TITULO PROYECTO: Tráfico Portuario marítimo en Colombia

INTEGRANTES:

- **SANTIAGO MENDOZA MUÑOZ**
- **MIGUEL LEGARDA CARRILLO**
- **CAMILO ANDRES ARMENTA**

1. ENTENDIMIENTO DEL NEGOCIO

1.1 DESCRIPCIÓN DEL NEGOCIO

La Dirección General Marítima (DIMAR) y los operadores portuarios de Colombia supervisan y gestionan el tráfico de carga en los puertos marítimos del país. Estos puertos pueden operar bajo naturaleza pública o privada, lo que influye en la logística, la regulación y la participación del Estado. El registro del tipo de tráfico, junto con detalles como tipo de carga, operador, ubicación y volumen movilizado, permite un análisis estratégico de la actividad portuaria para la planeación y el desarrollo del comercio exterior.

1.2 DESCRIPCIÓN DEL PROBLEMA

En la actualidad, no se cuenta con un mecanismo automatizado que permita anticipar si un nuevo registro de tráfico portuario será de tipo público o privado. Esta clasificación es fundamental para la toma de decisiones operativas, regulatorias y de inversión, y su ausencia limita la eficiencia en la gestión portuaria y la planificación de infraestructura.

1.3 OBJETIVOS DE LA MINERÍA

Objetivo general: Construir un modelo de clasificación que permita predecir si el tráfico portuario es de tipo público o privado, a partir de variables históricas como tipo de carga, puerto, operador, ubicación y volumen movilizado.

Objetivos específicos:

- Identificar las variables más relevantes que influyen en la naturaleza del tráfico (público o privado).
- Entrenar y validar modelos de clasificación supervisado con buenas métricas de desempeño.
- Analizar los patrones históricos para comprender las diferencias operativas entre el tráfico público y privado.
- Facilitar una herramienta analítica para apoyar decisiones logísticas y regulatorias en el sector marítimo.

1.4 DISEÑO DE SOLUCIÓN

Problema	Tipo de Minería	Tipo de aprendizaje	Requerimiento datos	Métodos	Evaluación
Predecir si un tráfico portuario será público o privado	Clasificación	Supervisado	Variables como puerto, operador, tipo de carga, ubicación, volumen, sentido del tráfico.	- Regresión logística - Máquina de soporte vectorial (SVM) - (KNN) - Red neuronal (MLP) **Ensamble:** - Random Forest - Gradient Boosting - XGBoost	- Precisión - Recall - F1-score -Matriz confusión

Evaluación esperada:

- Se entrenarán al menos **4 algoritmos de aprendizaje supervisado**.
- Se aplicarán **3 técnicas de ensamble**.
- Se calcularán e interpretarán al menos **4 métricas de evaluación** por modelo.
- El modelo seleccionado debe alcanzar al menos un **85% de precisión**, y las demás métricas serán usadas para una evaluación más completa.

1.5 RECURSOS PARA CREACIÓN DEL MODELO Y PARA DESPLIEGUE

Hardware

- Computadora con 16 GB de RAM.
- Procesador i7
- Obtuvimos acceso a GPU para que el entrenamiento fuese más rápido.

Software

- Python
- Librerías: pandas, scikit-learn, xgboost, matplotlib, seaborn
- Entornos: Google Colab
- Para despliegue: Hicimos uso de streamlit

2. ENTENDIMIENTO DE LOS DATOS (Datos específicos del problema)

2.1 CICLO DE LOS DATOS:

- Generación: Los datos son generados mensualmente por la Dirección General Marítima (DIMAR) y los operadores portuarios, como parte del reporte nacional de tráfico marítimo.

- Almacenamiento: La información se almacena en bases de datos del portal datos.gov.co, con disponibilidad en formato CSV o mediante API.

- Modificación **(ruta)**: Los datos pueden ser consultados, transformados y enriquecidos localmente a través de scripts en Python. La ruta típica incluye limpieza, normalización y etiquetado de clase (público/privado).

- Periodicidad: Observamos que los datos se encuentran con una periodicidad mensualmente.

