Clase 6: Métodos de Ensamble

Random Forest y Boosting

Matías Leoni

Aprendizaje Automático I
Maestría en IA - Universidad de San Andrés

22 de Julio de 2025

Agenda de la Clase

- **Bloque 1: Reduciendo la Varianza**

1. Ensambles de Modelos: Motivación y conceptos básicos.
2. Random Forest: Teoría, *bagging* y funcionamiento.
3. La matemática de Bagging: Reducción de la varianza.

Agenda de la Clase

- **Bloque 1: Reduciendo la Varianza**

1. Ensambles de Modelos: Motivación y conceptos básicos.
2. Random Forest: Teoría, *bagging* y funcionamiento.
3. La matemática de Bagging: Reducción de la varianza.

- **Bloque 2: Reduciendo el Sesgo**

4. Importancia de variables en Random Forest.
5. Introducción al Boosting: AdaBoost y Gradient Boosting.
6. Comparación: Random Forest vs. Boosting.
7. Aplicaciones y resumen.

Recordando: Árboles de Decisión (Clase 5)

- Los árboles de decisión son modelos muy interpretables y fáciles de visualizar.
- Particionan el espacio de predictores en regiones rectangulares.

Recordando: Árboles de Decisión (Clase 5)

- Los árboles de decisión son modelos muy interpretables y fáciles de visualizar.
- Particionan el espacio de predictores en regiones rectangulares.
- **Problema principal:** Un solo árbol de decisión tiende a tener alta varianza. Pequeños cambios en los datos de entrenamiento pueden resultar en árboles muy diferentes.

Recordando: Árboles de Decisión (Clase 5)

- Los árboles de decisión son modelos muy interpretables y fáciles de visualizar.
- Particionan el espacio de predictores en regiones rectangulares.
- **Problema principal:** Un solo árbol de decisión tiende a tener alta varianza. Pequeños cambios en los datos de entrenamiento pueden resultar en árboles muy diferentes.
- Son propensos al sobreajuste (*overfitting*) si no se podan adecuadamente.

Recordando: Árboles de Decisión (Clase 5)

- Los árboles de decisión son modelos muy interpretables y fáciles de visualizar.
- Particionan el espacio de predictores en regiones rectangulares.
- **Problema principal:** Un solo árbol de decisión tiende a tener alta varianza. Pequeños cambios en los datos de entrenamiento pueden resultar en árboles muy diferentes.
- Son propensos al sobreajuste (*overfitting*) si no se podan adecuadamente.
- **Pregunta clave:** ¿Podemos combinar muchos árboles para crear un modelo más robusto y preciso?

1. ¿Por qué Ensambles? La Sabiduría de la Multitud

- **Idea Central:** Un comité de expertos suele tomar mejores decisiones que un solo experto.

1. ¿Por qué Ensambles? La Sabiduría de la Multitud

- **Idea Central:** Un comité de expertos suele tomar mejores decisiones que un solo experto.
- En Machine Learning, combinamos las predicciones de varios modelos (llamados *weak learners* o aprendices débiles) para obtener una predicción final más potente.

1. ¿Por qué Ensambles? La Sabiduría de la Multitud

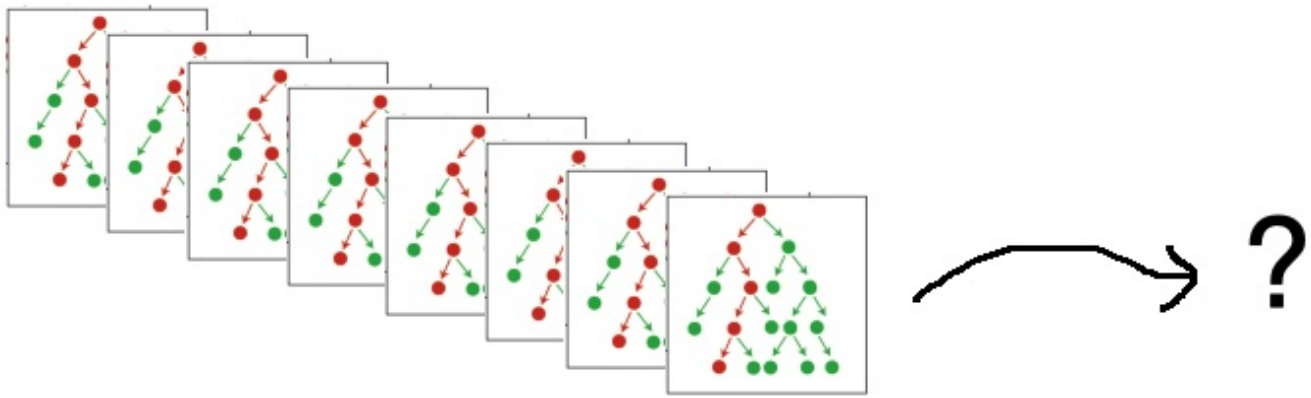
- **Idea Central:** Un comité de expertos suele tomar mejores decisiones que un solo experto.
- En Machine Learning, combinamos las predicciones de varios modelos (llamados *weak learners* o aprendices débiles) para obtener una predicción final más potente.
- Un aprendiz débil es un modelo que funciona solo ligeramente mejor que el azar. Los árboles de decisión pequeños son excelentes candidatos.

