INFORME FINAL – PROYECTO CIENCIA DE DATOS APLICADA

Clasificador de jugadas de la NFL en carrera o pase

Rubén Franco & María Reyes

Preparación de datos (20%)

Datos Utilizados

- **Fuente:** https://www.kaggle.com/competitions/nfl-big-data-bowl-2025/data.
- **Características principales:** 'playId', 'gameId', 'quarter', 'down', 'yardsToGo', 'gameClock', 'preSnapHomeScore' ,'preSnapVisitorScore', 'offenseFormation', 'isDropback', 'pff_runPassOption', 'receivingYards', 'passigYards', 'runningYards'.
- **Tamaño del conjunto de datos:** 16240 registros de información de los jugadores que realizaron las jugadas y 16124 registros de jugadas realizadas.
- **Preprocesamiento:** Tokenización de variables categóricas (one hot), escalado o normalización de características numéricas, y manejo de valores faltantes, conversión de variables no numéricas a numéricas, eliminación de valores duplicados y merge entre los dos sets de datos para complementar las características necesarias para el entrenamiento de los modelos.

Estrategia de validación y selección del modelo (10%)

División de los datos:

- División el conjunto en entrenamiento (80%), y prueba (20%).

Selección del modelo:

- Uso de métricas como F1-score (para balance entre precisión y recall) y AUC-ROC (para distinguir entre clases).
- Evaluación tanto Árboles de Decisión como Random Forest.

Hiperparámetros:

- Ajuste con técnicas como GridSearch o RandomizedSearchCV.

Construcción y evaluación del modelo (20%)

En este proyecto, se entrenaron dos modelos de machine learning, Random Forest y Árbol de Decisión, para predecir el tipo de jugada (carrera o pase) en juegos de la NFL. El objetivo principal

era identificar patrones en los datos históricos y desarrollar un sistema predictivo que apoye la toma de decisiones en tiempo real.

Modelo de Árbol de Decisión

- **Búsqueda de hiperparámetros:**

```
param_grid={'classifier__criterion': ['gini', 'entropy'],
            'classifier__max_depth': [3, 5, 7, 10, None],
            'classifier__min_samples_leaf': [1, 2, 4],
            'classifier__min_samples_split': [2, 5, 10]}
```

- **Ventajas:** Interpretable y rápido de entrenar.

Las matrices de confusión presentadas reflejan el desempeño de un modelo de clasificación en los conjuntos de entrenamiento y de prueba. Para el conjunto de entrenamiento, podemos ver la clase 0 Negativa, la cual clasifica los verdaderos negativos con valores como 4253 casos correctamente clasificados. En cuando a los Falsos Positivos, existen 36 casos incorrectamente clasificados dentro de la clase 1. Lo anterior muestra una tasa de precisión alta para la clase 0. Por su parte, la clase 1 hay 4106 casos correctamente clasificados como verdaderos positivos y 301 casos incorrectamente clasificados como falsos negativos.

Para el conjunto de prueba en la clase 0 existen 1072 casos correctamente clasificados como verdaderos negativos y 7 casos incorrectamente clasificados como clase 1 en falsos positivos.

