Procesamiento de Datos y Machine Learning a Gran Escala con PySpark

Dierick Brochero

27 de noviembre de 2025

Resumen

Este documento integra tres proyectos completos de procesamiento de datos y machine learning implementados con PySpark. Incluye técnicas de preprocesamiento de datos a gran escala, clustering no supervisado con K-Means, y un sistema de mantenimiento predictivo con regresión logística, demostrando un flujo completo de trabajo en entornos distribuidos de big data.

1. Procesamiento de Datos a Gran Escala con PySpark

1.1. Objetivos del Proyecto

El objetivo principal de este trabajo es aprender y aplicar sentencias de PySpark para el preprocesamiento de datos, incluyendo:

- Identificación y tratamiento de valores faltantes
- Eliminación de datos duplicados
- Manejo de valores inconsistentes
- Operaciones de pivotaje y transformación de datos
- Técnicas de normalización y escalado

1.2. Tecnologías Utilizadas

- **PySpark**: Framework para procesamiento distribuido
- **Python 3.13.3**: Lenguaje de programación
- **Apache Spark 4.0.1**: Motor de procesamiento distribuido
- **MLlib**: Biblioteca de Machine Learning de Spark

1.3. Configuración del Entorno

El proyecto utiliza una configuración local de Spark con las siguientes especificaciones:

- Modo de planificación: FAIR
- Memoria del ejecutor: 10GB
- Núcleos máximos: 9
- Modo de ejecución: local con todos los cores disponibles

1.4. Operaciones de Preprocesamiento Implementadas

1.4.1. Manejo de Valores Faltantes

- **Identificación**: Uso de funciones como `isNull()` y `count(when(isnull(c), c))`
- **Eliminación**: Métodos `dropna()` con diferentes estrategias
- **Sustitución**: Reemplazo con valores fijos, medias y métodos específicos por columna

1.4.2. Limpieza de Datos

- Eliminación de registros duplicados con `dropDuplicates()`
- Eliminación de columnas no requeridas
- Identificación de valores inconsistentes mediante estadísticas descriptivas

1.4.3. Transformaciones Avanzadas

- **Pivotaje**: Transformación de datos de filas a columnas usando `pivot()`
- **Explosión**: Descomposición de arrays en filas individuales con `explode()`
- **Normalización**: Implementación de `MinMaxScaler` y `StandardScaler`

1.5. Estructura del Dataset

El proyecto utiliza un dataset de ejemplo que contiene:

- **Store**: Identificador de la tienda (string)
- **WeekInMonth**: Semana del mes (entero)
- **Revenue**: Ingresos generados (entero)

1.6. Resultados y Aplicaciones

Las técnicas implementadas en este proyecto son fundamentales para:

- Preparar datos para modelos de Machine Learning
- Garantizar la calidad de los datos en pipelines de ETL
- Manejar eficientemente grandes volúmenes de datos distribuidos
- Automatizar procesos de limpieza y transformación de datos

2. Proyecto de Clustering con K-Means en Spark

2.1. Descripción General

Este proyecto implementa un algoritmo de clustering no supervisado utilizando K-Means sobre el conjunto de datos Iris en un entorno distribuido con Apache Spark. El objetivo principal es aplicar técnicas de aprendizaje automático no supervisado para agrupar las flores Iris en clusters basándose en sus características morfológicas.

