UNIVERSIDAD CATÓLICA BOLIVIANA

DEPARTAMENTO DE INGENIERIA Y CIENCIAS EXACTAS

MACHINE LEARNING

INFORME DE PROYECTO MODELO PREDICTIVO: VOLUNTARIADO

Estudiante:

HENRRY ALBERTO CORONADO VILLCA

Fecha: 06 de octubre de 2025

Santa Cruz – Bolivia

IMPORTANTE

El presente documento desarrollado tiene como objetivo presentar el desarrollo que se realizo en este proyecto, buscando documentar e informar el paso a paso de el producto. El resultado final y este documento es de dominio publico y no se buscara tener fines de lucro. El documento estará sujeto a cambios en el futuro por lo que se recomienda tener el ultimo versionado de este documento y su contenido.

Versión: 1.0

Tabla de contenido

1 INTRODUCCION

El presente informe documenta el desarrollo e implementación de un modelo de Machine Learning (ML) de Regresión Lineal Múltiple, cuyo objetivo principal es predecir la consistencia y el cumplimiento final de las horas de servicio de los voluntarios pastorales.

Tradicionalmente, la evaluación de riesgo se basa en la simple resta de horas acumuladas contra la meta. Este proyecto utiliza variables de patrón y capacidad (frecuencia, conflicto estructural, etc.) para ofrecer un diagnóstico predictivo avanzado a mitad del ciclo. El resultado es un sistema de apoyo que genera alertas tempranas y recomendaciones específicas para los voluntarios en riesgo de no alcanzar su meta al final del año, permitiendo a la Pastoral una gestión proactiva de la carga de trabajo.

2 OBJETIVOS

2.1 OBJETIVO GENERAL

Desarrollar un sistema de apoyo predictivo basado en Machine Learning para la gestión de voluntarios, capaz de pronosticar las Horas Totales Finales que cada voluntario acumulará, y utilizar esta predicción para emitir un diagnóstico de riesgo (Riesgo, Alerta, Normal) basado en la meta anual (N).

2.2 OBJETIVOS ESPECIFICOS

- Entrenar y evaluar un modelo de Regresión Lineal Múltiple que demuestre una capacidad de generalización (bajo RMSE y alto R2 en el conjunto de prueba).
- Implementar la persistencia para serializar el modelo entrenado y el escalador.
- Implementar reglas de negocio que traduzcan la predicción numérica (y-prediction) en un diagnóstico categórico práctico (Rojo, Amarillo, Verde).

3 DESARROLLO DEL TRABAJO

El trabajo se divide claramente en dos etapas para garantizar que el modelo no solo sea preciso, sino también funcional y usable en un entorno de producción (o en la herramienta de gestión diaria).

3.1 Etapa 1: Desarrollo y Diagnóstico del Modelo

Esta etapa se centra en la construcción, entrenamiento y validación rigurosa del algoritmo.

3.1.1 1. Preprocesamiento (División y Escalado)

División de Datos (60% Train / 20% Validation / 20% Test): Esta partición es la práctica estándar en ML. La separación del Test Set desde el inicio es crucial para garantizar que el error reportado (RMSETest) sea una estimación honesta de la performance del modelo en el mundo real (generalización).

Escalado de Características (StandardScaler): Las variables X fueron estandarizadas. El escalado asegura que todas las variables contribuyan de manera equitativa a la función de coste del optimizador Descenso de Gradiente Estocástico (SGD). El escalador se ajustó solo con el conjunto de entrenamiento para prevenir el Data Leakage.

3.1.2 2. Entrenamiento y Evaluación

Modelo de Regresión Lineal (SGDRegressor): Se eligió esta técnica por su alta interpretabilidad. Los coeficientes son directamente analizables, permitiendo a la Pastoral entender la contribución de cada factor a la predicción final.

Métricas de Evaluación (RMSE y R2):

El RMSE (Raíz del Error Cuadrático Medio) se usó para medir la magnitud del error en las unidades de la variable objetivo (horas).

El R2 (Coeficiente de Determinación) se usó para cuantificar el porcentaje de la varianza en Y explicado por las variables X.

3.1.3 3. Persistencia de Activos

Se utilizó la librería joblib para serializar el Modelo y el Escalador. Esto permite que el modelo se cargue en un entorno de aplicación sin necesidad de reentrenarlo, pasando a la Etapa 2.

3.2 Etapa 2: Operacionalización y Diagnóstico

Esta etapa describe cómo la Pastoral utiliza los activos guardados en un escenario de aplicación.

3.2.1 1. Carga de Activos y Nueva Entrada

Los archivos .pkl del modelo y del escalador son cargados en memoria.

Se ingresan las métricas actuales del voluntario en un nuevo DataFrame.