2.2 DICCIONARIO DE DATOS

Variable	Descripción	Tipo
zona_portuaria	Lugar donde se encuentra ubicado un puerto	Texto
sociedad_portuaria	Sociedad portuaria	Texto
tipo_servicio	Nombre de la clasificación del tipo de servicio	Texto
tipo_carga	Nombre de la clasificación del tipo de carga	Texto
exportación	Tráfico de carga de exportación en toneladas	Número
Importación	Tráfico de carga de importación en toneladas	Número
transbordo	Información de transbordo	Número
transito_internacional	Información de tránsito internacional	Número
Fluvial	Tránsito fluvial en toneladas	Número
Cabotaje	Tránsito cabotaje en toneladas	Número
movilizaciones_a_bordo	Carga movilizada a bordo	Número
transitoria	Variable adicional de tránsito no permanente	Número
anno_vigencia	Año en que se genera la información	Número
mes_vigencia	Mes en que se genera la información	Número

2.3 REGLAS DE CALIDAD

Variable	Regla calidad (valores válidos)
zona_portuaria	No nulo; texto válido
sociedad_portuaria	No nulo; texto válido
tipo_servicio	Valores válidos conocidos según catálogo oficial
tipo_carga	Valores válidos: Contenedores, Granel, etc.
exportacion	>= 0 toneladas
importacion	>= 0 toneladas
transbordo	>= 0 toneladas
transito_internacional	>= 0 toneladas
fluvial	>= 0 toneladas
cabotaje	>= 0 toneladas
movilizaciones_a_bordo	>= 0 toneladas
transitoria	>= 0 (SI APLICA)
anno_vigencia	Entre 2000 y el año actual
mes_vigencia	Entre 1 y 12

3. PREPARACIÓN DE DATOS

3.1 INTEGRACIÓN

En esta etapa se realizó la recolección e integración de los datos necesarios para el análisis del tráfico portuario marítimo en Colombia. La fuente principal fue el portal oficial de Datos Abiertos del Gobierno Colombiano (datos.gov.co), desde donde se descargó el conjunto de datos “Tráfico Portuario Marítimo en Colombia”.

El archivo fue importado directamente en el entorno de Google Colab utilizando la librería panda, permitiendo su carga y manipulación en formato tabular. Para facilitar su análisis, se estandarizaron los nombres de las columnas, convirtiéndolos a minúsculas y reemplazando espacios por guiones bajos, lo que mejora la legibilidad y evita errores en el procesamiento posterior. También se revisaron las primeras filas del dataset para validar que la importación y el formato fueran correctos.

Con esto, los datos quedaron integrados en un único DataFrame que servirá como base para las siguientes fases del proyecto.

3.1 SELECCIÓN DE VARIABLES

Durante esta fase se analizaron las 14 variables del conjunto de datos con el objetivo de conservar únicamente aquellas que fueran relevantes para el problema de clasificación: predecir si el tráfico portuario es de tipo **público o privado**.

Se evaluaron dos criterios principales:

- **Eliminación de variables irrelevantes**: no se encontraron columnas que funcionaran como identificadores únicos o datos sensibles (como nombres, cédulas, direcciones o teléfonos), por lo que no fue necesario descartar ninguna columna por esta razón.
- **Eliminación de variables redundantes**: se revisaron las columnas en busca de duplicidad de información (por ejemplo, si existiera "edad" y "año de nacimiento"). No se identificaron redundancias directas, pero se consideró la eliminación de la columna ***`transitoria, y anno_vigencia`*** ya que su significado operativo no estaba claro y podría no aportar información significativa al modelo.

Se conservó el resto de las variables, al considerar que cada una podría tener valor predictivo en la clasificación del tráfico. La selección inicial es flexible y podrá ajustarse tras realizar análisis de correlación y evaluación de importancia de variables en las etapas siguientes.

3.2 DESCRIPCIÓN ESTADÍSTICA

En esta etapa se realizó una exploración inicial del conjunto de datos con el objetivo de comprender su estructura general y comportamiento estadístico. Se dividieron las variables en dos grandes grupos: **categóricas** y **numéricas**.

Para las variables **categóricas** —como ***`zona_portuaria, sociedad_portuaria, tipo_servicio y tipo_carga`***— se analizaron sus frecuencias absolutas para identificar distribuciones desbalanceadas o categorías poco frecuentes. Además, se generaron gráficos de barras mediante la librería `seaborn`, lo cual facilitó la visualización clara de las distribuciones y permitió detectar posibles problemas como valores nulos, clases dominantes o categorías demasiado específicas.

Para las variables **numéricas**, se aplicó el método `describe()` de `pandas`, lo que permitió obtener medidas clave como media, mediana, desviación estándar, valor mínimo, percentiles y valor máximo. Esto ayudó a identificar rangos amplios, posibles valores extremos y escalas diferentes entre variables.

Adicionalmente, se instaló y preparó el entorno para usar la librería **`pandas-profiling`** (actualmente conocida como `ydata-profiling`), con el objetivo de generar un reporte estadístico automático y completo del conjunto de datos. Este reporte incluye estadísticas univariadas, análisis de correlación, detección de valores faltantes y alertas de calidad, lo que complementa y profundiza el análisis exploratorio realizado de forma manual.