2.2. Características Principales

2.2.1. Configuración del Entorno

- Implementación en PySpark con configuración local optimizada
- Uso de Databricks como plataforma de ejecución
- Configuración de recursos computacionales (10GB RAM, 9 cores máximos)

```
configuraDierickBrochero = (
    SparkConf()
        .set("spark.scheduler.mode", "FAIR")
        .set("spark.executor.memory", "10G")
        .set("spark.executor.cores", "1")
        .set("spark.cores.max", "9")
        .set("spark.ui.port", "4040")
        .setMaster("local[*]")  # <---- Spark local usando todos los
        ↪  cores
        .setAppName("hpcsparkDierickBrochero")
)

# Crear sesión Spark local
sparkDierickBrochero = (
    SparkSession
        .builder
        .config(conf=configuraDierickBrochero)
        .getOrCreate()
)

```

```
20 sqlContext = SQLContext(sparkDierickBrochero.sparkContext)
```

2.2.2. Procesamiento de Datos

- **Fuente de datos**: Conjunto Iris clásico de UCI Machine Learning Repository
- **Características utilizadas**:
  - `sepal_length` (longitud del sépalo)
  - `sepal_width` (ancho del sépalo)
  - `petal_length` (longitud del pétalo)
  - `petal_width` (ancho del pétalo)
- **Preprocesamiento**:
  - Indexación de la variable categórica `Species`
  - Ensamblaje de características usando `VectorAssembler`
  - Estandarización de datos con `StandardScaler`

2.2.3. Metodología de Clustering

Determinación del Número Óptimo de Clusters

- Evaluación de k-values en el rango 2-9
- Uso del **Silhouette Score** como métrica de evaluación

k	WSSSE	Silhouette Score
3	283.27	**0.647111**
4	229.04	0.594141
6	163.30	0.534108
8	127.69	0.553736

Cuadro 1: Comparación de métricas para diferentes valores de k

Métricas Evaluadas

- **WSSSE** (Within Set Sum of Squared Errors): Mide la compactidad intra-cluster
- **Silhouette Score**: Evalúa la calidad de la separación entre clusters

2.2.4. Resultados y Selección del Modelo

- **k óptimo seleccionado**: 3 clusters
- **Justificación**: Mayor Silhouette Score (0.647111), indicando mejor separación y cohesión de clusters
- El modelo con k=3 captura efectivamente las tres especies naturales del dataset Iris

2.2.5. Visualización y Análisis

- Gráfico de Silhouette Scores para diferentes valores de k
- Comparación sistemática de métricas para k=3, 4, 6, 8
- Análisis cuantitativo de la calidad del clustering