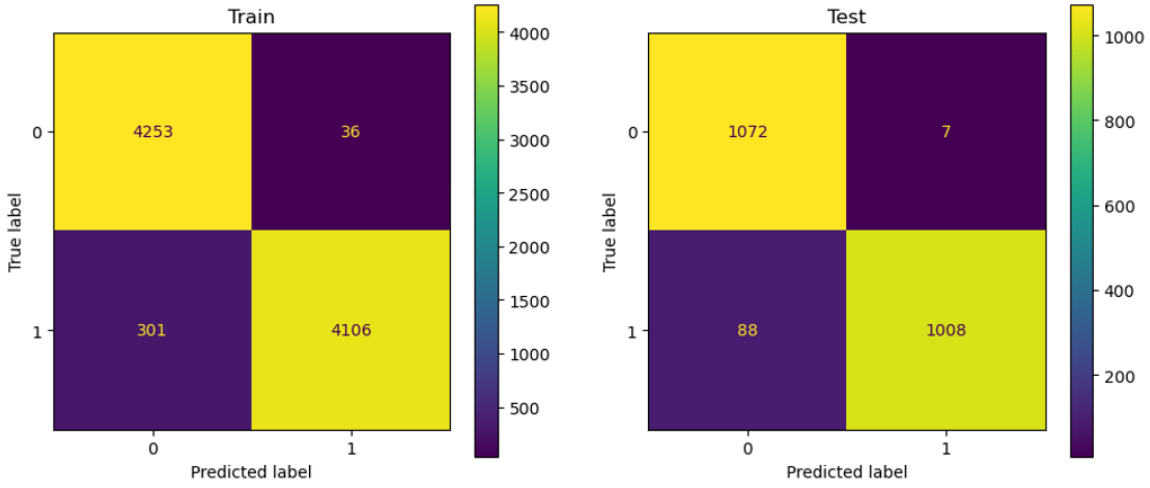

Figura 1. Matriz de confusión de los árboles de decisión para los conjuntos de entrenamiento y prueba.

```
Accuracy:
- Train/Val: 0.9612465501379944
- Test: 0.9563218390804598

Precision:
- Train/Val: 0.9913085465958474
- Test: 0.993103448275862

Recall:
- Train/Val: 0.9316995688677104
- Test: 0.9197080291970803

F1:
- Train/Val: 0.9605801848169376
- Test: 0.9549976314542871
```

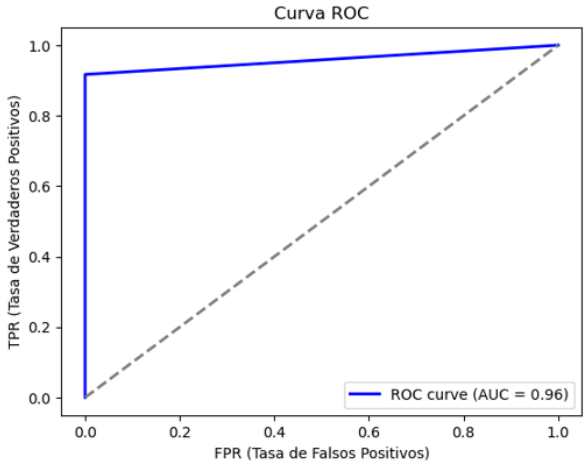

Figura 2. Métricas obtenidas para el modelo con Árbol de decisión.

Modelo de Random Forest

- **Búsqueda de hiperparámetros:**

  param_grid={'classifier__class_weight': ['balanced', None],
  'classifier__max_depth': [3, 4, 5, 6],
  'classifier__n_estimators': [50, 100, 150]}
- **Ventajas:** Reduce el sobreajuste y mejora la precisión mediante la agregación de predicciones de múltiples árboles.

Las matrices de confusión con el modelo de Random Forest representa para el conjunto de entrenamiento en la clase negativa que hay 4289 casos correctamente clasificados como clase 0 con verdaderos negativos y 0 casos mal clasificados en falsos positivos. Para la clase 1 existen 4099 casos correctamente clasificados como verdaderos positivos mientras que para los falsos negativos son al menos 308 casos están mal clasificados. Para el conjunto de prueba hay al menos 1079 casos correctamente clasificados, mientras que los mal clasificados son 0. En la clase positiva 1005 casos correctamente clasificados como verdaderos positivos y 91 casos mal clasificados como falsos negativos.

El modelo tiene una precisión sobresaliente para ambas clases, especialmente para la negativa, donde no comete errores en ninguno de los conjuntos de datos, tanto de entrenamiento como el de prueba.