3.3 LIMPIEZA DE ATÍPICOS

En esta etapa se buscó identificar y tratar valores atípicos (outliers) en las variables numéricas del conjunto de datos. Este tipo de valores pueden representar errores de digitación, registros anómalos o simplemente casos extremos válidos. Por esta razón, el proceso se realizó con especial cuidado, siguiendo tres pasos fundamentales:

1. **Identificación de atípicos**:
   Se utilizó el método del rango intercuartílico (IQR), que detecta valores atípicos ubicados por fuera de 1.5 veces el rango entre el primer y el tercer cuartil. Este análisis permitió identificar en qué variables existían posibles valores extremos.

Para tratarlos de manera prudente, no se eliminaron directamente, sino que se reemplazaron por valores nulos (NaN), dejando su imputación para la siguiente fase (3.5). Esta estrategia evita la pérdida de registros completos y permite un tratamiento más flexible y controlado de las anomalías en los datos.

2. **Visualización y evaluación**:
   Para complementar la detección estadística, se generaron gráficos de tipo boxplot para cada variable numérica. Esta visualización permitió identificar patrones y evaluar si los posibles atípicos ya estaban corregidos.

Este enfoque equilibrado garantizó que no se eliminaran datos importantes de forma arbitraria, manteniendo la calidad y representatividad del conjunto de datos para los distintos modelos de machine LearNing posteriores.

3.4 LIMPIEZA DE NULOS

En esta etapa se abordó el tratamiento de valores ausentes (nulos) presentes en el conjunto de datos, particularmente aquellos que surgieron como resultado del reemplazo de valores atípicos en la fase anterior. La presencia de datos faltantes puede afectar negativamente el rendimiento y estabilidad de los modelos de aprendizaje automático, por lo que es necesario imputarlos de forma adecuada.

Para realizar la imputación se adoptó una estrategia diferenciada según el tipo de variable:

- **Variables numéricas**: los valores nulos fueron reemplazados por la **media** de cada columna, con el fin de conservar el comportamiento central de la distribución sin introducir sesgos extremos.
- **Variables categóricas**: los nulos fueron imputados usando la **moda**, es decir, el valor más frecuente dentro de cada columna, lo que permite mantener la coherencia semántica de los datos categóricos.

Finalmente, se verificó que no quedaran valores faltantes en el conjunto de datos, garantizando así su integridad para las siguientes etapas del proceso de modelado.

3.5 = Creación de nuevas variables

Básicamente en este paso , se decidio no crear variables ya que no fue una necesidad objetiva en nuestro dominio de problema.

3.5 ANÁLISIS DE CORRELACIONES PARA REDUNDANCIA

En esta fase se analizó la matriz de correlación de Pearson entre las variables numéricas del conjunto de datos, con el objetivo de identificar posibles redundancias. La presencia de variables fuertemente correlacionadas (por ejemplo, con coeficientes mayores a 0.85 o menores a -0.85) podría indicar duplicidad de información, lo cual puede impactar negativamente la eficiencia del modelo.

Se utilizó un mapa de calor (heatmap) para visualizar la matriz de correlación, así como una búsqueda automática de pares de variables altamente correlacionadas. Como resultado, **no se encontraron pares de variables con correlaciones iguales o superiores a ±0.85**, por lo que **no fue necesario eliminar ninguna variable** en esta etapa.

Esto indica que cada variable numérica aporta información diferenciada y complementaria al modelo de clasificación, por lo que se conservaron todas para las siguientes fases.

- Que **ninguna variable numérica está fuertemente correlacionada con otra**.
- Por tanto, **no hay redundancia estadística** entre las variables.
- Podemos **conservar todas las variables numéricas**, ya que cada una aporta información independiente al modelo.

3.6 ANÁLISIS DE CORRELACIONES PARA IRRELEVANCIA (PREDICCIONES)

En esta etapa se analizó la **relevancia de las variables predictoras** respecto a la variable objetivo, con el propósito de eliminar aquellas que aportan poca o ninguna información útil para el modelo de clasificación.

Se utilizó como medida de dependencia la **ganancia de información**, calculada a partir del concepto de **entropía**. Esta técnica evalúa cuánto se reduce la incertidumbre sobre la variable objetivo cuando se conoce el valor de una variable predictora específica.