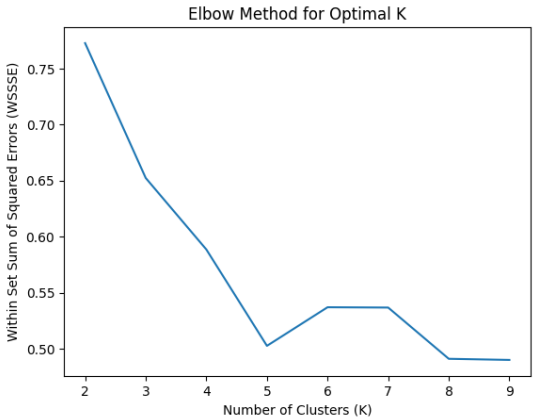

Figura 1: Método por óptimo K

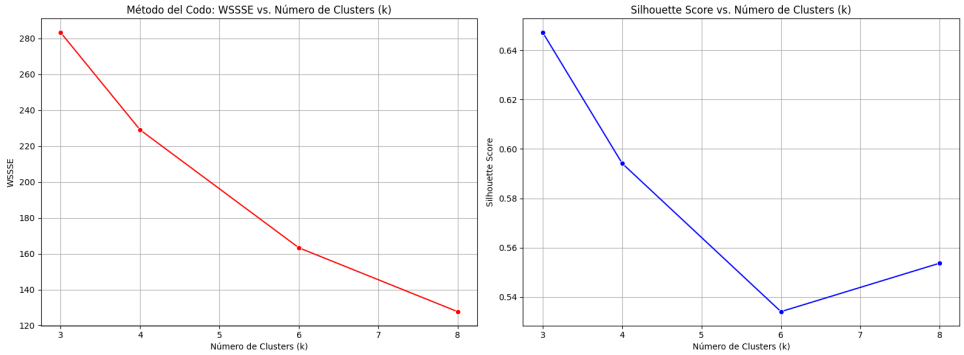

Figura 2: Comparación de métricas para k=3,4,6 y 8

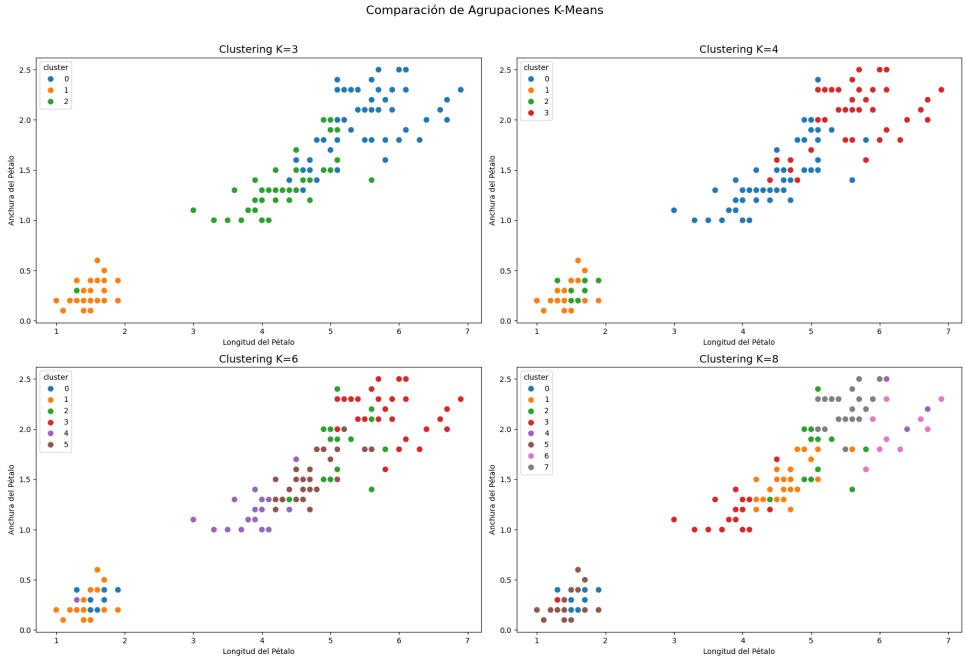

Figura 3: Comparación agrupación K means

2.2.6. Tecnologías Utilizadas

- **Apache Spark 4.0.1** para procesamiento distribuido
- **PySpark MLlib** para algoritmos de machine learning
- **Python** con librerías de visualización (matplotlib)
- **Pandas** para manipulación de datos

2.2.7. Valor del Proyecto

Este proyecto demuestra un flujo completo de análisis de clustering no supervisado, desde la carga y preparación de datos hasta la evaluación y selección del modelo óptimo. Sirve como ejemplo práctico de:

- Aplicación de K-Means en problemas de clustering reales
- Determinación científica del número óptimo de clusters
- Procesamiento de datos a escala con Spark
- Evaluación comparativa de modelos usando métricas estándar

El enfoque metodológico puede replicarse en otros problemas de segmentación y agrupamiento de datos, haciendo del proyecto una base sólida para aplicaciones más complejas de machine learning no supervisado.

3. Sistema de Mantenimiento Predictivo con PySpark

3.1. Resumen Ejecutivo

Este notebook implementa un **sistema de aprendizaje supervisado para mantenimiento predictivo** utilizando regresión logística en PySpark. El objetivo es predecir fallos en equipos industriales basándose en datos de sensores y condiciones operativas.