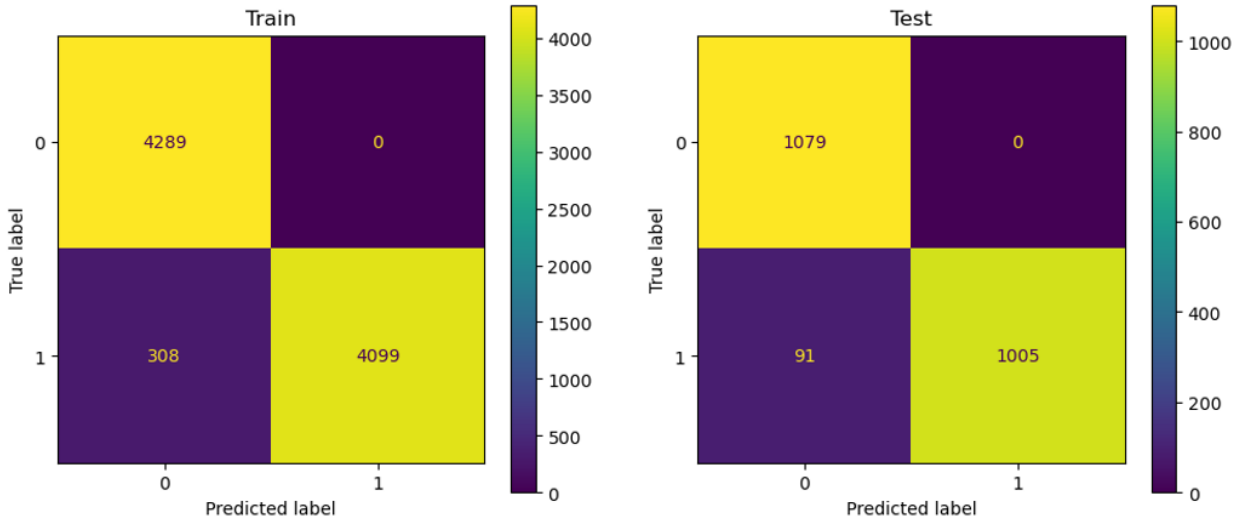

Figura 3. Matriz de confusión de Random Forest para los conjuntos de entrenamiento y prueba.

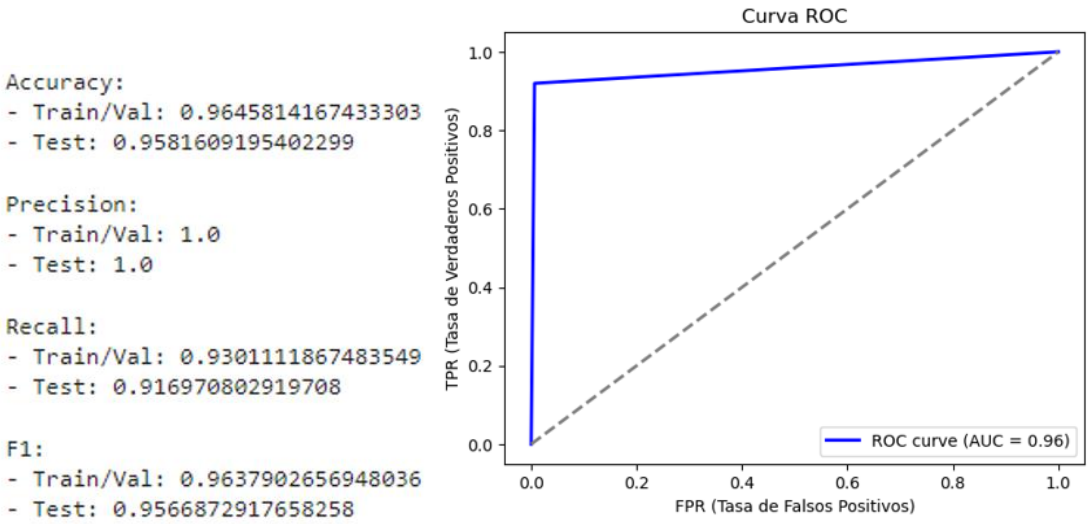

Figura 4. Métricas obtenidas para el modelo con Random Forest.

Los resultados proporcionan métricas para evaluar el desempeño de dos modelos en los conjuntos de datos, tanto el de entrenamiento como el de prueba. Para el modelo de árbol de decisión se clasifica correctamente un alto porcentaje de las instancias, tanto en entrenamiento como en prueba, lo que indica un desempeño general. En cuanto a la precisión se identifica correctamente la mayoría de las predicciones positivas y baja la tasa de falsos positivos y un recall

el modelo identifica correctamente una alta proporción de las instancias positivas con una caída en el conjunto de prueba indicando que podría haber falsos negativos.

