es relevante conocer la naturaleza de las predicciones incorrectas, sin embargo, se presenterán las matrices de confusión de los modelos ya que, a priori, se conoce que el modelo puede tener limitaciones para predecir jugadas en las que la evidencia estadística sea no significativa para poder predecir éxito o fracaso.

2.5 *Descripción de los experimentos o simulaciones*

En esta sección se incluye una descripción detallada de los experimentos o simulaciones computacionales realizados.

Se realizaron diferentes experimentos en el entrenamiento del modelo, algunos fueron optimizando los hiperparámetros mientras que otros fueron modificando las características del data set.

Los entrenamientos más relevantes fueron.

- *Feature Importance*. Se aplicó un modelo de regresión con el cuál, a través del peso de los coeficientes, se hicieron pruebas quitando las variables menos releventas según este criterio.
- *Random y Grid Search* para optimizar hiperparámetros. Al inicio, la selección de hiperparámetros fue con valores arbitrarios y buscando el patrón en el mismo que convergiera en mejores resultados, cuando se tuvo una idea de la posible área sobre la que convergían los mejores resultados se aplicó grid search sobre esa área para encontrar la combinación de hiperparámetros que optimizara las métricas.

3 Resultados y discusión

En este capítulo se presentan los resultados obtenidos del desarrollo de este trabajo y una discusión sobre el objeto de estudio.

3.1 Resultados y discusión

Los resultados obtenidos fueron satisfactorios. El *XG Boost* mostró el mejor rendimiento en las tres métricas evaluadas. Por otro lado, el modelo neuronal optimizado (*RNN*) mostró resultados muy similares en cuanto a *accuracy* y *recall*. Sin embargo, dada la cercanía en los resultados, factores como la interpretabilidad—que se ve limitada en los modelos neuronales debido a la complejidad de las transformaciones de pesos a través de funciones de activación—pueden cobrar mayor relevancia a la hora de seleccionar un modelo.

Modelo	**Accuracy**	**Recall**	**F1 Score**
Regresión Logística	79.566 %	82.9351 %	81.1352 %
Random Forest	75.226 %	80.2047 %	77.4299 %
XG Boost	85.5334 %	84.3003 %	86.0627 %
RNN	81.5551 %	80.8873 %	NA

Tabla 3.1: Resultados de los modelos. Después de aplicar fine tuning.

A partir de los resultados obtenidos en la Tabla 3.1 se pueden hacer algunas precisiones interesantes de cada modelo y su adaptabilidad al problema.

1. **Desempeño del modelo XG Boost.** El modelo *XG Boost* sobresale con la mejor precisión (85.53 %) y el mejor balence entre las otras métricas. Su *Recall* y *F1 Score* (84.30 % y 86 %) sugieren que el modelo no sólo es efectivo para predecir las jugadas que terminan en conversión sino que también mantiene un buen equilibro entre falsos positivos y falsos negativos que son parámetros improtantes para problemas de clasificación binaria.

2. **Ventajas de la Regresión Logística.** La regresión logística aunque tiene un accuracy menor (79.57 %) en comparación con el modelo *XG Boost* ofrece un *Recall* muy sólido (83 %) ofrece un modelo útil para situaciones donde la intepretabilidad son crítcias aun y esto implique

sacrificar precisión. Su facilidad de implementación y análisis hacen de la regresión logística un modelo muy interesante para usar como modelo base para el problema de las 4tas oportunidades.

3. **Decisiones prácticas.** Siendo la métrica del *Recall* el parámetro principal para la selección del modelo, el *XG Boost* se postula como la elección adecuada. El *RNN* y *Random Forest* considerando el balance entre la facilidad de análisis y el desempeño se descartan como modelos para este problema en específico.

Los hiperámetros con los que se lograron optimizar las métricas en cada uno de los modelos se encuentran en el apéndice 5.1.2. Para la regresión logística se propone una regularización ridge a los datos, con un parámetro de aprendizaje de .25. Para el *XG Boost* se propone un parámetro de aprendizaje de .25, 100 árboles y una profundida máxima de 4 niveles por árbol.