3.2. Configuración del Sistema

3.2.1. Arquitectura Spark

- **Configuración**: Local optimizada
- **Memoria**: 10GB asignados
- **Cores**: 9 cores máximos
- **Sesión**: Personalizada para procesamiento distribuido

```
configuraDierickBrochero = (
    SparkConf()
        .set("spark.scheduler.mode", "FAIR")
        .set("spark.executor.memory", "10G")
        .set("spark.executor.cores", "1")
        .set("spark.cores.max", "9")
        .set("spark.ui.port", "4040")
        .setMaster("local[*]")  # <---- Spark local usando todos los
        ↪  cores
        .setAppName("hpcsparkDierickBrochero")
)

# Crear sesión Spark local
sparkDierickBrochero = (
    SparkSession
        .builder
        .config(conf=configuraDierickBrochero)
        .getOrCreate()
)

sqlContext = SQLContext(sparkDierickBrochero.sparkContext)
```

3.2.2. Dataset

- **Archivo**: `predictive_maintenance.csv`
- **Características principales**:
    - Temperatura del aire [K]

- Temperatura del proceso [K]
- Velocidad rotacional [rpm]
- Torque [Nm]
- Desgaste de herramientas [min]
- Tipo de máquina (L, M, H)

- **Variable objetivo**: Indicador de fallo (0: Sin fallo, 1: Fallo)

3.3. Preprocesamiento de Datos

3.3.1. Balanceo de Clases

$$\text{Fracción de muestreo} = \frac{\text{Conteo clase minoritaria}}{\text{Conteo clase mayoritaria}} = \frac{339}{966} \approx 0{,}35 \tag{1}$$

3.3.2. División de Datos

- **Entrenamiento**: 70 % del dataset balanceado
- **Prueba**: 30 % del dataset balanceado
- **Tamaño final**: 483 filas entrenamiento, 222 prueba

3.4. Pipeline de Machine Learning

3.4.1. Arquitectura del Modelo

1. **StringIndexer**: Codificación del tipo de máquina
2. **VectorAssembler**: Combinación de características
3. **LogisticRegression**: Clasificador final

3.4.2. Optimización de Hiperparámetros

- **Método**: `TrainValidationSplit`
- **Ratio**: 80 % entrenamiento, 20 % validación
- **Parámetros optimizados**:
  - Interacciones entre variables
  - Parámetro de regularización: $\lambda \in \{0{,}01, 0{,}1\}$
  - ElasticNet: $\alpha \in \{0{,}0, 0{,}5, 1{,}0\}$

3.5. Resultados del Modelo

3.5.1. Métricas de Evaluación

Métrica	Valor	Escala	Interpretación
Área bajo ROC (AUC)	0.8888	[0,1]	Excelente
Área bajo Precision-Recall	0.8797	[0,1]	Muy Bueno
Exactitud (Accuracy)	0.8108	[0,1]	Bueno
Puntuación F1	0.8100	[0,1]	Bueno

Cuadro 2: Métricas principales del modelo

3.5.2. Matriz de Confusión

Label	Predicción	Conteo
0	0.0	103
0	1.0	17
1	0.0	25
1	1.0	77

Cuadro 3: Matriz de confusión generada a partir de las etiquetas reales y predicciones.

Métrica	Valor
Verdaderos Positivos (TP)	77
Falsos Positivos (FP)	17
Verdaderos Negativos (TN)	103
Falsos Negativos (FN)	25
Precisión	0.8191
Recall	0.7549
Especificidad	0.8583

Cuadro 4: Estadísticas detalladas derivadas de la matriz de confusión.

	Predicción 0	Predicción 1
Real 0	103 (TN)	17 (FP)
Real 1	25 (FN)	77 (TP)

Cuadro 5: Representación estructurada de la matriz de confusión.