Para el modelo de Random Forest el rendimeinto es ligeramente superior en precisión comparado con el modelo de entrenamiento y de prueba.la precisión del modelo no comete falsos positivos, lo que significa que cada instancia predicha como positiva es realmente positiva. El recall indica que el modelo sigue perdiendo algunas instancias positivas.

Construcción del producto de datos (20%)

IMPLEMENTACIÓN DE LA APLICACIÓN

1. Resumen del Proyecto

Este proyecto consiste en el desarrollo de una aplicación web que permite predecir jugadas en la NFL (Rush o Pase) utilizando un modelo de regresión, además de implementar un sistema para interactuar con el modelo de Machine Learning a través de una interfaz web moderna. La aplicación se ha dividido en dos partes principales:

- **Backend (Flask)**: Servidor que gestiona la comunicación con el modelo de Machine Learning y maneja las predicciones.
- **Frontend (React)**: Interfaz de usuario que permite a los usuarios ingresar datos y recibir predicciones del modelo.

Objetivo:

- Reducir los tiempos de predicción y proporcionar un sistema fácil de usar para los usuarios interesados en las jugadas de la NFL.

2. Desarrollo del Backend (Flask)

El backend de la aplicación está construido utilizando el framework **Flask** para Python. Este servidor se encarga de recibir las solicitudes de la interfaz de usuario, procesar las entradas y devolver las predicciones realizadas por el modelo de Machine Learning.

Pasos realizados:

1. **Instalación de Flask**: Se instalaron las dependencias necesarias para ejecutar Flask y trabajar con el modelo de Machine Learning.

a. Comando:

b. pip install flask joblib

2. **Creación de la API**:

a. Se desarrolló una API en Flask que expone un endpoint /predict que acepta una solicitud POST con los parámetros necesarios (por ejemplo, características de una jugada en la NFL).

b. El modelo de Machine Learning se carga usando joblib y se realiza la predicción en el servidor.

3. **Ejecución del servidor**:

a. El servidor de Flask se ejecuta usando python app.py, lo que permite que se comunique con el frontend a través de HTTP.

3. Desarrollo del Frontend (React)

El frontend se desarrolló usando **React** para crear una interfaz interactiva que permite a los usuarios ingresar los datos de las jugadas y recibir las predicciones del modelo.

Pasos realizados:

1. **Configuración de React**:

a. Se creó una aplicación de React usando create-react-app.

b. Se configuró una interfaz donde los usuarios pueden ingresar los parámetros necesarios para realizar la predicción.

2. **Integración con el Backend**:

a. Se utilizó la librería axios para enviar las solicitudes HTTP al servidor Flask y obtener las predicciones.

b. Se configuró un formulario interactivo donde los usuarios pueden introducir los valores para las características y recibir la predicción.

3. **Interfaz de Usuario**:

a. Se diseñó una interfaz simple y clara donde el usuario puede ingresar los parámetros, como características de jugadas, para obtener la predicción (si será un rush o un pase).

4. Automatización y Ejecución del Proyecto

Para facilitar la ejecución del proyecto, se creó un archivo .bat en Windows que permite iniciar tanto el servidor de Flask como la aplicación de React con un solo clic.

Archivo .bat:

El archivo .bat automatiza los siguientes pasos:

1. Navegar al directorio del backend y ejecutar el servidor Flask.
2. Navegar al directorio del frontend y ejecutar la aplicación React.

Este archivo facilita la ejecución del proyecto sin necesidad de abrir múltiples terminales.

5. Consideraciones Adicionales

- **Dependencias del Proyecto**:
  - Para el backend: Flask, joblib, entre otras.
  - Para el frontend: React, axios y otras librerías necesarias.

