Lead University
Curso: **2025- I BCD7214 Administración de Datos (Sección 1)**
Docente: Alejandro Zamora
Estudiantes: Carolina Salas, Kristhel Porras
Trabajo Grupal 1

1. Documentación Técnica: Sistema de Predicción de Exportación de Fertilizantes 1. Introducción

La predicción del país de destino en la exportación de fertilizantes representa un desafío debido a la ausencia de un modelo optimizado que integre datos históricos y características relevantes. Esta limitación puede generar errores en la planificación logística y costos adicionales. Para abordar este problema, se propone el desarrollo de un modelo de *Machine Learning* basado en XGBoost, diseñado para mejorar la precisión en la predicción del país de destino, optimizar la planificación logística y reducir costos. La solución incluye un *pipeline* automatizado de procesamiento de datos y una plataforma interactiva desarrollada en Streamlit.

Este documento presenta la documentación técnica del sistema de predicción de exportación de fertilizantes desde Costa Rica. Este se compone de un data pipeline junto con la aplicación del XGBoost. Para esto se hace una implementación Streamlit para la interfaz, Pandas y Scikit-learn para el procesamiento de datos, y XGBoost para el modelado predictivo. Se detallan la arquitectura del *pipeline* de datos, la integración del modelo de IA y las decisiones clave relacionadas con seguridad, criptografía y limpieza de datos.

2. Arquitectura del Data Pipeline

El Data Pipeline está diseñado para procesar los datos de exportación de fertilizantes y transformarlos en una estructura adecuada para la predicción. Sigue una arquitectura modular basada en los principios de ETL (Extracción, Transformación y Carga). A continuación, se describen sus componentes.

2.1 Extracción de Datos

La extracción de datos se realiza desde el Sistema de Administración de Datos de Agricultura (SADA) y los datos son almacenados en un archivo CSV en un repositorio de GitHub.

Las funciones utilizadas en esta fase incluyen:

- read_dataset: Carga los datos desde una URL o un archivo local, detectando automáticamente el separador.
- logging.info: Registra eventos importantes en un archivo de auditoría.

2.2 Transformación de Datos

El sistema aplica varias técnicas de limpieza, conversión de formatos, normalización y encriptación de datos.

2.2.1 Limpieza de datos

Para garantizar la calidad de los datos, se llevan a cabo las siguientes tareas:

- identify_duplicates: Detecta registros duplicados en el dataset.
- drop_duplicates: Elimina registros duplicados del dataset.
- check_missing_values: Identifica la cantidad de valores nulos en cada columna.
- delete_irrelevant_values: Identifica y elimina columnas que contienen un único valor en todos los registros.

2.2.2 Conversión de formatos

En esta etapa se transforman las variables para asegurar su correcto procesamiento en el modelo de predicción.

- _transform_column: Convierte las columnas categóricas en variables dummy o códigos categóricos según la configuración seleccionada.
- type_transform: Permite seleccionar si la transformación se aplicará a una columna específica o a todas las categóricas del dataset.

2.2.3 Normalización y escalado

Para garantizar que todas las características numéricas estén en una escala comparable, se aplica una normalización de los datos.

- standardize_data: Aplica la técnica de normalización con StandardScaler, asegurando que los valores tengan una media de 0 y una desviación estándar de 1.

2.2.4 Seguridad y encriptación

Para proteger los datos sensibles, el sistema incorpora un módulo de encriptación.

- _load_or_generate_key: Genera o carga una clave de encriptación que se utilizará para cifrar los datos sensibles.
- encrypt_column: Encripta una columna específica del dataset utilizando el algoritmo Fernet de la librería cryptography.
- save_encrypted_data: Guarda los datos encriptados en un archivo CSV para su almacenamiento seguro.

2.3 Carga de Datos

Una vez que los datos han sido procesados y transformados, se almacenan en diferentes ubicaciones según su propósito.

- save_data: Guarda los datos transformados en un archivo CSV en el repositorio de GitHub.
- backup_data: Crea una copia de seguridad de los datos limpios para prevenir pérdidas de información.