1. ¿Por qué Ensambles? La Sabiduría de la Multitud

- **Idea Central:** Un comité de expertos suele tomar mejores decisiones que un solo experto.
- En Machine Learning, combinamos las predicciones de varios modelos (llamados *weak learners* o aprendices débiles) para obtener una predicción final más potente.
- Un aprendiz débil es un modelo que funciona solo ligeramente mejor que el azar. Los árboles de decisión pequeños son excelentes candidatos.
- **Objetivo:** Mejorar la precisión predictiva y la estabilidad del modelo.

1. ¿Por qué Ensambles? La Sabiduría de la Multitud

- **Idea Central:** Un comité de expertos suele tomar mejores decisiones que un solo experto.
- En Machine Learning, combinamos las predicciones de varios modelos (llamados *weak learners* o aprendices débiles) para obtener una predicción final más potente.
- Un aprendiz débil es un modelo que funciona solo ligeramente mejor que el azar. Los árboles de decisión pequeños son excelentes candidatos.
- **Objetivo:** Mejorar la precisión predictiva y la estabilidad del modelo.
- Existen dos familias principales de métodos de ensamble: **Bagging** y **Boosting**.



2. Bagging: Bootstrap Aggregating

- **Bagging** es una técnica de ensamble diseñada principalmente para **reducir la varianza**.

2. Bagging: Bootstrap Aggregating

- **Bagging** es una técnica de ensamble diseñada principalmente para **reducir la varianza**.
- **Paso 1 (Bootstrap):** Generar B muestras de entrenamiento independientes a partir del dataset original, muestreando con reemplazo. Cada muestra bootstrap tiene el mismo tamaño que la original.

2. Bagging: Bootstrap Aggregating

- **Bagging** es una técnica de ensamble diseñada principalmente para **reducir la varianza**.
- **Paso 1 (Bootstrap):** Generar B muestras de entrenamiento independientes a partir del dataset original, muestreando con reemplazo. Cada muestra bootstrap tiene el mismo tamaño que la original.
- **Paso 2 (Aggregating):** Entrenar un modelo (ej. un árbol de decisión) en cada una de las B muestras.

2. Bagging: Bootstrap Aggregating

- **Bagging** es una técnica de ensamble diseñada principalmente para **reducir la varianza**.
- **Paso 1 (Bootstrap):** Generar B muestras de entrenamiento independientes a partir del dataset original, muestreando con reemplazo. Cada muestra bootstrap tiene el mismo tamaño que la original.
- **Paso 2 (Aggregating):** Entrenar un modelo (ej. un árbol de decisión) en cada una de las B muestras.
- **Paso 3 (Predicción):**
  - Para **regresión**: promediar las B predicciones.
  - Para **clasificación**: usar un "voto mayoritario" entre las B predicciones.

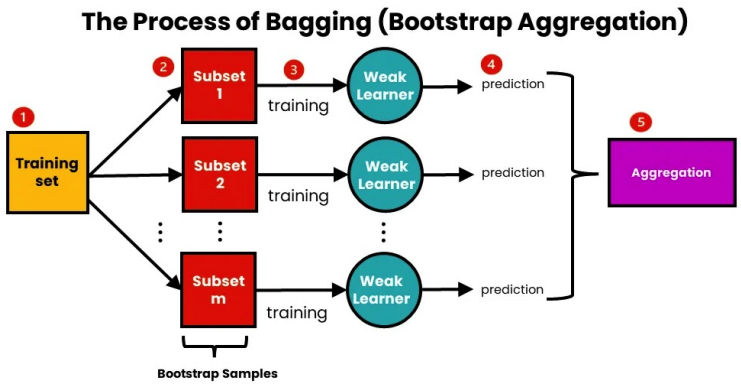


Figura: Esquema del proceso de Bagging.