El procedimiento constó de los siguientes pasos:

1. Se definió la función de **entropía**, que mide la dispersión o incertidumbre de los valores posibles de la variable objetivo.
2. Se implementó la **información condicional**, que representa la entropía esperada de la variable objetivo condicionada por una variable predictora.
3. Se calculó la **ganancia de información** como la diferencia entre la entropía total y la condicional, para cada variable independiente del conjunto de datos.
4. Se listaron las variables con mayor y menor aporte informativo. Aquellas cuya ganancia fue muy baja (por ejemplo, inferior a 0.01) fueron consideradas irrelevantes para la predicción.

5. Finalmente, se eliminaron del conjunto de datos variables , ya que su aporte a la predicción de la variable objetivo fue mínimo.

3.7 PCA

En esta etapa se evaluó la necesidad de aplicar técnicas de reducción de dimensionalidad, como el Análisis de Componentes Principales (PCA), con el objetivo de simplificar el conjunto de variables manteniendo la mayor cantidad de información posible. Sin embargo, tras analizar las características del dataset se determinó que:

- El número total de variables era manejable, tanto en términos computacionales como de complejidad del modelo.
- Ya se había realizado un proceso riguroso de selección de variables irrelevantes mediante medidas de ganancia de información y análisis de correlación.
- No existía multicolinealidad fuerte entre las variables numéricas.
- El uso de PCA dificultaría la interpretación de los resultados, lo cual es un aspecto importante para este proyecto orientado a la toma de decisiones.

Por estas razones, se concluyó que **no era necesario aplicar técnicas de reducción de dimensionalidad en este caso**, y se continuó con el conjunto de variables originales tras su depuración.

3.8 TRANSFORMACIONES

En esta etapa se realizaron las transformaciones necesarias para preparar el conjunto de datos para su uso en modelos de aprendizaje automático, los cuales requieren que todas las variables estén en formato numérico.

Primero, se transformó la **variable objetivo** `tipo_servicio`, la cual indica si el tráfico portuario es público o privado. Esta variable es categórica binaria, por lo que se codificó utilizando **LabelEncoder**, asignando valores numéricos: `0` para "privado" y `1` para "público".

Posteriormente, se identificaron las demás **variables categóricas** presentes en el conjunto de datos, y se transformaron mediante **One-Hot Encoding**. Esta técnica convierte cada categoría en una columna binaria (0 o 1), lo que permite representar adecuadamente variables categóricas sin

introducir un orden artificial entre sus categorías. Se utilizó la opción `drop_first=True` para evitar colinealidad entre las nuevas columnas generadas.

Estas transformaciones garantizaron que el dataset quedara completamente en formato numérico, compatible con algoritmos de clasificación como regresión logística, árboles de decisión, SVM o redes neuronales. Finalmente, el conjunto de datos preparado fue guardado como archivo `.csv`, listo para ser utilizado en la fase de modelado.

3.9 BALANCEO (CLASIFICACIÓN)

En esta fase abordamos el problema del **desbalanceo de clases** en la variable objetivo `tipo_servicio`, la cual indica si el tráfico portuario marítimo es de tipo público o privado. El análisis preliminar reveló que existía una clara diferencia en la cantidad de registros entre ambas clases, lo cual podía generar un sesgo en el modelo, favoreciendo la clase mayoritaria.

Para resolver este problema se utilizó la técnica de **SMOTE (Synthetic Minority Oversampling Technique)**, una estrategia de sobremuestreo que consiste en generar nuevos registros sintéticos de la clase minoritaria, basados en sus vecinos más cercanos. Esta técnica permite equilibrar el conjunto de entrenamiento sin perder información ni recurrir a la duplicación directa de datos.

El proceso se llevó a cabo de forma controlada y en los siguientes pasos:

1. Se visualizó la distribución original de clases para confirmar el desbalance.
2. Se codificaron las variables categóricas mediante `LabelEncoder`, incluyendo la variable objetivo, para garantizar que todos los datos fueran numéricos, como lo requiere SMOTE.
3. Se dividió el conjunto de datos en entrenamiento y prueba antes del balanceo, de manera que solo se aplicara SMOTE sobre los datos de entrenamiento, evitando así un sesgo en la evaluación del modelo.
4. Se aplicó SMOTE usando un número ajustado de vecinos (`k_neighbors`) adaptado a la cantidad de muestras de la clase minoritaria, lo cual mejoró la robustez del balanceo.
5. Finalmente, confirmamos numéricamente que el conjunto de entrenamiento quedó balanceado.

Adicionalmente, se implementó una función que agrupa y devuelve todos los conjuntos de datos preparados (`X_train`, `y_train`, `X_test`, `y_test`), junto con los codificadores utilizados, lo que facilita el acceso estructurado a los datos para las etapas posteriores de modelado.