2.4 Visualización y Monitoreo

Para facilitar la interpretación y el análisis exploratorio, el sistema cuenta con funciones de visualización de datos.

- show_head: Muestra las primeras filas del dataset para revisión rápida.
- plot_distributions: Genera histogramas que permiten visualizar la distribución de cada variable numérica.

- correlation_matrix: Crea una matriz de correlación que muestra las relaciones entre las variables numéricas del dataset.
- outlier_detection: Utiliza diagramas de caja (boxplots) para detectar valores atípicos en cada variable numérica.
- advanced_outlier_detection: Aplica el método del rango intercuartil (IQR) para identificar valores extremos en las variables.

3. Integración del Modelo de IA

El modelo XGBoost Regressor se analiza utilizando varias métricas de evaluación, incluyendo el error cuadrático medio (RMSE), el error absoluto medio (MAE), el porcentaje de error absoluto medio (MAPE) y el coeficiente de determinación (R^2). Además, se implementan herramientas de visualización como gráficos de dispersión para comparar predicciones con valores reales, histogramas de distribución de predicciones y análisis de residuos para identificar patrones en los errores del modelo. Estos análisis permiten evaluar el desempeño del modelo y detectar posibles mejoras en la configuración de hiperparámetros o en el preprocesamiento de datos.

3.1 Carga de Datos

El modelo obtiene los datos por medio de una función una vez la estandarización en el pipeline ha sido completada.

- Función load_data(): Carga los datos desde un archivo data_final_estandarizado.csv. Si el archivo no está disponible, se genera un mensaje de error para evitar fallos en la ejecución del modelo.
- Variable objetivo (y): La variable de salida del modelo es Pais Destino.
- Variables predictoras (X): Todas las columnas del dataset excepto Pais Destino.

3.2 Entrenamiento del Modelo

El modelo XGBoost Regressor se entrena sobre los datos sin aplicar transformaciones adicionales.

- División del conjunto de datos:
  - Se emplea train_test_split(X, y, test_size=0.2, random_state=42), reservando el 80% de los datos para entrenamiento y el 20% para evaluación.
  - Se utiliza random_state=42 para garantizar la reproducibilidad de los resultados.
- Definición y entrenamiento del modelo:
  - El modelo se configura con los siguientes hiperparámetros:
    - n_estimators=100: Utiliza 100 árboles en el ensamble.
    - learning_rate=0.1: Controla la tasa de aprendizaje para evitar sobreajuste.
    - random_state=42: Asegura que los resultados sean reproducibles.
  - Función train_model(X, y):
    - Divide los datos en conjuntos de entrenamiento y prueba.
    - Ajusta el modelo XGBoost con los parámetros definidos.
    - Devuelve el modelo entrenado y las predicciones sobre el conjunto de prueba.

3.3 Evaluación y Visualización de Resultados

Tras el entrenamiento del modelo, se presentan métricas clave y visualizaciones para interpretar su rendimiento.

- Métricas de evaluación:
  - RMSE (Root Mean Squared Error)
  - MAE (Mean Absolute Error)
  - MAPE (Mean Absolute Percentage Error)
  - R^2 Score
- Visualización de resultados:
  - Función analyze_results(y_test, y_pred):
    - ❖ Muestra en Streamlit los valores de las métricas y su interpretación.
  - **Gráficos generados en Plotly**:
    - ❖ Comparación entre valores reales y predichos (scatter plot).
    - ❖ Distribución de predicciones (histograma).
    - ❖ Análisis de outliers (diagrama de caja).
    - ❖ Gráfico de residuos (scatter plot de residuos).

4. Pensamiento Final

Este sistema proporciona una solución eficiente para predecir el destino de exportación de fertilizantes, integrando autenticación, preprocesamiento de datos y modelado predictivo con XGBoost Regressor. Su arquitectura modular permite fácil extensión y mejora futura. Además, el almacenamiento de datos en archivos CSV dentro de un repositorio de GitHub facilita la accesibilidad y control de versiones, asegurando la trazabilidad de los datos y su disponibilidad para análisis posteriores.