3. La Magia de Bagging: Reducción de Varianza

Demostración en Pizarrón

- Supongamos que tenemos B observaciones (o predicciones de modelos) $Z_1, Z_2, \ldots, Z_B$, cada una con una varianza σ^2.

3. La Magia de Bagging: Reducción de Varianza

Demostración en Pizarrón

- Supongamos que tenemos B observaciones (o predicciones de modelos) $Z_1, Z_2, \ldots, Z_B$, cada una con una varianza σ^2.
- Si las observaciones son **independientes y no correlacionadas**, la varianza de su promedio $\bar{Z}$ es:

$$Var(\bar{Z}) = \sigma^2/B$$

- **Implicancia:** Al promediar las predicciones de B modelos no correlacionados, ¡reducimos la varianza del modelo final por un factor de B!

3. La Magia de Bagging: Reducción de Varianza

Demostración en Pizarrón

- Supongamos que tenemos B observaciones (o predicciones de modelos) $Z_1, Z_2, \ldots, Z_B$, cada una con una varianza σ^2.
- Si las observaciones son **independientes y no correlacionadas**, la varianza de su promedio $\bar{Z}$ es:

$$Var(\bar{Z}) = \sigma^2/B$$

- **Implicancia:** Al promediar las predicciones de B modelos no correlacionados, ¡reducimos la varianza del modelo final por un factor de B!
- **En la práctica:** Los árboles de Bagging no están descorrelacionados (se entrenan con datos similares). Pero la reducción de varianza sigue siendo muy significativa.

3. La Magia de Bagging: Reducción de Varianza

Demostración en Pizarrón

- Supongamos que tenemos B observaciones (o predicciones de modelos) $Z_1, Z_2, \ldots, Z_B$, cada una con una varianza σ^2.
- Si las observaciones son **independientes y no correlacionadas**, la varianza de su promedio $\bar{Z}$ es:

$$Var(\bar{Z}) = \sigma^2/B$$

- **Implicancia:** Al promediar las predicciones de B modelos no correlacionados, ¡reducimos la varianza del modelo final por un factor de B!
- **En la práctica:** Los árboles de Bagging no están descorrelacionados (se entrenan con datos similares). Pero la reducción de varianza sigue siendo muy significativa.

Para el Pizarrón

¿Qué pasa cuando existe una correlación ρ entre los árboles? Esto motivará el siguiente paso: Random Forest.

4. De Bagging a Random Forest

- **Problema de Bagging:** Si hay una variable predictora muy fuerte, la mayoría de los árboles la seleccionarán como el primer split.

4. De Bagging a Random Forest

- **Problema de Bagging:** Si hay una variable predictora muy fuerte, la mayoría de los árboles la seleccionarán como el primer split.
- Esto causa que los árboles generados estén **correlacionados** entre sí.

4. De Bagging a Random Forest

- **Problema de Bagging:** Si hay una variable predictora muy fuerte, la mayoría de los árboles la seleccionarán como el primer split.
- Esto causa que los árboles generados estén **correlacionados** entre sí.
- Como vimos, la correlación limita la efectividad de la reducción de varianza.

4. De Bagging a Random Forest

- **Problema de Bagging:** Si hay una variable predictora muy fuerte, la mayoría de los árboles la seleccionarán como el primer split.
- Esto causa que los árboles generados estén **correlacionados** entre sí.
- Como vimos, la correlación limita la efectividad de la reducción de varianza.
- **Solución de Random Forest:** Forzar a los árboles a ser diferentes introduciendo más aleatoriedad.

4. De Bagging a Random Forest

- **Problema de Bagging:** Si hay una variable predictora muy fuerte, la mayoría de los árboles la seleccionarán como el primer split.
- Esto causa que los árboles generados estén **correlacionados** entre sí.
- Como vimos, la correlación limita la efectividad de la reducción de varianza.
- **Solución de Random Forest:** Forzar a los árboles a ser diferentes introduciendo más aleatoriedad.
- **La gran idea:** En cada división (split) del árbol, considerar solo un subconjunto aleatorio de predictores.