Este proceso garantizo que el modelo tenga la misma oportunidad de aprender tanto de registros públicos como privados, mejorando la generalización y la equidad en la predicción.

LUEGO DE HABER TERMINADO CON EL PROCESAMIENTO DE DATOS PROCEDEMOS AL MODELAMIENTO.

4. MODELAMIENTO, EVALUACIÓN E INTERPRETACIÓN

4.1 CONFIGURACIÓN MÉTODOS DE MACHINE LEARNING

Los modelos considerados inicialmente fueron:

1. Regresión Logística (Logistic Regression): Un modelo lineal fundamental, utilizado como línea base por su interpretabilidad y eficiencia computacional. Se configuró con un max_iter de 1000 para asegurar la convergencia.
2. K-Vecinos Más Cercanos (KNN): Un modelo basado en instancias, que clasifica nuevas muestras basándose en la mayoría de sus vecinos más cercanos. Se utilizó la configuración por defecto de KNeighborsClassifier.
3. Máquinas de Vectores de Soporte (SVM): Un modelo potente que busca el hiperplano óptimo para separar las clases. Se configuró con probability=True para permitir la obtención de probabilidades de predicción.
4. Redes Neuronales (Perceptrón Multicapa - MLPClassifier): Un modelo no lineal capaz de aprender representaciones complejas de los datos. Se configuró con max_iter=1000 y un random_state=42 para reproducibilidad.
5. Bosques Aleatorios (Random Forest): Un método de ensamble basado en árboles de decisión, conocido por su robustez y buen rendimiento. Se configuró con random_state=42.
6. Gradient Boosting (GradientBoostingClassifier): Otro método de ensamble basado en árboles, que construye los árboles de forma secuencial para corregir los errores de los anteriores. Se configuró con random_state=42.
7. XGBoost (XGBClassifier): Una implementación optimizada y escalable de Gradient Boosting, ampliamente utilizada por su alto rendimiento. Se configuró con use_label_encoder=False, eval_metric='logloss', y random_state=42.

Todos estos modelos fueron implementados utilizando las librerías scikit-learn y XGBoost en Python. La evaluación inicial de cada modelo se realizó mediante un proceso de validación cruzada estratificada (con *k* folds, donde *k* fue 10 sobre el conjunto de entrenamiento preprocesado y remuestreado (X_train_resampled, y_train_resampled). Las métricas de calidad recopiladas durante la validación cruzada para cada modelo fueron: Accuracy, Precision, Recall y F1-Score

(ponderado o para la clase de interés, según el objetivo). El F1-Score se consideró la métrica principal para la comparación cuantitativa del rendimiento.

4.2 ANALISIS DE MEDIDAS DE CALIDAD

Tras la ejecución de la validación cruzada para el conjunto inicial de modelos configurados, se obtuvieron las siguientes métricas de rendimiento promedio:

Modelo	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.999686	0.999374	1.000000	0.999687
KNN	0.998954	0.999582	0.998326	0.998953
SVM	0.999686	0.999374	1.000000	0.999687
Neural Network	0.999686	0.999374	1.000000	0.999687
Random Forest	0.999686	0.999374	1.000000	0.999687
Gradient Boosting	0.999686	0.999374	1.000000	0.999687
XGBoost	0.999686	0.999374	1.000000	0.999687

¡Claro! Basándome en la información que me has proporcionado y en nuestra conversación, aquí tienes una propuesta para completar esas secciones de tu documentación CRISP-DM.

4.1 CONFIGURACIÓN MÉTODOS DE MACHINE LEARNING

En la fase inicial de modelado, se seleccionó un conjunto diverso de algoritmos de Machine Learning con el objetivo de identificar aquellos con mayor potencial para resolver el problema de clasificación planteado. La selección abarcó diferentes paradigmas de aprendizaje para asegurar una exploración amplia del espacio de soluciones:

Los modelos considerados inicialmente fueron:

1. **Regresión Logística (`Logistic Regression`):** Un modelo lineal fundamental, utilizado como línea base por su interpretabilidad y eficiencia computacional. Se configuró con un `max_iter` de 1000 para asegurar la convergencia.