Log files con los resultados de cada experimento pueden ser consultados en los apéndices 5.2.2 y 5.2.1.

Aunque los resultados fueron buenos, existen algunos matices que el lector debe de considerar para implementar *ML* en los deportes en general.

Primero que nada, la incertidumbre aleatoria [14] [1] en los deportes es entendida como aquellas observaciones que ni el más preparados de los expertos podría haber predicho correctamente el resultado de la jugada, podemos poner el ejemplo del Leicester ganando la Premier League en 2016, pocos modelos (o expertos en futbol) habrían predicho ese suceso.

[1] En un modelo de ML, el error puede ser medido usando métodos *UQ* (*Uncertainty Quantification*), uno de los approaches más aceptados es el de separar la incertidumbre (el error de un modelo) en Epistémico (propio del modelo) e Aleatoria (propia de la data con la que se trabaja).

Segundo, las observaciones ruidosas. En el futbol americano, por la naturaleza del problema existen dos tipos de observaciones ruidosas. Las primeras son aquellas observaciones que, imaginando una frontera de decisión trazada por un hiperplano en el espacio de características, están muy cerca de la frontera de decisión; es decir comparten una probabilidad casi igual de pertenecer a una clase u otra, en algunos problemas de ML se decide crear un canal al rededor de la frontera de clasificación en donde todos esas observaciones no forman parte del resultado, en nuestro caso, por lo que implicaría en términos de interpretabilidad (no olvidar que el receptor de este mensaje es una persona sin bagaje técnico) se decidió dejar esas observaciones y asumir el costo (en términos de poder predictivo) de esas observaciones ruidosas.

Otro de las ideas que se explorarán más a detalle en el siguiente capítulo, la idea de cambiar el acercamiento del modelo a un modelo Bayesiano podría resultar en no sólo mejorar las métricas sino la interpretabilidad del mismo.

4 Conclusiones y trabajo futuro

4.1 Conclusiones

El proyecto proponia la tarea de comparar modelos de Machine Learning aplicados a la predicción de éxito o fracaso en cuartas oportunidades en la NFL, con el fin de identificar modelos que no solo ofrezcan un buen rendimiento, sino también una interpretación relevante en el contexto deportivo.

Los resultados expuestos en el capítulo 3 proponen al modelo *XG Boost* como el modelo óptimo estrictamente en función de las métricas ya que mostró un rendimiento superior en todas las métricas de interés (*accuracy*, *recall* y *F1 Score*) lo cual lo convierte en un modelo efectivo para capturar los patrones de las jugadas que terminan en conversión (principal objeto de estudio del documento). La capacidad de balancear falsos postivos con los falsos negativos lo hace un modelo bueno para problemas de clasificación binaria en el futbol americano donde las decisiones deben ser rápidas y precisas.

Es importante plantear como un modelo interesante a la Regresión Logística, aunque con un rendimiento ligeramente inferior al *XG Boost*, el modelo tuvo un *recall* (capacidad de predecir las instancias positivas correctamente) alto y tiene muchas ventajas en cuanto a la interpretabilidad que para gente de deportes puede ser de mucho más valor que mejorar un par de puntos porcentuales otros modelos más complejos. En el mismo balance de métricas vs interpretabilidad la red neuronal no supone resultados categóricamente superiores como para preferirla sobre *XG Boost* o la Regresión Logística misma.

A lo largo de este trabajo, se lograron los objetivos específicos. Se realizó un análisis detallado de las cuartas oportunidades en la NFL, seleccionando e implementando modelos de *Machine Learning* que mostraron potencial en este contexto. Se aplicaron métricas de rendimiento clave, lo que permitió una comparación sólida entre los modelos y facilitó la selección del más adecuado en función de la precisión y la interpretabilidad.