4. De Bagging a Random Forest

- **Problema de Bagging:** Si hay una variable predictora muy fuerte, la mayoría de los árboles la seleccionarán como el primer split.
- Esto causa que los árboles generados estén **correlacionados** entre sí.
- Como vimos, la correlación limita la efectividad de la reducción de varianza.
- **Solución de Random Forest:** Forzar a los árboles a ser diferentes introduciendo más aleatoriedad.
- **La gran idea:** En cada división (split) del árbol, considerar solo un subconjunto aleatorio de predictores.

4. ¿Cómo Funciona un Random Forest?

- Es un ensamble de árboles de decisión que utiliza Bagging y una modificación adicional.

4. ¿Cómo Funciona un Random Forest?

- Es un ensamble de árboles de decisión que utiliza Bagging y una modificación adicional.
- **Algoritmo:**
  1. Generar B muestras bootstrap del set de entrenamiento.

4. ¿Cómo Funciona un Random Forest?

- Es un ensamble de árboles de decisión que utiliza Bagging y una modificación adicional.
- **Algoritmo:**
  1. Generar B muestras bootstrap del set de entrenamiento.
  2. Para cada muestra bootstrap, crecer un árbol de decisión.

4. ¿Cómo Funciona un Random Forest?

- Es un ensamble de árboles de decisión que utiliza Bagging y una modificación adicional.
- **Algoritmo:**
  1. Generar B muestras bootstrap del set de entrenamiento.
  2. Para cada muestra bootstrap, crecer un árbol de decisión.
  3. Al crecer cada árbol, en cada nodo candidato a ser dividido:
     - Seleccionar aleatoriamente m predictores del total de p disponibles.
     - Elegir el mejor predictor y punto de corte **solo de entre esos** m.

4. ¿Cómo Funciona un Random Forest?

- Es un ensamble de árboles de decisión que utiliza Bagging y una modificación adicional.
- **Algoritmo:**
  1. Generar B muestras bootstrap del set de entrenamiento.
  2. Para cada muestra bootstrap, crecer un árbol de decisión.
  3. Al crecer cada árbol, en cada nodo candidato a ser dividido:
     - Seleccionar aleatoriamente m predictores del total de p disponibles.
     - Elegir el mejor predictor y punto de corte **solo de entre esos** m.
  4. Agregar las predicciones de los B árboles (promedio o voto mayoritario).
- **Parámetro clave:** m. Usualmente se elige $m \approx \sqrt{p}$ para clasificación y $m \approx p/3$ para regresión. Si $m = p$, es simplemente Bagging.

Bonus de Bagging: Out-of-Bag (OOB) Error

- En el muestreo bootstrap, en promedio, cada árbol utiliza solo $\approx 2/3$ de las observaciones originales.

Bonus de Bagging: Out-of-Bag (OOB) Error

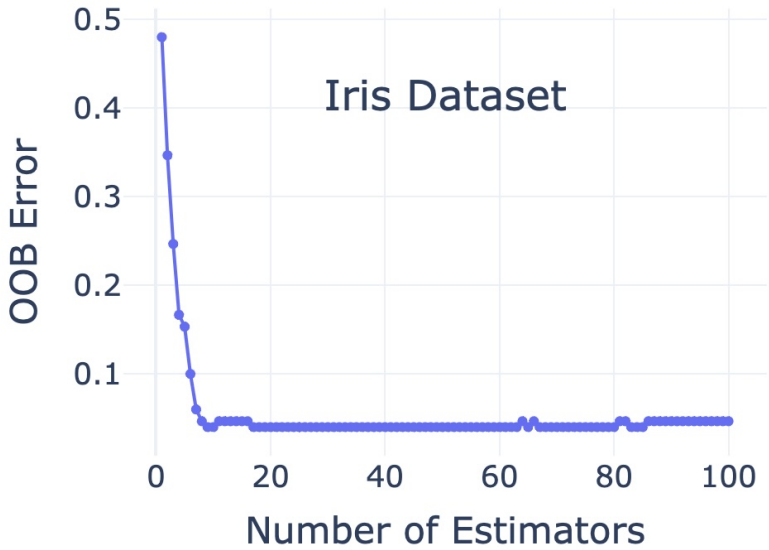
- En el muestreo bootstrap, en promedio, cada árbol utiliza solo $\approx 2/3$ de las observaciones originales.
- El $\approx 1/3$ restante son las observaciones **Out-of-Bag (OOB)** $\rightarrow$ Tenemos gratis estas observaciones como set de validación.

