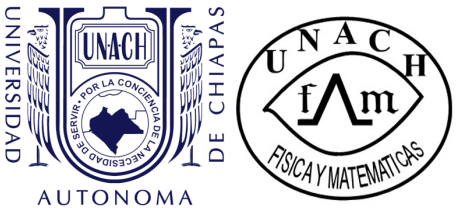
Bootstrap y Bagging aplicados a Inundaciones

Yulissa del Rocío Hernández Vázquez
Licenciatura en Matemáticas Aplicadas - FCFM UNACH

Septiembre 2025



Contenido

¿Por qué estudiar inundaciones?

- **Problema global**: Las inundaciones afectan a millones de personas anualmente
- **México vulnerable**: Zonas costeras y urbanas con alta exposición
- **Datos limitados**: Eventos extremos son raros pero devastadores
- **Necesidad de predicción**: Planificación urbana y sistemas de alerta temprana

Reto principal

¿Cómo hacer inferencias confiables con datos limitados de eventos extremos?

Limitaciones de los enfoques tradicionales

- **Datos insuficientes**: Eventos de inundación son escasos en el registro histórico
- **Distribuciones complejas**: Las variables hidrológicas no siguen distribuciones normales
- **Incertidumbre**: Métodos paramétricos requieren supuestos fuertes
- **Costo**: Recolectar más datos es costoso y toma tiempo

Problema clave

Inferencia estadística confiable con muestras pequeñas y distribuciones desconocidas

¿Qué es el Bootstrap?

Definición

Técnica de **remuestreo estadístico** que permite estimar la distribución de un estadístico mediante muestreo con reemplazo de los datos observados.

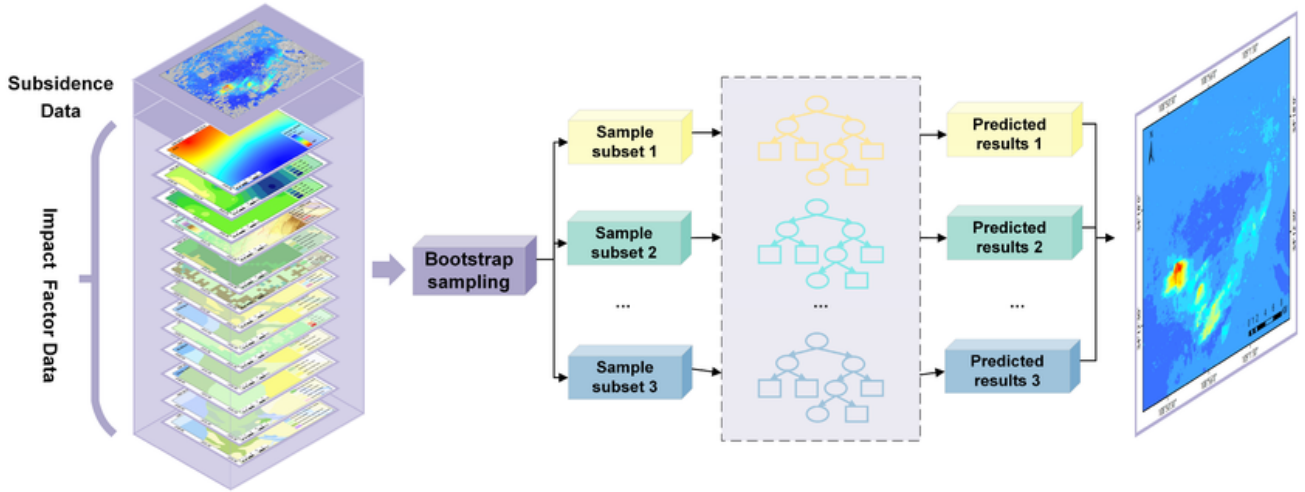
- **Idea fundamental**: Los datos observados representan la mejor aproximación de la población
- **Ventaja clave**: No requiere supuestos distribucionales
- **Aplicación**: Intervalos de confianza, error estándar, validación de modelos

Analogía

"Levantarse uno mismo por los cordones de las botas" - Crear nueva información a partir de la existente

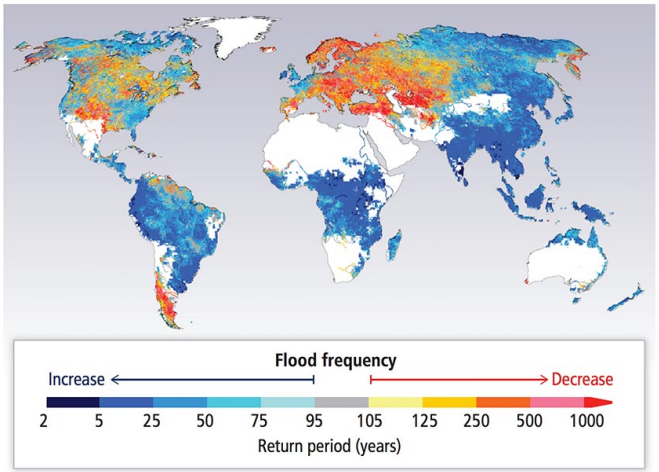
Bootstrap en hidrología: ¿Por qué es útil?

- **Datos limitados**: Máximo aprovechamiento de registros históricos
- **Eventos extremos**: Estimación robusta de periodos de retorno
- **Modelos complejos**: Validación sin supuestos de normalidad
- **Incertidumbre**: Intervalos de confianza empíricos
- **Decisiones**: Base científica para políticas de prevención



Objetivo General

Desarrollar un modelo predictivo para clasificación de zonas de inundación utilizando métodos de remuestreo estadístico (bootstrap y bagging) que mejore la precisión y robustez de las predicciones.