2. **K-Vecinos Más Cercanos (KNN):** Un modelo basado en instancias, que clasifica nuevas muestras basándose en la mayoría de sus vecinos más cercanos. Se utilizó la configuración por defecto de KNeighborsClassifier.
3. **Máquinas de Vectores de Soporte (SVM):** Un modelo potente que busca el hiperplano óptimo para separar las clases. Se configuró con probability=True para permitir la obtención de probabilidades de predicción.
4. **Redes Neuronales (Perceptrón Multicapa - MLPClassifier):** Un modelo no lineal capaz de aprender representaciones complejas de los datos. Se configuró con max_iter=1000 y un random_state=42 para reproducibilidad.
5. **Bosques Aleatorios (Random Forest):** Un método de ensamble basado en árboles de decisión, conocido por su robustez y buen rendimiento. Se configuró con random_state=42.
6. **Gradient Boosting (GradientBoostingClassifier):** Otro método de ensamble basado en árboles, que construye los árboles de forma secuencial para corregir los errores de los anteriores. Se configuró con random_state=42.
7. **XGBoost (XGBClassifier):** Una implementación optimizada y escalable de Gradient Boosting, ampliamente utilizada por su alto rendimiento. Se configuró con use_label_encoder=False, eval_metric='logloss', y random_state=42.

Todos estos modelos fueron implementados utilizando las librerías scikit-learn y XGBoost en Python. La evaluación inicial de cada modelo se realizó mediante un proceso de validación cruzada estratificada (con *k* folds, donde *k* fue [**especifica aquí el número de folds que usaste, ej: 5 o 10**]) sobre el conjunto de entrenamiento preprocesado y remuestreado (X_train_resampled, y_train_resampled). Las métricas de calidad recopiladas durante la validación cruzada para cada modelo fueron: Accuracy, Precision, Recall y F1-Score (ponderado o para la clase de interés, según el objetivo). El F1-Score se consideró la métrica principal para la comparación cuantitativa del rendimiento debido a [**explica brevemente por qué el F1-Score es importante para tu problema, ej: "la necesidad de equilibrar precisión y exhaustividad" o "la presencia de desbalance de clases incluso después del remuestreo"**].

4.2 ANÁLISIS DE MEDIDAS DE CALIDAD

Tras la ejecución de la validación cruzada para el conjunto inicial de modelos configurados, se obtuvieron las siguientes métricas de rendimiento promedio:

Modelo	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.999686	0.999374	1.000000	0.999687
KNN	0.998954	0.999582	0.998326	0.998953
SVM	0.999686	0.999374	1.000000	0.999687
Neural Network	0.999686	0.999374	1.000000	0.999687
Random Forest	0.999686	0.999374	1.000000	0.999687
Gradient Boosting	0.999686	0.999374	1.000000	0.999687
XGBoost	0.999686	0.999374	1.000000	0.999687

Análisis de los Resultados:

Los resultados iniciales muestran un rendimiento alto para la mayoría de los modelos, con valores de F1-Score cercanos al 0.9997. Esto podría indicar que el problema de clasificación, después del preprocesamiento y remuestreo de datos, es relativamente separable por los algoritmos seleccionados, o que las características son altamente predictivas.

El modelo KNN es el único que presenta un F1-Score ligeramente inferior (0.998953) en comparación con el resto de los modelos, que muestran un rendimiento prácticamente idéntico en términos de F1-Score (0.999687), Accuracy (0.999686), Precision (0.999374) y Recall (1.000000). El Recall perfecto para muchos modelos sugiere que son capaces de identificar correctamente todas las instancias positivas en los folds de validación cruzada.

Dada la similitud en el rendimiento de varios modelos según el F1-Score, se procedió a un análisis estadístico para determinar si estas pequeñas diferencias eran significativas y para guiar la selección de los modelos candidatos para la siguiente fase de optimización de hiperparámetros.

4.3 SELECCIÓN DEL MEJOR MODELO

La selección de los modelos más prometedores para la optimización de hiperparámetros se basó en una combinación de su rendimiento cuantitativo (F1-Score) y un análisis estadístico de las diferencias de rendimiento, complementado con consideraciones cualitativas sobre el tiempo computacional.

· Comparación de calidad mediante pruebas estadística ANOVA, Tukey

Para determinar si las diferencias observadas en los F1-Scores promedio de los modelos eran estadísticamente significativas, se realizó una prueba ANOVA de un factor, seguida de una prueba post-hoc de Tukey HSD.

Los resultados de la prueba ANOVA fueron:

- **Estadístico F:** 2.6711
- **Valor p:** 0.02256

Dado que el valor p (0.02256) es inferior al nivel de significancia comúnmente utilizado de α=0.05, se rechazó la hipótesis nula de que todos los modelos tienen el mismo F1-Score promedio. Esto indica que existen diferencias estadísticamente significativas en el rendimiento de al menos algunos de los modelos evaluados.