Bonus de Bagging: Out-of-Bag (OOB) Error

- En el muestreo bootstrap, en promedio, cada árbol utiliza solo $\approx 2/3$ de las observaciones originales.
- El $\approx 1/3$ restante son las observaciones **Out-of-Bag (OOB)** $\rightarrow$ Tenemos gratis estas observaciones como set de validación.
- Para cada observación x_i, promediamos las predicciones de los árboles que **no** la incluyeron en su muestra de entrenamiento.

Bonus de Bagging: Out-of-Bag (OOB) Error

- En el muestreo bootstrap, en promedio, cada árbol utiliza solo $\approx 2/3$ de las observaciones originales.
- El $\approx 1/3$ restante son las observaciones **Out-of-Bag (OOB)** $\rightarrow$ Tenemos gratis estas observaciones como set de validación.
- Para cada observación x_i, promediamos las predicciones de los árboles que **no** la incluyeron en su muestra de entrenamiento.
- Esto nos da una estimación del error de test (el **error OOB**) sin necesidad de usar Cross-Validation o un set de test explícito. ¡Muy eficiente!



Pausa

15 minutos

5. Importancia de Variables en Random Forest

- Los ensambles son a menudo considerados “cajas negras”, pero Random Forest nos da una métrica útil de importancia de variables.

5. Importancia de Variables en Random Forest

- Los ensambles son a menudo considerados “cajas negras”, pero Random Forest nos da una métrica útil de importancia de variables.
- **Idea intuitiva:** Si una variable es importante, usarla en las divisiones debería mejorar la “pureza” de los nodos del árbol.

5. Importancia de Variables en Random Forest

- Los ensambles son a menudo considerados “cajas negras”, pero Random Forest nos da una métrica útil de importancia de variables.
- **Idea intuitiva:** Si una variable es importante, usarla en las divisiones debería mejorar la “pureza” de los nodos del árbol.
- **Mean Decrease in Impurity (MDI):**
  - Se calcula para cada variable la reducción total del criterio de división (Entropía para clasificación, RSS para regresión) promediada sobre todos los árboles del ensamble.

5. Importancia de Variables en Random Forest

- Los ensambles son a menudo considerados "cajas negras", pero Random Forest nos da una métrica útil de importancia de variables.
- **Idea intuitiva:** Si una variable es importante, usarla en las divisiones debería mejorar la "pureza" de los nodos del árbol.
- **Mean Decrease in Impurity (MDI):**
  - Se calcula para cada variable la reducción total del criterio de división (Entropía para clasificación, RSS para regresión) promediada sobre todos los árboles del ensamble.
  - Es rápido de calcular pero puede estar sesgado hacia variables con más categorías.

5. Importancia de Variables en Random Forest

- Los ensambles son a menudo considerados "cajas negras", pero Random Forest nos da una métrica útil de importancia de variables.
- **Idea intuitiva:** Si una variable es importante, usarla en las divisiones debería mejorar la "pureza" de los nodos del árbol.
- **Mean Decrease in Impurity (MDI):**
  - Se calcula para cada variable la reducción total del criterio de división (Entropía para clasificación, RSS para regresión) promediada sobre todos los árboles del ensamble.
  - Es rápido de calcular pero puede estar sesgado hacia variables con más categorías.
- **Permutation Importance (Mean Decrease in Accuracy):**
  - Más robusto. Para cada variable, se permutan aleatoriamente sus valores en el set de OOB y se mide cuánto empeora el error del modelo.

5. Importancia de Variables en Random Forest

- Los ensambles son a menudo considerados "cajas negras", pero Random Forest nos da una métrica útil de importancia de variables.
- **Idea intuitiva:** Si una variable es importante, usarla en las divisiones debería mejorar la "pureza" de los nodos del árbol.
- **Mean Decrease in Impurity (MDI):**
  - Se calcula para cada variable la reducción total del criterio de división (Entropía para clasificación, RSS para regresión) promediada sobre todos los árboles del ensamble.
  - Es rápido de calcular pero puede estar sesgado hacia variables con más categorías.
- **Permutation Importance (Mean Decrease in Accuracy):**
  - Más robusto. Para cada variable, se permutan aleatoriamente sus valores en el set de OOB y se mide cuánto empeora el error del modelo.
  - Una caída grande en el rendimiento indica una variable muy importante.

5. Visualizando la Importancia de Variables

- La importancia de variables nos ayuda a entender qué predictores son más influyentes en nuestro modelo.