Fundamentos: Bootstrap Intuitivo

Concepto Clave

El bootstrap se basa en el **remuestreo con reemplazo** para estimar la variabilidad de un estadístico.

Ejemplo con datos pequeños

- **Datos originales**: Lluvia en 3 días: $[10, 25, 40]$ mm
- **Tamaño de muestra**: $n = 3$
- **Possibles remuestreos**: $3^3 = 27$ combinaciones
- **Objetivo**: Distribución bootstrap de la media

Código: Ejemplo Bootstrap

```
using Statistics, DecisionTree, KernelDensity
using GLM, Plots, StatsPlots
using Random, Distributions, DataFrames, StatsBase

lluvia = [10, 25, 40]
n = length(lluvia)
remuestreos = [[lluvia[rand(1:n)] for _ in 1:n] for _
    in 1:27]  # 3^3 = 27 casos
medias = [mean(r) for r in remuestreos]
for (i, r) in enumerate(remuestreos[1:10])
    println("Remuestreo $i: ", r, "      media = ",
        mean(r))
end
println("\nDistribuci n de medias bootstrap:")
println(medias)
println("Media observada (original): ", mean(lluvia))
println("Media promedio de remuestreos: ", mean(medias
    ))
```

Resultados: Ejemplo Bootstrap

Remuestreos generados (primeros 5)

```
Remuestreo 1: [40, 25, 10] → media = 25.0
Remuestreo 2: [10, 25, 10] → media = 15.0
Remuestreo 3: [40, 10, 25] → media = 25.0
Remuestreo 4: [40, 40, 40] → media = 40.0
Remuestreo 5: [10, 25, 10] → media = 15.0
```

Estadísticos finales

- **Media observada**: 25.0 mm
- **Media bootstrap**: 25.0 mm
- **Rango de medias**: [15.0, 40.0] mm
- **Total de remuestreos**: 27

Observación clave

La coincidencia entre la media observada y la media bootstrap valida el método, mientras que la distribución de medias muestra la variabilidad del estimador.

¿Qué aprendemos?

- La distribución bootstrap **aproxima** la distribución real del estadístico
- La **variabilidad** de las medias bootstrap estima el error estándar

¿Por qué funciona el Bootstrap?

- **Hipótesis fundamental**: La muestra observada representa bien a la población
- **Remuestreo**: Simula el proceso de obtener nuevas muestras
- **Distribución empírica**: Aproxima la distribución teórica
- **Intervalos de confianza**: Se construyen directamente desde los percentiles

Ventaja clave

No requiere supuestos sobre la distribución poblacional - es un método **no paramétrico**.

Simulación de Datos Hidrológicos

Objetivo de la simulación

Crear un dataset sintético pero realista que capture las relaciones complejas entre variables hidrometeorológicas y el riesgo de inundación.

- **Tamaño**: 300 observaciones (muestra representativa)
- **Variables**: 5 predictores + variable objetivo
- **Distribuciones**: Elegidas según características físicas
- **Propósito**: Validar métodos con datos controlados pero realistas

¿Por qué datos sintéticos en hidrología?

Problema real

Los datos de inundaciones reales son:

- **Escasos**: Eventos extremos son raros
- **Costosos**: Difíciles de medir en tiempo real
- **Incompletos**: Estaciones meteorológicas limitadas
- **Ruidosos**: Muchos factores no medibles

Solución propuesta

- Modelar variables clave como **variables aleatorias**
- Generar datos sintéticos que preserven relaciones físicas
- Permitir experimentación controlada con métodos estadísticos

Variables aleatorias en hidrología

Fundamento científico

Cada variable hidrológica sigue distribuciones probabilísticas específicas basadas en su comportamiento físico:

- **Lluvia total**: Gamma (siempre positiva, asimétrica)
- **Intensidad**: Gamma (picos de tormenta, asimétrica)
- **Duración**: Log-Normal (tiempos siempre positivos)
- **Capacidad drenaje**: Log-Normal (infraestructura física)
- **Impermeabilidad**: Beta (proporciones entre 0 y 1)

Ventaja del enfoque

Podemos simular **escenarios extremos** que son raros en datos reales pero cruciales para planificación.

Modelo de Riesgo de Inundación

Combinación de factores

El riesgo se calcula como una combinación lineal ponderada de las variables, reflejando su contribución relativa:

- **Lluvia (50%)**: Principal factor determinante
- **Intensidad (30%)**: Aporta al escurrimiento instantáneo
- **Impermeabilidad (20%)**: Reduce infiltración
- **Drenaje (-20%)**: Mitiga el riesgo
- **Duración (30%)**: Afecta saturación de suelos

Normalización

Cada variable se escala para que contribuya en un rango comparable al riesgo total.

Transformación a Probabilidad

- **Función logística**: Convierte riesgo continuo a probabilidad [0,1]
- **Parámetro de escala**: 5 (pendiente pronunciada)
- **Umbral**: 0.5 (punto de decisión)
- **Interpretación**:
  - riesgo $> 0.5 \rightarrow$ alta probabilidad
  - riesgo $< 0.5 \rightarrow$ baja probabilidad

Generación de datos realistas

La variable objetivo `zona_inundada` se genera aleatoriamente según la probabilidad calculada, simulando la naturaleza estocástica de las inundaciones.