6. Próximos Pasos

- **Despliegue de la aplicación**: Configuración de la aplicación para producción, incluyendo la implementación en un servidor web (por ejemplo, Heroku, AWS, etc.).
- **Mejoras en la interfaz**: Personalización y diseño más sofisticado, adaptado a un tema de NFL, utilizando logos, colores y otros elementos gráficos.
- **Optimización**: Mejorar la interacción con el modelo de Machine Learning, haciendo que las predicciones sean más rápidas y precisas.

EJECUCIÓN DE LA APLICACIÓN

1. Requisitos previos

Antes de comenzar, asegúrate de tener lo siguiente instalado en tu computadora:

1. **Python**:
   a. Debe estar instalado y accesible desde la línea de comandos.
   b. Verifique con:

python --version

2. **Node.js y npm**:
   a. Necesarios para ejecutar la interfaz React.
   b. Verifique con:

node --version

npm --version

3. **Clona el repositorio del proyecto**:

Debido a que los ejecutables del repositorio son archivos .bat, es probable que al descargar el .zip del repositorio el computador no permita la descarga. Por eso es recomendable clonar el repositorio en un directorio local.

- Ejecutar el siguiente comando en la línea de comandos:

git clone https://github.com/RubenFranc/RushPassPredictor-ProyectoFinalCDA.git

2. Instalación de dependencias

Sigue estos pasos para instalar todas las dependencias necesarias:

1. **Ubícate en la carpeta raíz del proyecto**:

a. Abre el explorador de archivos y dirígete a la carpeta donde descargaste el repositorio.

2. **Ejecuta el archivo .bat de instalación**:

a. Haz doble clic en el archivo .bat que automatiza la instalación de dependencias (installDependencies.bat).

b. Este archivo realizará:

i. La instalación de las dependencias de Python desde requirements.txt.

ii. La instalación de las dependencias de React mediante npm install.

3. **Confirma que la instalación se completó**:

a. Verifica que no haya errores en la ventana que se abre durante la instalación.

b. Al finalizar, las dependencias estarán listas.

3. Ejecución de la aplicación

Una vez instaladas las dependencias, sigue estos pasos para ejecutar la aplicación:

1. **Ejecuta el archivo .bat de ejecución**:

a. Haz doble clic en el archivo .bat que inicia la aplicación (RushPassPredictor.bat).

b. Este archivo:

i. Iniciará el servidor Flask para el backend.

ii. Iniciará el servidor de React para el frontend.

2. **Abre la aplicación en tu navegador**:

a. El archivo .bat configurará automáticamente las rutas y abrirá una pestaña en tu navegador.

3. **Interactúa con la aplicación**:

a. Usa la interfaz React para enviar datos y obtener predicciones desde el backend Flask.

4. Cierre de la aplicación

1. **Detén la ejecución**:

a. Cierra las ventanas de terminal que se abrieron para los servidores de Flask y React.

2. **Reinicio en el futuro**:

a. Si necesitas ejecutar nuevamente la aplicación, simplemente repite el paso de ejecución con el archivo RushPassPredictor.bat.

Si lo desea, en el siguiente enlace de YouTube se encuentra un tutorial en el que se llevan a cabo los pasos necesarios para la ejecución de la aplicación: https://youtu.be/iWn2VPxWnOE

Retroalimentación por parte de la organización (5%)

Se usaron datos públicos alojados en Kaggle, no se trabajó con ninguna empresa, por lo que no se obtuvo retroalimentación.

Conclusiones (15%)

- **Desempeño del Modelo**: Ambos modelos (Árbol de Decisión y Random Forest) demostraron métricas excelentes en la predicción de jugadas (carrera o pase). El Random Forest destacó por su estabilidad y capacidad para generalizar.
- **Contribución de las Variables**: Las variables clave, tanto de tracking como de análisis avanzado (*pff_*), aportaron significativamente al modelo, especialmente aquellas relacionadas con posiciones iniciales y movimiento de los jugadores.
- **Aplicabilidad**: Estos modelos ofrecen un marco sólido para análisis predictivo en tiempo real, con potencial de integrarse en herramientas de estrategia para equipos de la NFL.
- **Próximos Pasos**: Incluir datos adicionales, como situaciones de juego o contexto histórico, para mejorar la robustez del modelo.