5. Visualizando la Importancia de Variables

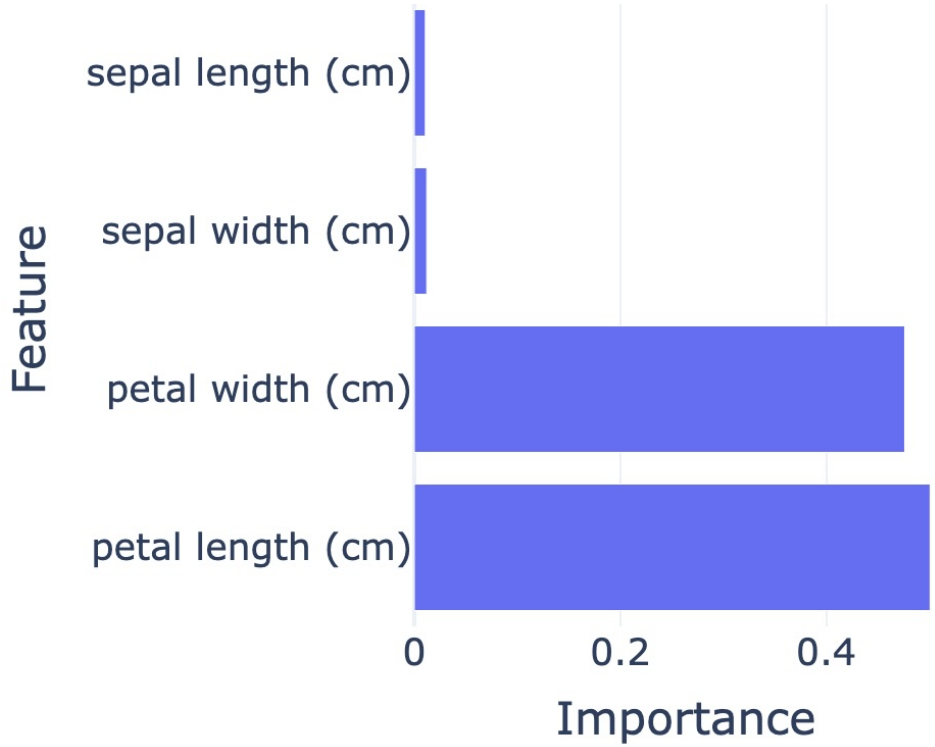
- La importancia de variables nos ayuda a entender qué predictores son más influyentes en nuestro modelo.
- Es una herramienta valiosa para la interpretación del modelo y la selección de características (*feature selection*).

5. Visualizando la Importancia de Variables

- La importancia de variables nos ayuda a entender qué predictores son más influyentes en nuestro modelo.
- Es una herramienta valiosa para la interpretación del modelo y la selección de características (*feature selection*).
- Típicamente se visualiza como un gráfico de barras.

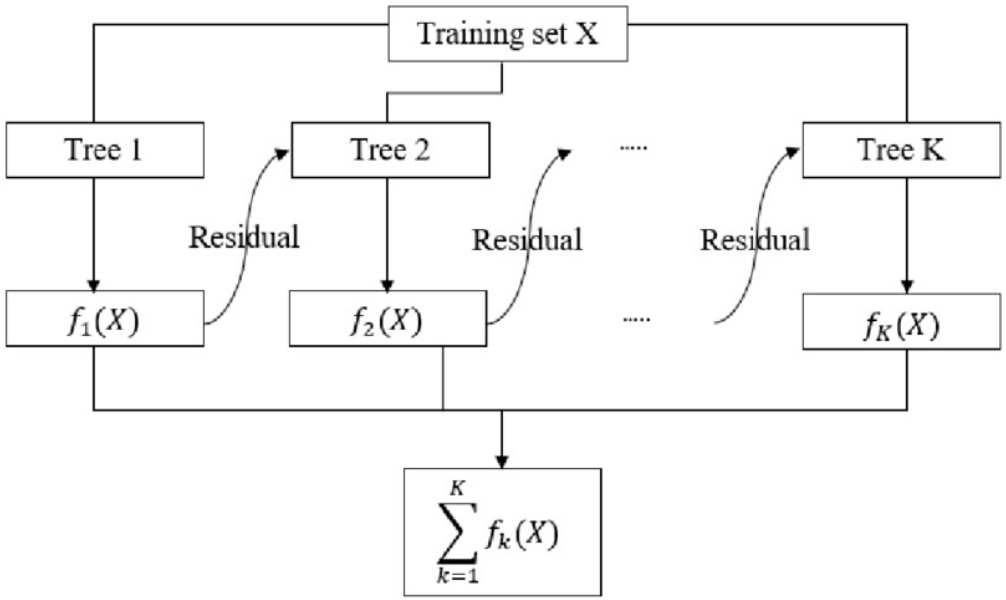
5. Visualizando la Importancia de Variables