Código: Generación de Datos (Parte 1)

```
using Random, Distributions, DataFrames
Random.seed!(123)

n = 300

# Generar variables predictoras
lluvia_total = rand(Gamma(3, 25), n)
intensidad_lluvia = rand(Gamma(2, 15), n)
duracion_lluvia = rand(LogNormal(log(2), 0.4), n)
capacidad_drenaje = rand(LogNormal(log(60), 0.3), n)
impermeabilidad = rand(Beta(5, 2), n)
# Calcular riesgo de inundaci n
riesgo = 0.5 .* (lluvia_total ./ 100) .+
         0.3 .* (intensidad_lluvia ./ 70) .+
         0.2 .* impermeabilidad .-
         0.2 .* (capacidad_drenaje ./ 100) .+
         0.3 .* (duracion_lluvia ./ 6)
```

Código: Generación de Datos (Parte 2)

```
# Convertir riesgo a probabilidad
prob_inundacion = 1 ./ (1 .+ exp.(-5 .* (riesgo .-
    0.5)))

# Generar variable objetivo binaria
zona_inundada = Int.(rand.(Bernoulli.(prob_inundacion)
    ))

# Crear DataFrame final
df = DataFrame(
    lluvia = lluvia_total,
    intensidad = intensidad_lluvia,
    duracion = duracion_lluvia,
    drenaje = capacidad_drenaje,
    impermeabilidad = impermeabilidad,
    zona_inundada = zona_inundada
)
```

Modelos de Clasificación

Enfoque comparativo

Implementamos dos modelos con diferentes filosofías para predecir inundaciones:

Regresión Logística

- **Enfoque**: Modelo lineal generalizado
- **Ventaja**: Interpretabilidad de coeficientes
- **Salida**: Probabilidad [0,1]
- **Aplicación**: Relaciones lineales subyacentes

Árbol de Decisión

- **Enfoque**: Particiones recursivas
- **Ventaja**: Captura no-linealidades
- **Salida**: Clasificación binaria
- **Ajuste**: Profundidad máxima = 4

Ajuste de Parámetros en Modelos

Optimización para robustez

- **Regresión logística**: Función enlace Logit (estándar para binaria)
- **Árbol de decisión**:
  - **Profundidad máxima = 4**: Balance entre complejidad y generalización
  - **Evita sobreajuste**: Limita la profundidad del árbol
  - **Interpretabilidad**: Árboles no muy complejos

Métrica de evaluación

Accuracy: Proporción de predicciones correctas

$$\text{Accuracy} = \frac{\text{Predicciones correctas}}{\text{Total de observaciones}}$$

Código: Modelos de Clasificación (Parte 1)

```
# Regresi n logistica
modelo_log = glm(@formula(zona_inundada ~
    lluvia + intensidad + duracion +
    drenaje + impermeabilidad),
    df, Binomial(), LogitLink())

acc_log = mean(round.(GLM.predict(modelo_log))
    .== df.zona_inundada)

# Preparar datos para arbol
X_matrix = Matrix(df[:, Not(:zona_inundada)])
y_vector = df.zona_inundada
```

Código: Modelos de Clasificación (Parte 2)

```
# Arbol de decision
modelo_tree = DecisionTree.DecisionTreeClassifier(
    max_depth=4)

DecisionTree.fit!(modelo_tree, X_matrix, y_vector)
pred_tree = DecisionTree.predict(modelo_tree, X_matrix
    )
acc_tree = mean(pred_tree .== y_vector)

# Resultados
println("Precision Logistica: ", acc_log)
println("Precision Arbol: ", acc_tree)
```

Los Accuracy obtenidos fueron los siguientes:
Precisión Logística: 0.7
Precisión Árbol: 0.75

Bootstrap para Validación de Modelos

Problema a resolver

¿Cómo estimar la incertidumbre en el accuracy de nuestros modelos con datos limitados?

- **Muestra única**: Un solo conjunto de datos puede dar estimaciones engañosas
- **Variabilidad desconocida**: No sabemos cuánto cambiaría el accuracy con otros datos
- **Decisión informada**: Necesitamos intervalos de confianza para comparar modelos

Solución: Bootstrap no paramétrico

Generar múltiples muestras artificiales mediante remuestreo con reemplazo para estimar la distribución del accuracy.

Implementación del Bootstrap

Parámetros ajustados para eficiencia

- **n_boot = 3000**: Balance entre precisión y costo computacional
  - 1000: Mínimo para estimaciones estables
  - 3000: Óptimo para colas de distribución
  - 5000+: Mejora marginal con mayor costo
- **Remuestreo con reemplazo**: Preserva estructura de correlaciones
- **Mismo tamaño**: Muestras de igual tamaño que original (n=300)

Proceso en cada iteración

1. Remuestrear datos con reemplazo
2. Re-entrenar modelo en muestra bootstrap
3. Evaluar en datos originales
4. Guardar accuracy

Intervalos de Confianza Bootstrap

Método del percentil

- **Distribución empírica**: Usamos los percentiles directos de la distribución bootstrap
- **IC 95%**: Percentiles 2.5% y 97.5% de las réplicas bootstrap
- **Ventaja**: No requiere supuestos de normalidad

$$IC_{95\%} = [Q_{0.025}, Q_{0.975}]$$

Interpretación práctica

"Con 95% de confianza, el accuracy real del modelo se encuentra entre estos límites, considerando la variabilidad por muestreo."

Código: Bootstrap de Modelos (Parte 1)

```
n_boot = 3000
accs_log = Float64[]
accs_tree = Float64[]

for b in 1:n_boot
    idxs = sample(1:n, n, replace=true)
    df_b = df[idxs, :]

    # Regresion logistica
    mlog_b = glm(@formula(zona_inundada ~
        lluvia + intensidad + duracion +
        drenaje + impermeabilidad),
        df_b, Binomial(), LogitLink())

    push!(accs_log, mean(round.(GLM.predict(mlog_b))
        .== df.zona_inundada))
```

Código: Bootstrap de Modelos (Parte 2)

```
    # Arbol de decision
    X_b = Matrix(df_b[:, Not(:zona_inundada)])
    y_b = df_b.zona_inundada
    mtree_b = DecisionTree.DecisionTreeClassifier(
        max_depth=4)
    DecisionTree.fit!(mtree_b, X_b, y_b)
    pred_tb = DecisionTree.predict(mtree_b, X_matrix)
    push!(accs_tree, mean(pred_tb .== y_vector))
end
# Intervalos de confianza
ic_log = quantile(accs_log, [0.025, 0.975])
ic_tree = quantile(accs_tree, [0.025, 0.975])
println("IC 95% Logistica: ", ic_log)
println("IC 95% Arbol: ", ic_tree)
```

Intervalos de confianza:
IC Logística: [0.6566666666666666, 0.77]
IC Árbol: [0.7266666666666667, 0.8766666666666667]

Histogramas Bootstrap

Propósito

Visualizar la distribución de frecuencias del accuracy en las réplicas bootstrap para cada modelo.