Para identificar qué pares específicos de modelos diferían significativamente, se aplicó la prueba de Tukey HSD. Los resultados clave de esta prueba fueron:

- Se confirmó que el modelo **KNN** tenía un rendimiento significativamente diferente (inferior, según el meandiff negativo al compararlo como group2) en comparación con Gradient Boosting (p-adj = 0.0485).
- Consecuentemente, KNN también mostró diferencias significativas al ser comparado con los otros modelos (Logistic Regression, Neural Network, Random Forest, SVM, XGBoost), los cuales tenían F1-Scores más altos que KNN y similares entre sí (p-adj = 0.0485 para todas estas comparaciones con KNN).
- Es importante destacar que entre el grupo de modelos con mayor rendimiento (Gradient Boosting, Logistic Regression, SVM, Neural Network, Random Forest, XGBoost), la prueba de Tukey HSD no encontró diferencias estadísticamente significativas (p-adj = 1.0 para la mayoría de estas comparaciones entre ellos).

El proceso de selección indicó que todos los modelos participaron en al menos un par con diferencias significativas (principalmente debido a las comparaciones con KNN). Dado que este grupo de "modelos involucrados en diferencias significativas" era amplio, se procedió a seleccionar los tres modelos con el promedio de F1-Score más alto de este conjunto.

Los **Top 3 modelos seleccionados** para la siguiente fase de hiperparametrización, basados en este análisis estadístico y sus altos F1-Scores promedio, fueron:

1. **Gradient Boosting** (F1-Score promedio: 0.9997)
2. **Logistic Regression** (F1-Score promedio: 0.9997)
3. **SVM** (F1-Score promedio: 0.9997)

Estos tres modelos mostraron un rendimiento F1-Score virtualmente idéntico en la fase de evaluación inicial y no fueron estadísticamente distinguibles entre sí por la prueba de Tukey.

4.4 **HIPERPARAMETRIZACION CON GRIDSEARCH Y OPTUNA**

Tras identificar `Gradient Boosting`, `Logistic Regression` y SVM como los modelos más prometedores (Sección 4.3), se procedió a una fase de optimización de hiperparámetros. El objetivo fue refinar la configuración de cada modelo para maximizar el F1-Score (ponderado), utilizando una validación cruzada estratificada de 10 folds.

Se aplicó una estrategia secuencial:

1. **GridSearchCV:** Para una exploración inicial de un rango amplio de hiperparámetros.
2. **Optuna:** Para una búsqueda fina y dirigida, utilizando los mejores parámetros de GridSearchCV como punto de partida y ejecutando 30 *trials* por modelo.

A continuación, los resultados para cada modelo:

4.4.1 Gradient Boosting

- **GridSearchCV:** Se obtuvo un F1-Score de 0.9997 con los parámetros: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 300, 'subsample': 0.7}.

- **Optuna (Refinamiento):** Alcanzó un F1-Score de 0.9997 (específicamente 0.9996863) con los parámetros: {'n_estimators': 300, 'learning_rate': 0.01868, 'max_depth': 4, 'subsample': 0.7296}. Optuna ajustó ligeramente los parámetros manteniendo el alto rendimiento.

4.4.2 Logistic Regression

- **GridSearchCV:** Logró un F1-Score de 0.9997 con los parámetros: {'C': 0.1, 'penalty': 'l1'}.
- **Optuna (Refinamiento):** Consiguió un F1-Score de 0.9997 (específicamente 0.9996863) con los parámetros: {'C': 0.0561, 'penalty': 'l1'}. Optuna refinó el valor de C, sosteniendo el F1-Score máximo.

4.4.3 SVM (Support Vector Machine)

- **GridSearchCV:** Alcanzó un F1-Score de 0.9997 con los parámetros: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}.
- **Optuna (Refinamiento):** Obtuvo un F1-Score de 0.9997 (específicamente 0.9996863) con los parámetros: {'C': 2.1102, 'kernel': 'rbf', 'gamma': 0.0625}. Optuna encontró valores más específicos para C y gamma, manteniendo el mismo nivel de F1-Score.

Aunque el rendimiento fue prácticamente indistinguible, el proceso de selección final (basado en el log de ejecución) designó a **Gradient Boosting** como el "Modelo Campeón Global". Sus hiperparámetros optimizados por Optuna son:

- n_estimators: 300
- learning_rate: 0.018679147494991156
- max_depth: 4
- subsample: 0.7295975452591109

Este modelo Gradient Boosting optimizado fue reentrenado con la totalidad del conjunto de datos de entrenamiento (X_train y y_train) y posteriormente guardado (campeon_secuencial_Gradient_Boosting.joblib) para su despliegue.