El estudio también reveló la importancia de considerar la incertidumbre inherente en los deportes, como las observaciones

ruidosas y los eventos improbables que no pueden predecirse con precisión. Estos hallazgos sugieren que futuras investigaciones podrían beneficiarse de enfoques Bayesianos que incluyan dependencias entre jugadas sucesivas, explorando cómo el resultado de una jugada anterior puede influir en las decisiones y probabilidades de la siguiente.

En conclusión, el modelo *XG Boost* se posiciona como la opción más efectiva para la predicción de cuartas oportunidades, mientras que la Regresión Logística proporciona una alternativa útil para escenarios en los que la interpretabilidad es clave. Estos resultados abren la puerta a un trabajo futuro enfocado en modelos bayesianos y métodos de cuantificación de la incertidumbre, con el fin de mejorar la aplicabilidad y robustez de los modelos predictivos en contextos deportivos complejos.

4.2 *Trabajo futuro*

El trabajo evidenció ciertas limitantes del proyecto. La primera tiene que ver con incertidumbre, hasta el alcance del proyecto se trataron de generar los mejores conjuntos de hiperparámetros sin embargo métodos de UQ pueden ayudar a tener una idea más clara de qué tanto del error proviene del ruido propio de la data y qué del ruido del modelo. Si tener un modelo más robusto fuera el objetivo los análisis basados en UQ pueden ayudar a saber si se necesita hacer otra ronda de fine tuning o si (como se propone en los resultados) se eliminan observaciones ruidosas, sabiendo que cuando el modelo evalue observaciones *OoD* [14] pues serán observaciones que no serán correctamente (dentro de lo que cabe) calsificadas.

Otra de las ideas que surgieron durante el desarrollo del proyecto fue que, en realidad, las cuartas oportunidades no son eventos independientes, sino que son probabilidades condicionadas a los resultados de las oportunidades previas. La premisa base de esta idea es: ¿qué pasa si la probabilidad de éxito de una jugada **n** está condicionada por el resultado de la jugada previa? Es decir, todo es parte de un sistema Bayesiano donde partimos de una creencia inicial (dada por un enfoque frecuentista) que después cambia en función de la jugada $\mathbf{n-1}$. Para ilustrarlo, imagine usted que el tiro anterior le dio (o le quitó) confianza; quizás el resultado del tiro anterior sí influye en el devenir del siguiente tiro. Algo así podría ser un camino para mejorar los resultados del modelo (aunque claramente es una hipótesis).

Siguiendo la línea del modelo Bayesiano y para explorar representaciones (predicciones) más interpretables, en lugar de un evento binario, se podría investigar el uso de un modelo neuronal Bayesiano [15] [1]

[1] En términos simples, una Red Neuronal Bayesiana es una red neuronal que genera distribuciones de probabilidad sobre sus predicciones. Esto permite tener una medida de incertidumbre en lugar de un resultado 0 o 1, mejorando la interpretabilidad y la toma de decisiones en comparación con los modelos binarios.

5 Apéndices

5.1 Documentos sobre la implementación del modelo

5.1.1 Pipeline de transformación de datos

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder, RobustScaler
import config
import logging
import os

logging.basicConfig(filename=config.LOGGING_PATH + 'data_preparation.log', level=logging.INFO)

def load_data(file_path):
    """
    Load data from a pickle file.
    """
    try:
        data = pd.read_pickle(file_path).drop(['yardline_group', 'play_type'], axis = 1)
        logging.info(f"Data loaded successfully from {file_path}")
        return data
    except Exception as e:
        logging.error(f"Error loading data: {e}")
        raise

def preprocess_data(data):
    """
    Preprocess the data by encoding categorical variables and scaling numerical variables.
    """
    try:
        categorical_variables = config.DATA_PARAMS['categorical_variables']
        numerical_variables = config.DATA_PARAMS['numerical_variables']

        # Encode categorical variables
        encoder = OneHotEncoder(sparse_output=False)
        encoded_data = encoder.fit_transform(data[categorical_variables])
        encoded_df = pd.DataFrame(encoded_data, columns=encoder.get_feature_names_out(categorical_variables))
        data = data.drop(categorical_variables, axis=1).reset_index(drop=True)
        data = pd.concat([data, encoded_df], axis=1)