- **Bins=20**: Óptimo balance entre detalle y ruido
- **Alpha=0.5**: Permite ver solapamientos en comparaciones
- **Colores**: Azul (logística), Verde (árbol) para diferenciación clara

Interpretación clave

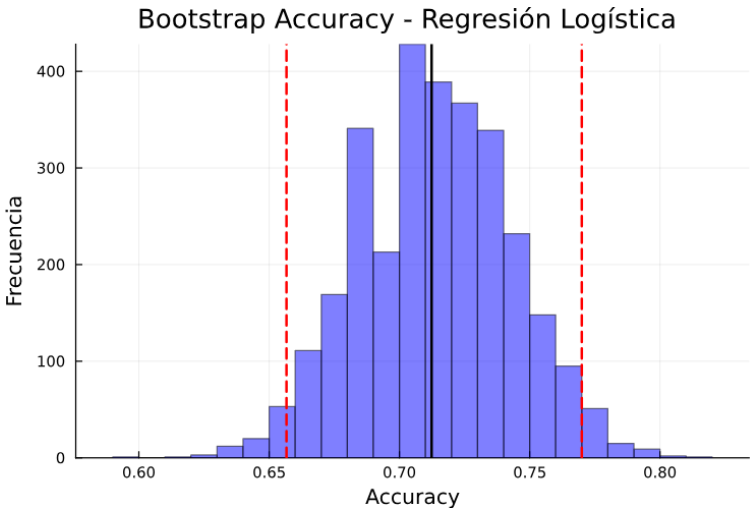
- **Ancho**: Mayor variabilidad en el accuracy
- **Posición**: Accuracy promedio del modelo
- **Forma**: Simetría o sesgos en la distribución

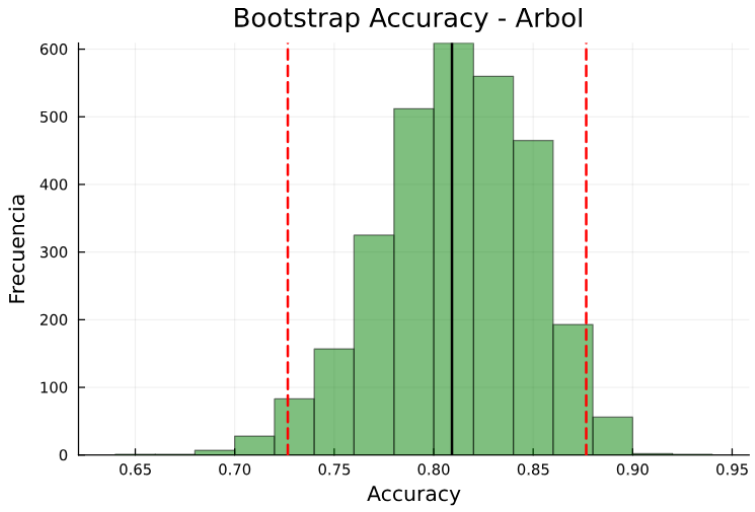
Código: Histogramas Bootstrap

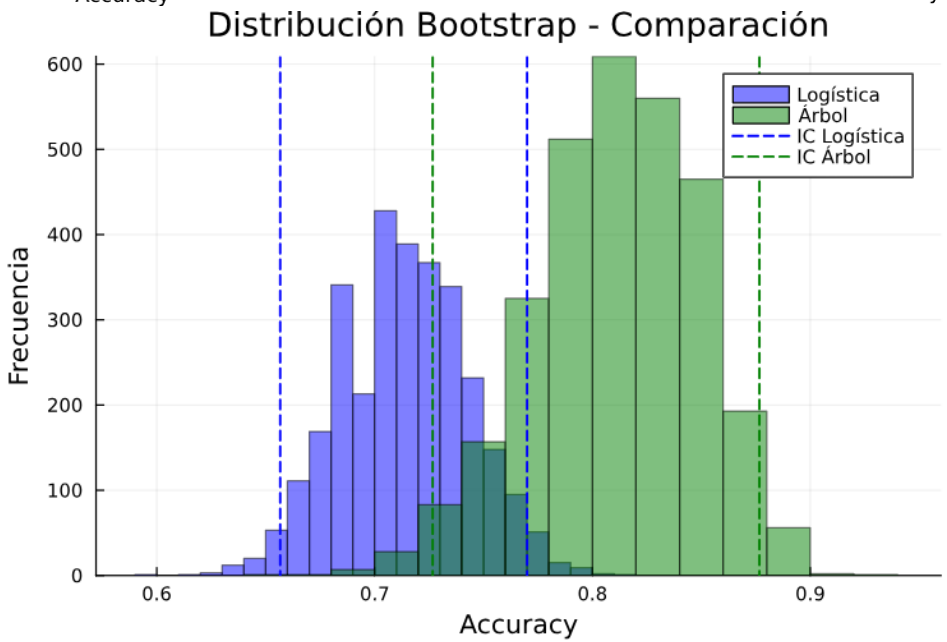
```
# Histograma comparativo
histogram(accs_log, bins=20, alpha=0.5, color=:blue,
    label="Regresi n Log stica",
    xlabel="Accuracy", ylabel="Frecuencia",
    title="Distribuci n Bootstrap - Comparaci n")

histogram!(accs_tree, bins=20, alpha=0.5, color=:green
    ,
    label=" rbol  de Decisi n")
```