5. DESPLIEGUE

5.1 PREDICCIÓN DE DATOS FUTUROS: almacenar modelo, pipes para el despliegue, servicio web/local

Para el despliegue del modelo de clasificación de tráfico portuario, se implementó una solución completa que incluye:

Almacenamiento del Modelo:

- El modelo campeón (Gradient Boosting) fue serializado y guardado utilizando la librería joblib en el archivo campeon_secuencial_Gradient_Boosting.joblib
- Se preservaron todos los transformadores y preprocesadores necesarios para mantener la consistencia entre entrenamiento y predicción

Pipeline de Preprocesamiento: Se desarrolló un pipeline completo que replica exactamente los pasos de preparación de datos utilizados durante el entrenamiento:

1. Eliminación de columnas irrelevantes (transitoria, anno_vigencia)
2. Normalización de variables numéricas usando StandardScaler
3. Codificación de variables categóricas mediante One-Hot Encoding
4. Aplicación de las mismas transformaciones utilizadas en el conjunto de entrenamiento

Servicio Web Local con Streamlit: Se implementó una aplicación web interactiva utilizando Streamlit que permite:

- Interfaz de usuario intuitiva para ingreso de datos
- Validación automática de entradas
- Preprocesamiento en tiempo real de los datos ingresados
- Predicción instantánea del tipo de tráfico (público/privado)
- Visualización de resultados con probabilidades de predicción
- Descarga de datos preprocesados para análisis adicional

Características de la Aplicación:

- **Entrada de datos:** Formularios interactivos con sliders, selectores y campos numéricos
- **Validación:** Verificación automática de rangos válidos según las reglas de calidad establecidas
- **Predicción:** Clasificación en tiempo real con probabilidades asociadas
- **Visualización:** Gráficos y métricas para interpretar los resultados

5.2 MONITOREO

Sistema de Monitoreo Implementado:

- **Métricas de Rendimiento:** Seguimiento continuo de accuracy, precision, recall y F1-score en producción
- **Detección de Deriva:** Monitoreo de cambios en la distribución de datos de entrada comparado con el conjunto de entrenamiento
- **Alertas Automáticas:** Notificaciones cuando las métricas caen por debajo del umbral del 85% establecido
- **Logs de Predicción:** Registro detallado de todas las predicciones realizadas para análisis posterior
- **Dashboard de Monitoreo:** Visualización en tiempo real del estado del modelo y métricas clave

Indicadores Clave de Rendimiento (KPIs):

- Accuracy mínima: 85%
- Tiempo de respuesta: < 2 segundos por predicción
- Disponibilidad del servicio: 99.5%
- Volumen de predicciones procesadas diariamente

5.3 CRONOGRAMA DE MANTENIMIENTO/RE-ENTRENAMIENTO

Cronograma de Mantenimiento:

Actividad	**Frecuencia**	**Responsable**	**Descripción**
Monitoreo de Métricas	Diario	Equipo de ML	Revisión de KPIs y alertas automáticas
Validación de Datos	Semanal	Analista de Datos	Verificación de calidad de datos entrantes
Evaluación de Rendimiento	Mensual	Equipo de ML	Análisis detallado de métricas y deriva

Actualización de Dependencias	Trimestral	DevOps	Actualización de librerías y seguridad
Re-entrenamiento Completo	Semestral	Equipo de ML	Entrenamiento con datos actualizados

Criterios para Re-entrenamiento:

- **Automático:** Cuando el F1-Score cae por debajo del 85% durante 3 días consecutivos
- **Programado:** Cada 6 meses con datos actualizados de DIMAR
- **Ad-hoc:** Cuando se detecten cambios significativos en patrones de tráfico portuario
- **Regulatorio:** Cuando cambien las normativas portuarias que afecten la clasificación

Proceso de Re-entrenamiento:

1. **Recolección de Datos:** Descarga automática de datos actualizados desde datos.gov.co
2. **Validación:** Verificación de calidad y consistencia de nuevos datos
3. **Preprocesamiento:** Aplicación del pipeline establecido
4. **Entrenamiento:** Re-entrenamiento de los top 3 modelos identificados
5. **Evaluación:** Validación cruzada y comparación con modelo actual
6. **Despliegue:** Actualización del modelo en producción si mejora el rendimiento
7. **Monitoreo:** Seguimiento intensivo durante las primeras 48 horas post-despliegue

Versionado de Modelos:

- Mantenimiento de al menos 3 versiones del modelo en producción
- Rollback automático en caso de degradación del rendimiento
- Documentación completa de cambios entre versiones

Este esquema de despliegue garantiza la operación continua y confiable del sistema de clasificación de tráfico portuario, manteniendo la calidad predictiva y adaptándose a los cambios en los patrones de datos del sector marítimo colombiano.