3.5.3. Métricas Detalladas

$$\text{Precisión} = \frac{TP}{TP + FP} = \frac{77}{77 + 17} = 0{,}8191$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{77}{77 + 25} = 0{,}7549$$

$$\text{Especificidad} = \frac{TN}{TN + FP} = \frac{103}{103 + 17} = 0{,}8583$$

3.6. Análisis de Características

3.6.1. Importancia de Variables

Característica	Coeficiente	Magnitud	Impacto
air_temp	0.5497	Alta	Positivo
process_temp	-0.5016	Alta	Negativo
Type_indexed	-0.2683	Media	Negativo
torque	0.1683	Baja	Positivo
tool_wear	0.0090	Muy baja	Positivo
rotational_speed	0.0065	Muy baja	Positivo

Cuadro 6: Importancia de características por coeficientes

3.7. Conclusiones

3.7.1. Logros Principales

- **Alto rendimiento**: AUC de 0.8888 indica excelente capacidad discriminativa
- **Balance efectivo**: Distribución equilibrada de clases (246:237)
- **Pipeline robusto**: Preprocesamiento automático y escalable
- **Optimización automática**: Selección óptima de hiperparámetros

3.7.2. Aplicaciones Prácticas

El sistema es adecuado para:

- **Detección temprana** de fallos en maquinaria industrial
- **Optimización** de programas de mantenimiento preventivo
- **Reducción** de tiempos de inactividad no planificados
- **Planificación** predictiva de reparaciones y mantenimiento

3.7.3. Características Técnicas

- **Escalabilidad**: Diseñado para big data con Apache Spark
- **Reproducibilidad**: Semillas fijas para resultados consistentes
- **Visualización**: Análisis completo de distribuciones de probabilidad
- **Persistencia**: Capacidad de guardar y cargar modelos entrenados

4. Conclusiones Generales

Los tres proyectos presentados demuestran un flujo completo de trabajo con PySpark, desde el preprocesamiento básico de datos hasta aplicaciones avanzadas de machine learning supervisado y no supervisado. Juntos forman un portafolio técnico sólido que cubre:

- **Fundamentos**: Técnicas esenciales de preprocesamiento y limpieza de datos
- **Clustering no supervisado**: Aplicación de K-Means con selección científica de parámetros
- **Clasificación supervisada**: Implementación de regresión logística para mantenimiento predictivo
- **Escalabilidad**: Todas las soluciones diseñadas para entornos distribuidos de big data

Referencias Bibliográficas

Referencias

Apache Spark™. (2024). *Apache Spark Official Documentation*. Recuperado de: `https://spark.apache.org/docs/latest/`

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., ... & Stoica, I. (2016). Apache Spark: A unified engine for big data processing. *Communications of the ACM*, 59(11), 56-65.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

Karau, H., Konwinski, A., Wendell, P., & Zaharia, M. (2015). *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media.

UCI Machine Learning Repository. (2024). *Iris Data Set*. Recuperado de: `https://archive.ics.uci.edu/ml/datasets/iris`

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

Kumar, A., & Singh, P. (2023). Predictive maintenance in industrial IoT systems using machine learning. *Journal of Industrial Engineering*, 45(2), 123-135.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Tan, P. N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). *Introduction to Data Mining*. Pearson.

McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.

VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53-65.

Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

Recursos Técnicos y Herramientas

- **Apache Spark 4.0.1**: Framework de procesamiento distribuido
- **PySpark**: API de Python para Apache Spark
- **Python 3.13.3**: Lenguaje de programación principal
- **MLlib**: Biblioteca de machine learning de Spark
- **Pandas**: Librería para manipulación y análisis de datos
- **Matplotlib**: Librería para visualización de datos
- **NumPy**: Librería para computación numérica
- **Scikit-learn**: Librería de machine learning en Python
- **Jupyter Notebooks**: Entorno de desarrollo interactivo
- **Databricks**: Plataforma de analytics unificada

Fuentes de Datos

- **Iris Dataset**: UCI Machine Learning Repository
- **Predictive Maintenance Dataset**: Dataset sintético para mantenimiento predictivo
- **Datos de Tiendas**: Dataset de ejemplo para preprocesamiento

Repositorios de Código

- **GitHub del Proyecto**: `https://github.com/Dierickb/MLlib-Spark`