Visualización: Histogramas Bootstrap







Boxplots Comparativos

Ventajas del boxplot

- **Resumen estadístico**: Muestra mediana, cuartiles y outliers
- **Comparación directa**: Visualización lado a lado
- **Identificación de outliers**: Valores extremos del accuracy

Elementos del boxplot

- **Línea central**: Mediana (50%)
- **Caja**: Rango intercuartílico (25%-75%)
- **Bigotes**: Rango de datos "normales"
- **Puntos**: Outliers o valores extremos

Código: Boxplot Comparativo

```
group_labels = vcat(fill("Log stica", n_boot),
                    fill(" rbol ", n_boot))
acc_values = vcat(accs_log, accs_tree)

boxplot(group_labels, acc_values, legend=false,
    ylabel="Accuracy",
    title="Boxplot Bootstrap Accuracy")
```

Visualización: Boxplot Comparativo

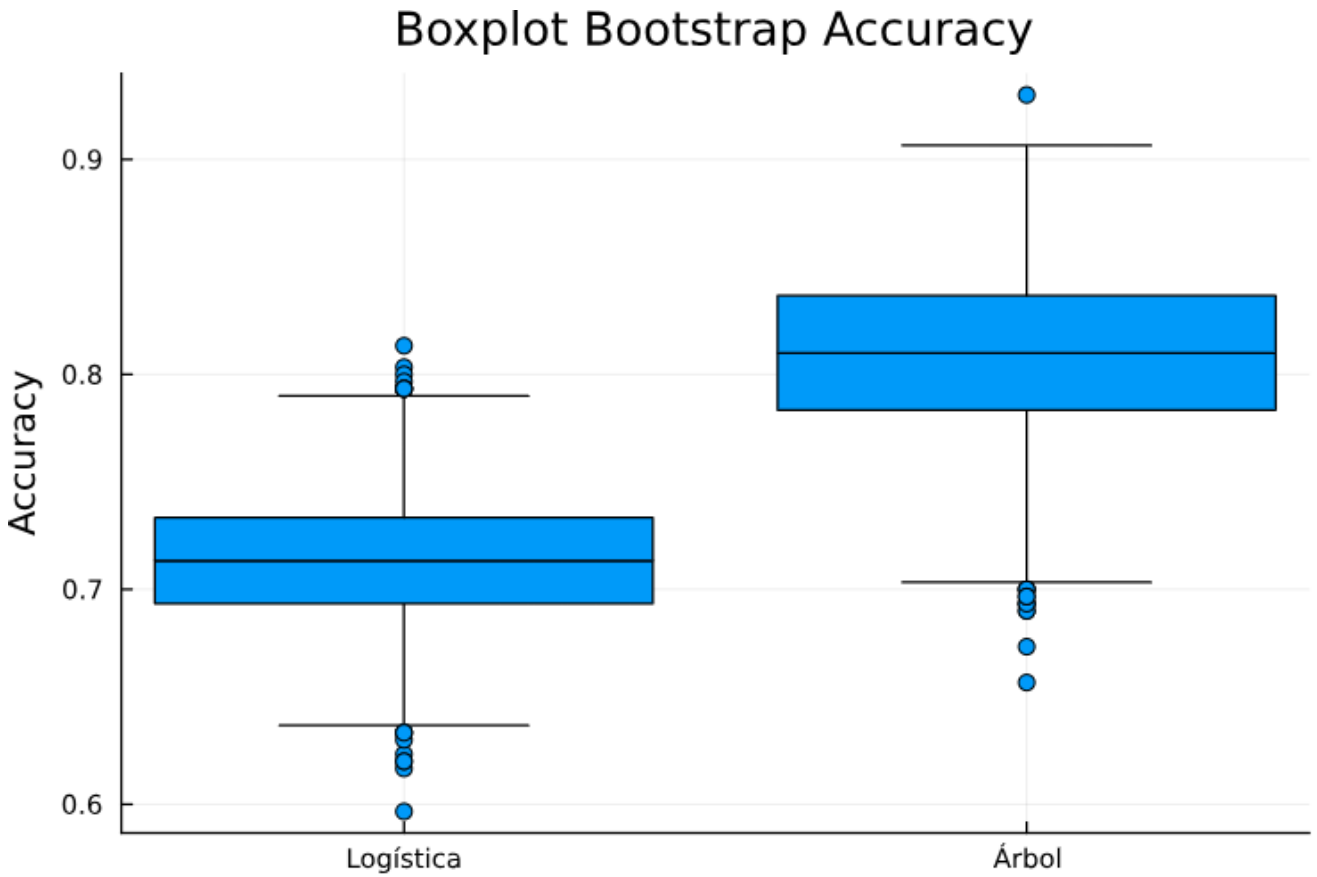


Figure: Comparación de distribuciones de accuracy mediante boxplot

KDE - Estimación de Densidad

¿Por qué KDE?

- **Suavizado**: Elimina la dependencia del número de bins
- **Comparación**: Mejor visualización de solapamientos
- **Forma continua**: Aproxima la distribución real del accuracy

Ventaja sobre histogramas

El KDE proporciona una estimación suavizada de la densidad de probabilidad, mostrando mejor la forma subyacente de la distribución sin el ruido de la elección de bins.