3.2.2 2. Preprocesamiento Operacional

Uso del Escalador Persistido: El nuevo DataFrame se somete a la transformación (.transform()) utilizando el escalador cargado.

Justificación: Los datos de entrada deben estandarizarse exactamente con la misma media y desviación estándar utilizadas durante el entrenamiento del modelo para que la predicción sea válida.

3.2.3 3. Predicción y Reglas de Negocio

Predicción Numérica: El modelo predice un valor continuo: (Y-prediction) (horas totales proyectadas).

Aplicación de Reglas de Negocio: La (Y-prediction) se compara con la Meta Anual (N) y los umbrales predefinidos (85% de N y 100% de N) para emitir el diagnóstico final:

ROJO (Riesgo Estructural): Proyección bajo el 85%.

AMARILLO (Cuerda Floja): Proyección bajo la Meta (N).

VERDE (Cumplimiento): Proyección supera la Meta.

4 RESULTADOS OBTENIDOS

La evaluación del modelo entrenado arrojó los siguientes resultados en el conjunto de prueba (Test Set), lo que indica su capacidad de generalización:

Métrica	Valor Obtenido	Interpretación
RMSE (Raíz del Error Cuadrático Medio)	±7.96 Horas	El modelo predice las horas finales con un margen de error promedio de aproximadamente ±8 horas.
R2 (Coeficiente de Determinación)	0.8640	El 86.40% de la variación observada en las horas finales de los voluntarios es explicada por las variables X incluidas en el modelo.

4.1 Coeficientes Aprendidos (Pesos del Modelo)

Los coeficientes (pesos) del modelo confirman el patrón conceptual diseñado, indicando la influencia de cada variable en la predicción de Horas Finales.

Variable	Peso (Coeficiente)	Interpretación
X1_Horas_Actuales	+17.65	Es el predictor más fuerte. Por cada desviación estándar en horas actuales, el resultado final aumenta significativamente.
X2_Frec_Semanal	+5.26	Alta frecuencia constante es un fuerte predictor positivo de cumplimiento final.
X3_Horas_Fallidas	−3.67	Las horas fallidas por conflicto actúan como un penalizador importante, indicando problemas estructurales.
X5_Disp_Neta_Restante.	+0.22	La capacidad restante de participar en actividades es un factor positivo, aunque menor que el ritmo actual.
X6_Antiguedad	+3.08	La experiencia (antigüedad) ofrece un beneficio predictivo positivo.
X4_Semanas_Restantes	≈0	Se excluye debido a que fue una variable constante en el conjunto de entrenamiento, siendo su efecto absorbido por el Intercepto

5 CONCLUSIONES

El modelo de Regresión Lineal optimizado con SGD es una herramienta predictiva robusta y eficiente para la Pastoral.

- Diagnóstico Proactivo: El sistema permite pasar de una gestión reactiva a una proactiva, identificando voluntarios que enfrentan un riesgo estructural. La separación en dos etapas asegura que la herramienta sea sostenible y fácil de actualizar.

- Validez Científica: El valor de R2≈0.89 demuestra la fuerte relación entre los factores de ritmo/conflicto y la consistencia final del voluntario.
- Usabilidad y Persistencia: La serialización con joblib y la clara distinción entre la fase de entrenamiento y la fase de uso (Etapa 1 y Etapa 2) hacen que el modelo sea inmediatamente operacionalizable en el sistema de gestión del voluntariado.

6 ANEXOS

6.1 Anexo 1: Training

6.1.1 Anexo1.1

VoluntaryPredictor_ModelTraining.py > ...

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import numpy as np
import joblib
import os

# Cargar el Dataset simulado
df = pd.read_csv('voluntariado_dataset.csv')

# Definir X (Variables Independientes) y Y (Variable Dependiente)
X = df[['X1_Horas_Actuales', 'X2_Frec_Semanal', 'X3_Horas_Fallidas',
        'X4_Semanas_Restantes', 'X5_Disp_Neta_Restante', 'X6_Antiguedad']]
Y = df['Y_Horas_Totales_Finales']

# Separar el Test Set (20%) del resto (80%)
X_train_val, X_test, Y_train_val, Y_test = train_test_split(
    X, Y, test_size=0.2, random_state=42
)

# Separar Validation Set (25% de X_train_val = 20% del total)
X_train, X_val, Y_train, Y_val = train_test_split(
    X_train_val, Y_train_val, test_size=0.25, random_state=42
)

print(f"Total de muestras (100%): {len(df)}")
print(f"Muestras Train (60%): {len(X_train)}")
print(f"Muestras Validation (20%): {len(X_val)}")
print(f"Muestras Test (20%): {len(X_test)}")

```

6.1.2 Anexo1.2

VoluntaryPredictor_ModelTraining.py > ...