- La importancia de variables nos ayuda a entender qué predictores son más influyentes en nuestro modelo.
- Es una herramienta valiosa para la interpretación del modelo y la selección de características (*feature selection*).
- Típicamente se visualiza como un gráfico de barras.



6. Introducción al Boosting

- A diferencia de Bagging (paralelo), **Boosting** es una técnica de ensamble **secuencial**.
- Los modelos se construyen uno tras otro, y cada nuevo modelo se enfoca en corregir los errores cometidos por los modelos anteriores.
- El objetivo principal del Boosting es **reducir el sesgo** del modelo.
- Los árboles en boosting son usualmente muy pequeños (a veces solo un split, llamados *stumps* o tocones).



6. Algoritmos de Boosting

La Intuición Matemática: Descenso de Gradiente

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.
  - En cada iteración, aumenta el peso de las observaciones que fueron mal clasificadas por el modelo anterior.

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.
  - En cada iteración, aumenta el peso de las observaciones que fueron mal clasificadas por el modelo anterior.
  - El siguiente aprendiz se entrena para enfocarse más en estas observaciones "difíciles".

6. Algoritmos de Boosting

La Intuición Matemática: Descenso de Gradiente

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.
  - En cada iteración, aumenta el peso de las observaciones que fueron mal clasificadas por el modelo anterior.
  - El siguiente aprendiz se entrena para enfocarse más en estas observaciones "difíciles".
- **Gradient Boosting:**
  - Un enfoque más generalizado y potente.

6. Algoritmos de Boosting

La Intuición Matemática: Descenso de Gradiente

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.
  - En cada iteración, aumenta el peso de las observaciones que fueron mal clasificadas por el modelo anterior.
  - El siguiente aprendiz se entrena para enfocarse más en estas observaciones "difíciles".
- **Gradient Boosting:**
  - Un enfoque más generalizado y potente.
  - En lugar de re-ponderar observaciones, cada nuevo modelo se entrena para predecir los **residuos** (el error) del modelo anterior.

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.
  - En cada iteración, aumenta el peso de las observaciones que fueron mal clasificadas por el modelo anterior.
  - El siguiente aprendiz se entrena para enfocarse más en estas observaciones "difíciles".
- **Gradient Boosting:**
  - Un enfoque más generalizado y potente.
  - En lugar de re-ponderar observaciones, cada nuevo modelo se entrena para predecir los **residuos** (el error) del modelo anterior.
  - Funciona optimizando una función de pérdida (loss function) mediante descenso de gradiente, de ahí su nombre.

6. Algoritmos de Boosting

La Intuición Matemática: Descenso de Gradiente

- **AdaBoost (Adaptive Boosting):**
  - El primer algoritmo de boosting exitoso.
  - En cada iteración, aumenta el peso de las observaciones que fueron mal clasificadas por el modelo anterior.
  - El siguiente aprendiz se entrena para enfocarse más en estas observaciones "difíciles".
- **Gradient Boosting:**
  - Un enfoque más generalizado y potente.
  - En lugar de re-ponderar observaciones, cada nuevo modelo se entrena para predecir los **residuos** (el error) del modelo anterior.
  - Funciona optimizando una función de pérdida (loss function) mediante descenso de gradiente, de ahí su nombre.

Para el Pizarrón

Veremos cómo el algoritmo de Gradient Boosting es un descenso de gradiente en el espacio de funciones. Esto justifica por qué se entrena sobre los "pseudo-residuos" y conecta con conceptos de optimización.

7. Random Forest vs. Boosting: Un Duelo de Titanes

Random Forest

Reduce la Varianza

- Entrena árboles profundos en paralelo.
- Robusto al sobreajuste. Aumentar el número de árboles no suele sobreajustar.
- Menos parámetros que tunear (principalmente m y B).
- Computacionalmente más sencillo de paralelizar.