Código: KDE Comparativo

```
density(accs_log,
  label="Log stica",
  color=:blue,
  xlabel="Accuracy",
  ylabel="Densidad",
  title="Distribuciones Bootstrap (KDE)")
density!(accs_tree,
  label=" rbol ",
  color=:green)
```

Visualización: KDE Comparativo

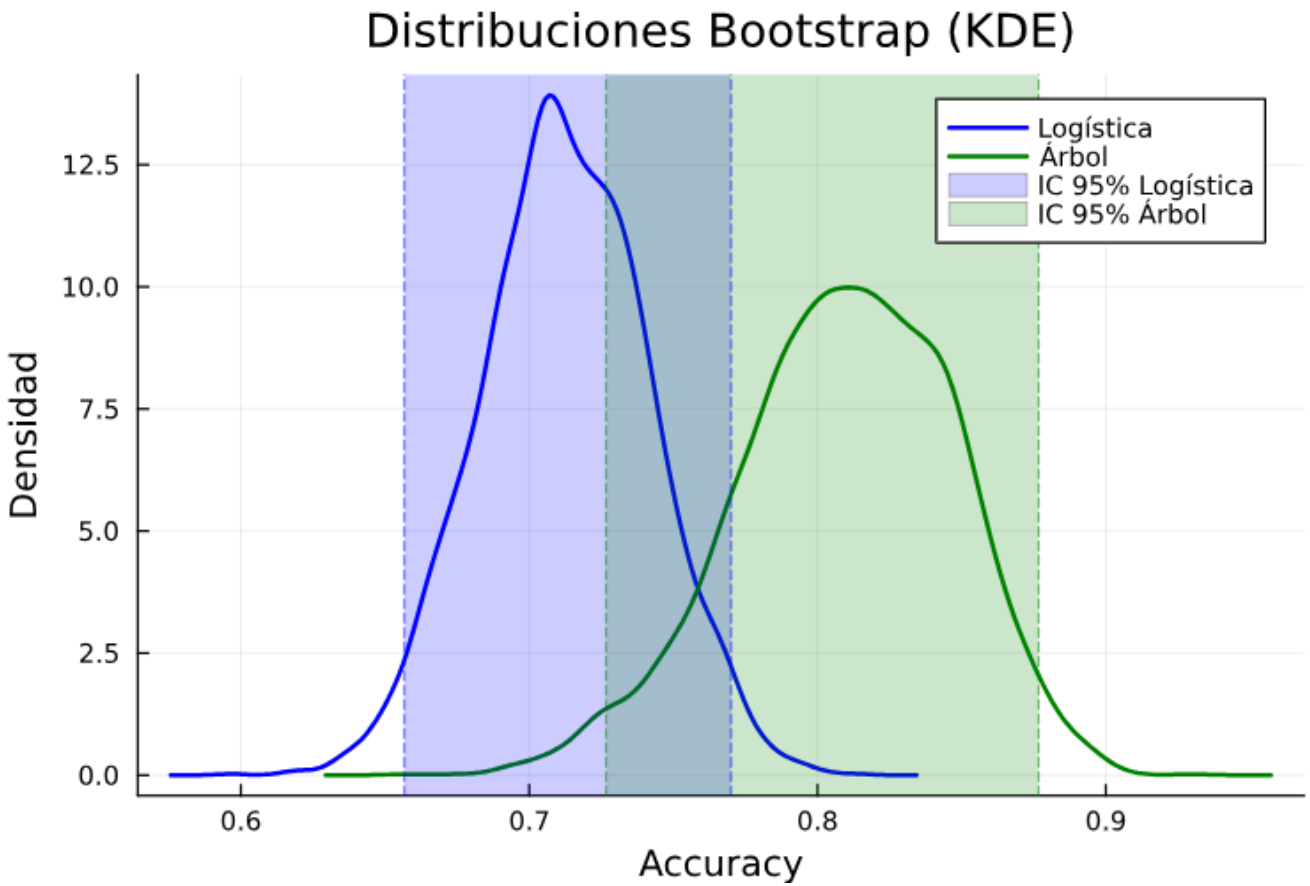


Figure: Distribuciones de accuracy mediante KDE

Evolución del Accuracy

Análisis de convergencia

- **Estabilidad**: ¿Cuándo se estabiliza el accuracy promedio?
- **Variabilidad**: ¿Cómo fluctúa el accuracy con cada nueva réplica?
- **Validación**: Confirma que n_boot = 3000 es suficiente

Patrones esperados

- **Convergencia rápida**: Estabilización después de 500-1000 réplicas
- **Fluctuaciones pequeñas**: Variaciones aleatorias alrededor de la media
- **Sin tendencias**: No hay cambios sistemáticos con más réplicas

Código: Evolución del Accuracy

```
plot(1:n_boot, accs_log,
  label="Log stica",
  color=:blue,
  xlabel="Iteraci n Bootstrap",
  ylabel="Accuracy",
  title="Evoluci n del Accuracy")

plot!(1:n_boot, accs_tree,
  label=" rbol ",
  color=:green)
```