```python
# Instanciamos el StandardScaler para estandarizar las características
scaler = StandardScaler()

# Entrenar el Scaler
scaler.fit(X_train)

# 2. Aplicar la Transformación a TODOS los conjuntos
X_train_scaled = scaler.transform(X_train)
X_val_scaled = scaler.transform(X_val)
X_test_scaled = scaler.transform(X_test)

# Instanciar el SGDRegressor:
model = SGDRegressor(
    loss='squared_error', # Función de coste para Regresión Lineal (Error Cuadrático Medio)
    max_iter=1000,
    tol=1e-3,
    random_state=42
)

# Entrenamiento: Ajuste de los parámetros (pesos Theta) para encontrar el patrón.
model.fit(X_train_scaled, Y_train)

# Mostrar el patrón aprendido por el modelo (coeficientes)
print("\n--- Patrón Aprendido (Coeficientes Theta) ---")
feature_names = X.columns
for name, coef in zip(feature_names, model.coef_):
    print(f"Peso de {name:<25}: {coef:>6.2f}")

# Predicciones en los tres conjuntos
Y_train_pred = model.predict(X_train_scaled)
Y_val_pred = model.predict(X_val_scaled)
Y_test_pred = model.predict(X_test_scaled)

```

6.1.3 Anexo1.3

VoluntaryPredictor_ModelTraining.py > ...

```python
# Función para calcular el RMSE (Raíz del Error Cuadrático Medio)
def calculate_rmse(Y_true, Y_pred):
    return np.sqrt(mean_squared_error(Y_true, Y_pred))

# Cálculo del RMSE para diagnóstico
rmse_train = calculate_rmse(Y_train, Y_train_pred)
rmse_val = calculate_rmse(Y_val, Y_val_pred)

print("\n--- Diagnóstico de Generalización ---")
print(f"RMSE (Train Set): {rmse_train:.2f} Horas")
print(f"RMSE (Validation Set): {rmse_val:.2f} Horas")

# Análisis de diagnóstico: Buscando Overfitting (Alta Varianza)
if rmse_val > rmse_train * 1.1: # Si el error de validación es > 10% mayor que el de train
    print("⚠ ¡ATENCIÓN! Alta Varianza (Overfitting). El modelo predice mejor sus datos de entrenamiento.")
else:
    print("✅ Buena Generalización. Los errores son similares, indicando bajo Overfitting.")

# Evaluación Final (Generalización Real)
rmse_test = calculate_rmse(Y_test, Y_test_pred)
r2_test = r2_score(Y_test, Y_test_pred) # Cálculo del R^2

print(f"\n--- Evaluación Final (Generalización Real) ---")
print(f"Error promedio de predicción (RMSE): ±{rmse_test:.2f} horas.")
print(f"Coeficiente de Determinación (R^2): {r2_test:.4f}")

print(f"Interpretación: El {r2_test*100:.2f}% de la variación en las horas finales es explicada por las variables del modelo.")

# Crear la carpeta para guardar los activos
if not os.path.exists('model_assets'):
    os.makedirs('model_assets')

joblib.dump(model, 'model_assets/voluntariado_regresor.pkl')
joblib.dump(scaler, 'model_assets/voluntariado_scaler.pkl')

print("\n✅ Persistencia exitosa: Modelo y Escalador guardados en 'model_assets/'.")
```

6.2 Anexo 2: Output Training

```
(entVirtual) C:\Users\HenrryCoronado\Documents\UCB\Semestre_VI\Machine L\Practices\Proyect>py VoluntaryPredictor_ModelTraining.py
Total de muestras (100%): 1000
Muestras Train (60%): 600
Muestras Validation (20%): 200
Muestras Test (20%): 200

--- Patrón Aprendido (Coeficientes Theta) ---
Peso de X1_Horas_Actuales        :  17.65
Peso de X2_Frec_Semanal          :   5.26
Peso de X3_Horas_Fallidas        :  -3.67
Peso de X4_Semanas_Restantes     :   0.00
Peso de X5_Disp_Neta_Restante    :   0.22
Peso de X6_Antiguedad            :   3.08

--- Diagnóstico de Generalización ---
RMSE (Train Set): 7.87 Horas
RMSE (Validation Set): 7.78 Horas
✅ Buena Generalización. Los errores son similares, indicando bajo Overfitting.

--- Evaluación Final (Generalización Real) ---
Error promedio de predicción (RMSE): ±7.96 horas.
Coeficiente de Determinación (R^2): 0.8640
Interpretación: El 86.40% de la variación en las horas finales es explicada por las variables del modelo.

✅ Persistencia exitosa: Modelo y Escalador guardados en 'model_assets/'.

(entVirtual) C:\Users\HenrryCoronado\Documents\UCB\Semestre_VI\Machine L\Practices\Proyect>
```

6.3 Anexo 3: Testing

6.3.1 Anexo3.1

VoluntaryPredictor_UseModel.py > ...