        # Scale numerical variables
```

```
        scaler = RobustScaler()
        scaled_data = scaler.fit_transform(data[numerical_variables])
        scaled_df = pd.DataFrame(scaled_data, columns=numerical_variables)
        data[numerical_variables] = scaled_df

        logging.info("Data preprocessing completed successfully")
        return data
    except Exception as e:
        logging.error(f"Error in data preprocessing: {e}")
        raise

def feature_importance_reduction(data):
    """
    Delete not important features according to regression parameters
    """
    try:
        raws_to_drop = ['no_huddle', 'half_seconds_remaining', '
             play_location_right', 'play_location_left',
                        'play_location_middle', 'shotgun', '
                             posteam_type_away', 'posteam_type_home']
        data = data.drop(raws_to_drop, axis = 1)

        logging.info("Feature importance completed successfully")
        return data
    except Exception as e:
        logging.error(f"Error in feature deleting: {e}")
        raise

def save_preprocessed_data(data, file_path):
    """
    Save preprocessed data to a pickle file.
    """
    try:
        data.to_pickle(file_path)
        logging.info(f"Preprocessed data saved to {file_path}")
    except Exception as e:
        logging.error(f"Error saving preprocessed data: {e}")
        raise

if __name__ == "__main__":
    os.makedirs(config.LOGGING_PATH, exist_ok=True)
    raw_data = load_data(config.DATA_PATH)
    preprocessed_data = preprocess_data(raw_data)
    feat_importance_data = feature_importance_reduction(preprocessed_data)
    save_preprocessed_data(feat_importance_data, config.DATA_PATH.replace('.
         pkl', '_preprocessed.pkl'))
```

Listing 5.1: Data Transformation Pipeline

5.1.2 *Configuración de rutas y parámetros*

```

# Configuration file for paths and parameters
DATA_PATH = "data/fourth_down_data.pkl"
SAVED_MODEL_PATH = "models/"
LOGGING_PATH = "logs/"

# Parameters for model training
MODEL_PARAMS = {
```

```
    "LogisticRegression": {
        "penalty": "l2",
        "C": 0.25,
        "solver": "liblinear",
        "random_state": 123
    },
    "RandomForestClassifier": {
        "n_estimators": 50,
        "max_depth": 4,
        "min_samples_split": 0.1,
        "criterion": "gini",
        "bootstrap": True,
        "random_state": 123
    },
    "XGBClassifier": {
        "learning_rate": 0.25,
        "max_depth": 4,
        "n_estimators": 100,
        "random_state": 123
    },
    "NeuralNetwork": {
        "epochs": 50,
        "batch_size": 32,
        "input_dim": None  # to be set dynamically
    }
}

# Parameters for data processing
DATA_PARAMS = {
    "categorical_variables": ['posteam_type', 'game_half', 'play_location',
         'play_subtype'],
    "numerical_variables": ['yardline_100', 'half_seconds_remaining', '
         ydstogo', 'ydsnet', 'score_differential'],
    "binary_variables": ['goal_to_go', 'shotgun', 'no_huddle', 'qb_dropback'
         ],
    "target_variable": 'fourth_down_converted'
}
```

Listing 5.2: Configuration file for paths and parameters

5.2 *Archivos de registro*

En esta sección se muestran los *log files* con los resultados de todos los entrenamientos a los que se sometió el modelo.

5.2.1 *Registros de modelos no neuronales*

```
    INFO:root:Preprocessed data loaded successfully from data/fourth_down_data.pkl
    INFO:root:Data split into training and testing sets
    INFO:root:Results before feature importance transformation
    INFO:root:LogisticRegression trained successfully
    INFO:root:Model evaluation completed with Accuracy: 0.7142857142857143,
Recall: 0.725, F1 Score: 0.7131147540983607
    INFO:root:LogisticRegression saved successfully
```