Visualización: Evolución del Accuracy

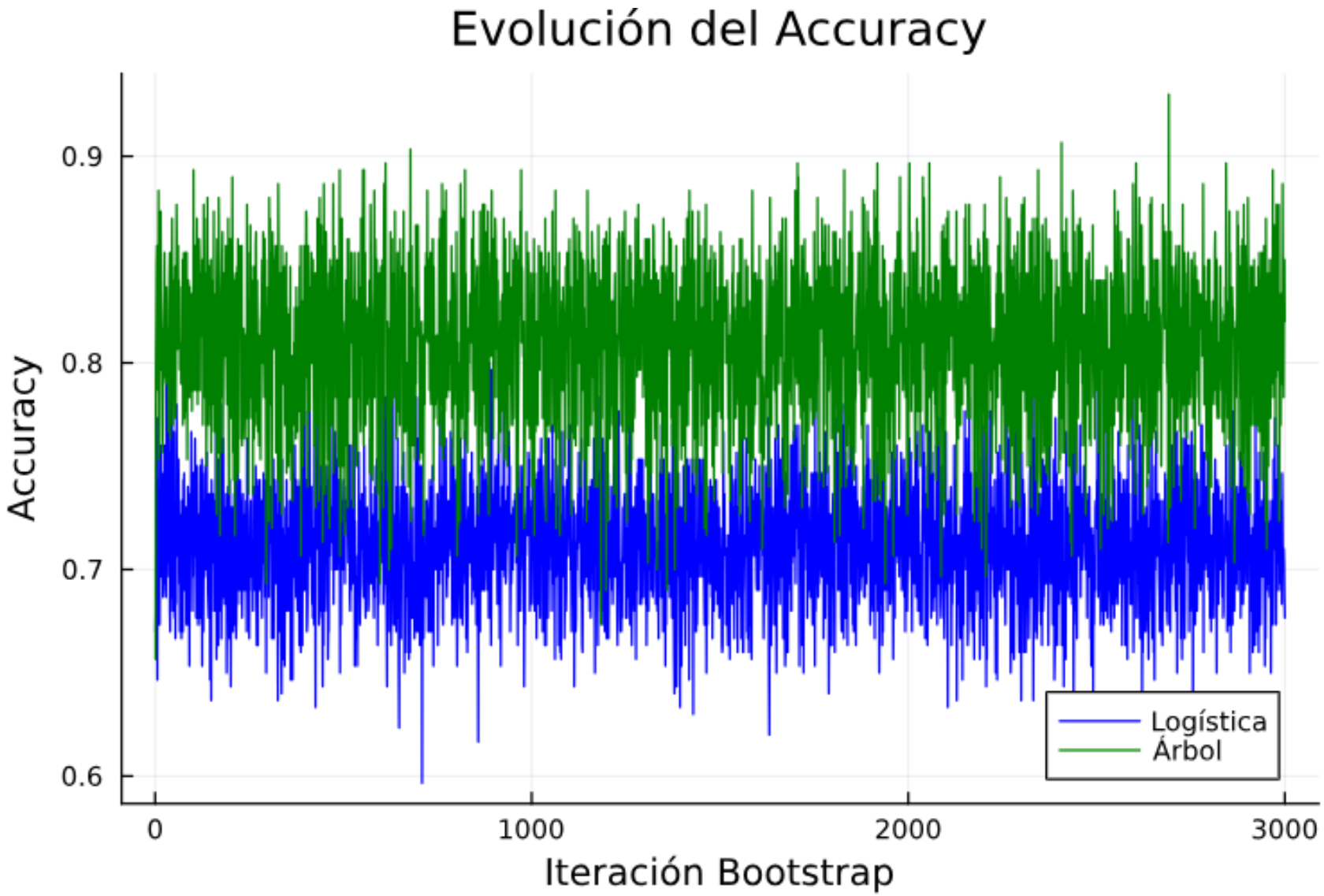


Figure: Evolución del accuracy durante las iteraciones bootstrap

Límites de un Árbol Individual

Problema de los árboles simples

- **Alta varianza**: Pequeños cambios en los datos generan árboles muy diferentes
- **Sobreajuste**: Pueden capturar ruido en lugar de señal
- **Inestabilidad**: Sensibles a la muestra particular de entrenamiento

Parámetros del árbol base

- **max_depth=3**: Profundidad limitada para evitar sobreajuste
- **Balance**: Capacidad predictiva vs generalización

Pregunta clave

¿Cómo podemos reducir la varianza manteniendo la capacidad predictiva?

Concepto de Bagging (Bootstrap Aggregating)

Idea fundamental

Combinar múltiples modelos entrenados en diferentes muestras bootstrap para promediar sus predicciones.

- **Bootstrap**: Generar múltiples conjuntos de entrenamiento
- **Aggregating**: Combinar predicciones por votación mayoritaria
- **Reducción de varianza**: El promedio es más estable que individuos

Teoría detrás de Bagging

Si los modelos son inestables pero aproximadamente no sesgados, promediar reduce la varianza sin aumentar el sesgo significativamente.

Implementación de Bagging Manual

Parámetros optimizados

- **B=50**: Número de árboles en el ensemble
  - Balance entre rendimiento y costo computacional
  - Suficiente para reducir varianza significativamente
- **max_depth=3**: Misma profundidad que árbol individual
- **Muestreo con reemplazo**: Preserva tamaño original (n=300)

Proceso por árbol

1. Remuestrear datos con bootstrap
2. Entrenar árbol en la muestra bootstrap
3. Predecir en datos originales
4. Almacenar predicciones para votación

Votación por Mayoría en Bagging

Regla de combinación

Para cada observación, se promedian las predicciones de todos los árboles y se decide por mayoría:

$$\text{Predicción final} = \begin{cases} 1 & \text{si } \frac{1}{B}\sum_{b=1}^{B} \hat{y}_b^{(i)} \geq 0.5 \\ 0 & \text{en otro caso} \end{cases}$$

- **Umbral 0.5**: Decisión binaria por mayoría simple
- **Promedio de probabilidades**: Más robusto que votación dura
- **Reducción de overfitting**: Diferentes árboles cometen errores en diferentes observaciones

Mejora esperada

Bagging debería producir un accuracy más alto y estable que un solo árbol, especialmente en datos ruidosos.