```python
import joblib
import pandas as pd
import numpy as np

HORAS_META_N = 100

# cargar modelo entrenado
loaded_model = joblib.load('model_assets/voluntariado_regresor.pkl')
loaded_scaler = joblib.load('model_assets/voluntariado_scaler.pkl')

# ejemplo
nuevos_datos_voluntario = pd.DataFrame([{
    'X1_Horas_Actuales': 30,
    'X2_Frec_Semanal': 1.0,
    'X3_Horas_Fallidas': 15,
    'X4_Semanas_Restantes': 8,
    'X5_Disp_Neta_Restante': 70, # Capacidad restante de actividades
    'X6_Antiguedad': 1
}])

# usar el scaler cargado para transformar los nuevos datos
X_nuevo_escalado = loaded_scaler.transform(nuevos_datos_voluntario)

# generar prediccion con los nuevos datos escalados
prediccion_horas_array = loaded_model.predict(X_nuevo_escalado)
y_prediccion = prediccion_horas_array[0]

print(f"Predicción Numérica del Modelo (y_prediccion): {y_prediccion:.1f} horas")

# definir umbrales para el diagnóstico
UMBRAL_RIESGO = HORAS_META_N * 0.85 # 85 horas
UMBRAL_ALERTA = HORAS_META_N * 1.00 # 100 horas

diagnostico = {
    'estado': 'Indefinido',
    'color': 'gris',
    'recomendacion_voluntario': '',
    'carga_semestral': 0
}

```

6.3.2 Anexo 3.2

VoluntaryPredictor_UseModel.py > ...

```python

# Lógica de Diagnóstico
if y_prediccion < UMBRAL_RIESGO:

    # condicion de riesgo (no cumple de ninguna manera)
    horas_faltantes = HORAS_META_N - y_prediccion

    diagnostico['estado'] = "RIESGO ESTRUCTURAL / ANOMALÍA"
    diagnostico['color'] = "ROJO"
    diagnostico['recomendacion_voluntario'] = (
        f"Alerta inmediata: Se proyecta una falta de {horas_faltantes:.1f} horas. "
        "Debe solicitar un plan de recuperación urgente y priorizar el cumplimiento de X5."
    )
    diagnostico['carga_semestre_2'] = horas_faltantes

elif y_prediccion < UMBRAL_ALERTA:

    # condicion de cuerda floja
    diagnostico['estado'] = "CUERDA FLOJA / ALERTA"
    diagnostico['color'] = "AMARILLO"
    diagnostico['recomendacion_voluntario'] = (
        "Atención: Se proyecta que quedará ligeramente por debajo de la meta. "
        "Debe asistir a TODAS las actividades remanentes posibles para cumplir. El margen de error es mínimo."
    )
    diagnostico['carga_semestre_2'] = HORAS_META_N - y_prediccion

else:
    # condicion normal
    horas_extra = y_prediccion - HORAS_META_N

    diagnostico['estado'] = "OPERACIÓN NORMAL / CUMPLIMIENTO"
    diagnostico['color'] = "VERDE"
    diagnostico['recomendacion_voluntario'] = (
        f"Felicidades: Se proyecta que superará la meta con {horas_extra:.1f} horas de colchón."
    )
    diagnostico['carga_semestre_2'] = 0

# mostrar diagnóstico
print("\n==========================================")
print(f"| Diagnóstico ML para el Voluntario: {diagnostico['estado']}")
print(f"| Estado de Riesgo: {diagnostico['color']}")
print("==========================================")
print(f"Recomendación para el Voluntario: {diagnostico['recomendacion_voluntario']}")
print(f"Proyección de Carga para Semestre 2: {diagnostico['carga_semestre_2']:.1f} horas a arrastrar.")
```

6.4 Anexo 4: Output Testing

```
(entVirtual) C:\Users\HenrryCoronado\Documents\UCB\Semestre_VI\Machine L\Practices\Proyect\entVirtual\Scripts>cd ../..

(entVirtual) C:\Users\HenrryCoronado\Documents\UCB\Semestre_VI\Machine L\Practices\Proyect>py VoluntaryPredictor_UseModel.py
Predicción Numérica del Modelo (y_prediccion): 60.2 horas


==========================================
| Diagnóstico ML para el Voluntario: RIESGO ESTRUCTURAL / ANOMALÍA
| Estado de Riesgo: ROJO
==========================================
Recomendación para el Voluntario: Alerta inmediata: Se proyecta una falta de 39.8 horas. Debe solicitar un plan de recuperación urgente y priorizar el cumpli
miento de X5.
Proyección de Carga para Semestre 2: 39.8 horas a arrastrar.

(entVirtual) C:\Users\HenrryCoronado\Documents\UCB\Semestre_VI\Machine L\Practices\Proyect>
```