```
    INFO:root:Results before feature importance transformation
& fine tuning
    INFO:root:RandomForestClassifier trained successfully
    INFO:root:Model evaluation completed with Accuracy: 0.6761904761904762,
Recall: 0.775, F1 Score: 0.7010050251256281
    INFO:root:RandomForestClassifier saved successfully
    INFO:root:Results before feature importance transformation
& fine tuning
    INFO:root:XGBClassifier trained successfully
    INFO:root:Model evaluation completed with Accuracy: 0.780952380952381,
Recall: 0.8, F1 Score: 0.7815468113975577
    INFO:root:XGBClassifier saved successfully
    INFO:root:Preprocessed data loaded successfully from data/fourth_down_data.pkl
    INFO:root:Data split into training and testing sets
    INFO:root:Results after feature importance transformation
    ERROR:root:Error training LogisticRegression: could not convert
string to float: 'home'
    INFO:root:LogisticRegression trained successfully
    INFO:root:Results after feature importance transformation
    INFO:root:Model evaluation completed with Accuracy: 0.7956600361663653,
Recall: 0.8293515358361775, F1 Score: 0.8113522537562604
    INFO:root:LogisticRegression saved successfully
    INFO:root:Results after feature importance transformation
& fine tuning
    INFO:root:RandomForestClassifier trained successfully
    INFO:root:Model evaluation completed with Accuracy: 0.7522603978300181
    INFO:root:RandomForestClassifier saved successfully
    INFO:root:Results after feature importance transformation
& fine tuning
    INFO:root:XGBClassifier trained successfully
    INFO:root:Model evaluation completed with Accuracy: 0.8553345388788427
    INFO:root:XGBClassifier saved successfully
```

5.2.2 *Registros de modelos neuronales*

```
    INFO:root:Results before feature importance transformation
    INFO:root:Neural network trained successfully
    INFO:root:Results before feature importance transformation
    INFO:root:Neural network trained successfully
    INFO:root:Model evaluation completed with Loss: 0.5121899247169495,
Accuracy: 0.7469387650489807, Recall: 0.7138888835906982
    INFO:root:NeuralNetwork saved successfully
    INFO:root:Preprocessed data loaded successfully from data/fourth_down_data_preprocessed.pkl
    INFO:root:Results after feature importance transformation
```

```
    INFO:root:Neural network trained successfully
    INFO:root:Model evaluation completed with Loss: 0.3988756239414215,
Accuracy: 0.8155515193939209, Recall: 0.80887371301651
    INFO:root:NeuralNetwork saved successfully
```

Bibliografía

[1] L. Fox, "How the nfl uses analytics, according to the lead analyst of a super bowl champion," 2021.

[2] NFL.

[3] L. Carl and R. Baldwin, "Nflfastr: Play-by-play data for nfl games." https://github.com/nflverse/nflfastR, 2023. Accessed: 2024-09-21.

[4] I. M. Peláez, "Modelos de regresión: lineal simple y regresión logística," *Revistaseden.org*, 2006.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] J. Mingers, "An empirical comparison of pruning methods for decision tree induction," 1989.

[7] J. S. B Ravi Kiran, "Cost-complexity pruning of random forests," 2017.

[8] C. Sheppard, *Tree-based Machine Learning Algorithms*. Clinton Shppard, 2019.

[9] S. with Josh Starmer, "Random forests part 1- building, using and evaluating," '2018'.

[10] G. Developers, "Introduction to gradient boosted decision trees (gbdt)."

[11] R. Santhanam, S. Raman, S. Banerjee, and N. Uzir, "Experimenting xgboost algorithm for prediction and classification of different datasets," *International Journal of Control Theory and Applications*, 2016.

[12] A. W. SERVICES, "Funcionamiento de xgboost."

[13] A. C. Gonzalo Martínez-Munoz, Candice Bentejac, "A comparative analysis of xgboost," 2019.

[14] I. P. de Jong, A. I. Sburlea, and M. Valdenegro-Toro, "How disentangled are your classification uncertainties?," 2024.

[15] V. Mullachery, A. Khera, and A. Husain, "Bayesian neural networks," 2018.