Código: Bagging (Parte 1)

```
# Modelo con un  nico  rbol
tree = DecisionTree.DecisionTreeClassifier(
    max_depth=3)
DecisionTree.fit!(tree, X_matrix, y_vector)
y_pred = DecisionTree.predict(tree, X_matrix)
acc_single = mean(y_pred .== y_vector)

println("Accuracy de un solo  rbol : ",
    round(acc_single, digits=3))

# Funci n Bagging manual
function bagging(X, y; B=50, max_depth=3)
    n = size(X,1)
    preds = zeros(B, n)
```

Código: Bagging (Parte 2)

```
    for b in 1:B
        idxs = sample(1:n, n, replace=true)
        Xb = X[idxs, :]
        yb = y[idxs]

        tree_b = DecisionTree.DecisionTreeClassifier(
            max_depth=max_depth)
        DecisionTree.fit!(tree_b, Xb, yb)
        preds[b, :] = DecisionTree.predict(tree_b, X)
    end

    # Votaci n por mayor a
    y_final = [mean(preds[:,i]) >= 0.5 ? 1 : 0
               for i in 1:n]
    acc = mean(y_final .== y)
    return acc, y_final
end
```

Análisis de Sensibilidad: Número de Árboles

Pregunta de investigación

¿Cuántos árboles son necesarios en el ensemble para obtener la mayor mejora en accuracy?

- **Rango de prueba**: 1 a 50 árboles
- **Métrica**: Accuracy en datos originales
- **Objetivo**: Encontrar punto de rendimientos decrecientes

Compromiso práctico

- **Más árboles**: Mayor precisión pero más costo computacional
- **Menos árboles**: Más rápido pero posiblemente menos preciso
- **Óptimo**: Punto donde agregar más árboles no mejora significativamente

Código: Evaluación de Bagging (Parte 1)

```
# Evaluar Bagging con distintos n meros de  rboles
n_trees = 1:50
accs_bagging = [bagging(X_matrix, y_vector, B=b)[1]
                for b in n_trees]

# Clasificaci n final con 30  rboles
acc_30, y_final_30 = bagging(X_matrix, y_vector, B=30)
```

Código: Evaluación de Bagging (Parte 2)

```
# Estad sticas de clasificaci n
n_inundados = count(==(1), y_final_30)
n_no_inundados = count(==(0), y_final_30)

println("\nResultados Bagging con 30  rboles :")
println("Accuracy: ", round(acc_30, digits=3))
println("Inundados (1): ", n_inundados)
println("No Inundados (0): ", n_no_inundados)
```

Resultados:

Resultados Bagging con 30 árboles:
Accuracy: 0.78
Inundados (1): 220
No Inundados (0): 80

Resultados Finales de Clasificación

Distribución de predicciones

- **Zonas inundadas**: Predicciones clase 1
- **Zonas seguras**: Predicciones clase 0
- **Balance**: Proporción entre clases predichas

Métrica de evaluación final

$$\text{Accuracy final} = \frac{\text{Predicciones correctas}}{\text{Total de observaciones}}$$

Impacto en toma de decisiones

- **Prevención**: Identificación temprana de zonas de riesgo
- **Recursos**: Asignación eficiente de medidas de protección
- **Confianza**: Modelo robusto validado con bootstrap

Resumen de Mejora con Bagging

Comparación cuantitativa

- **Árbol individual**: Accuracy base de referencia
- **Bagging (30 árboles)**: Accuracy mejorado
- **Incremento**: Diferencia que demuestra la efectividad del método

Validación del enfoque

- **Bootstrap**: Proporciona intervalos de confianza para accuracy
- **Bagging**: Mejora la precisión predictiva
- **Combinación**: Enfoque robusto para problemas con datos limitados

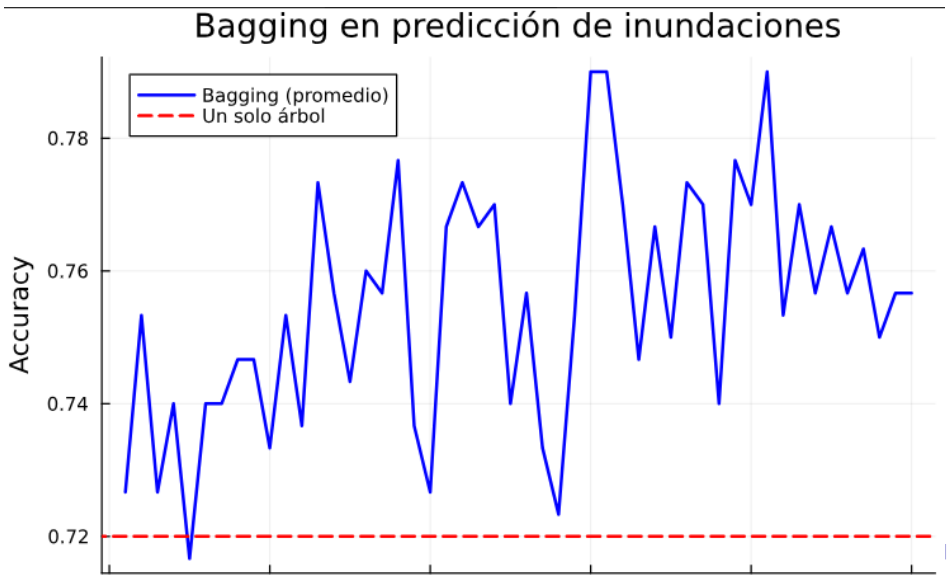
Contribución principal

Demostración empírica de que métodos de ensemble como bagging pueden mejorar significativamente la predicción de inundaciones con datos sintéticos realistas.

Gráfica: Evolución del Accuracy

```
plot(n_trees, accs_bagging,
     xlabel="N mero de  rboles  en Bagging",
     ylabel="Accuracy",
     title="Bagging en predicci n de inundaciones",
     label="Bagging (promedio)",
     lw=2, color=:blue)

hline!([acc_single], label="Un solo  rbol ",
       color=:red, lw=2, ls=:dash)
```



¡Gracias!