Boosting

Reduce el Sesgo

- Entrena árboles superficiales de forma secuencial.
- Propenso al sobreajuste si el número de árboles es muy alto.
- Más sensible a los hiperparámetros (tasa de aprendizaje λ, número de árboles B, profundidad d).
- Puede alcanzar un rendimiento superior si se calibra cuidadosamente.

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".
  - Ideal cuando se necesita una solución rápida y fiable con menos esfuerzo de calibración.

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".
  - Ideal cuando se necesita una solución rápida y fiable con menos esfuerzo de calibración.
- **Usar Boosting para exprimir el rendimiento:**

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".
  - Ideal cuando se necesita una solución rápida y fiable con menos esfuerzo de calibración.
- **Usar Boosting para exprimir el rendimiento:**
  - Cuando el objetivo es la máxima precisión posible y se dispone de tiempo para una calibración cuidadosa de hiperparámetros.

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".
  - Ideal cuando se necesita una solución rápida y fiable con menos esfuerzo de calibración.
- **Usar Boosting para exprimir el rendimiento:**
  - Cuando el objetivo es la máxima precisión posible y se dispone de tiempo para una calibración cuidadosa de hiperparámetros.
  - Muy popular en competiciones de Machine Learning (e.g., Kaggle).

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".
  - Ideal cuando se necesita una solución rápida y fiable con menos esfuerzo de calibración.
- **Usar Boosting para exprimir el rendimiento:**
  - Cuando el objetivo es la máxima precisión posible y se dispone de tiempo para una calibración cuidadosa de hiperparámetros.
  - Muy popular en competiciones de Machine Learning (e.g., Kaggle).
  - Requiere validación cruzada para encontrar el número óptimo de árboles y la tasa de aprendizaje para evitar el sobreajuste.

7. ¿Cuándo Usar Cada Uno?

- **Empezar con Random Forest:**
  - Es una excelente primera opción. Es robusto, fácil de usar y a menudo proporciona un rendimiento muy bueno "de fábrica".
  - Ideal cuando se necesita una solución rápida y fiable con menos esfuerzo de calibración.
- **Usar Boosting para exprimir el rendimiento:**
  - Cuando el objetivo es la máxima precisión posible y se dispone de tiempo para una calibración cuidadosa de hiperparámetros.
  - Muy popular en competiciones de Machine Learning (e.g., Kaggle).
  - Requiere validación cruzada para encontrar el número óptimo de árboles y la tasa de aprendizaje para evitar el sobreajuste.
- **No hay una respuesta única:** La mejor elección siempre dependerá del problema específico y los datos. ¡La experimentación es clave!

8. Aplicaciones Prácticas y Casos de Uso

- **Finanzas:** Detección de transacciones fraudulentas, evaluación de riesgo crediticio (*credit scoring*).

8. Aplicaciones Prácticas y Casos de Uso

- **Finanzas:** Detección de transacciones fraudulentas, evaluación de riesgo crediticio (*credit scoring*).
- **Biomedicina:** Clasificación de tumores a partir de datos genéticos, diagnóstico de enfermedades.

8. Aplicaciones Prácticas y Casos de Uso

- **Finanzas:** Detección de transacciones fraudulentas, evaluación de riesgo crediticio (*credit scoring*).
- **Biomedicina:** Clasificación de tumores a partir de datos genéticos, diagnóstico de enfermedades.
- **E-commerce y Marketing:** Sistemas de recomendación, predicción de la tasa de cancelación de clientes (*churn rate*).

8. Aplicaciones Prácticas y Casos de Uso

- **Finanzas:** Detección de transacciones fraudulentas, evaluación de riesgo crediticio (*credit scoring*).
- **Biomedicina:** Clasificación de tumores a partir de datos genéticos, diagnóstico de enfermedades.
- **E-commerce y Marketing:** Sistemas de recomendación, predicción de la tasa de cancelación de clientes (*churn rate*).
- **Visión por Computadora:** Detección de objetos en imágenes (fue un uso histórico clave para AdaBoost).

8. Aplicaciones Prácticas y Casos de Uso

- **Finanzas:** Detección de transacciones fraudulentas, evaluación de riesgo crediticio (*credit scoring*).
- **Biomedicina:** Clasificación de tumores a partir de datos genéticos, diagnóstico de enfermedades.
- **E-commerce y Marketing:** Sistemas de recomendación, predicción de la tasa de cancelación de clientes (*churn rate*).
- **Visión por Computadora:** Detección de objetos en imágenes (fue un uso histórico clave para AdaBoost).
- **Ecología:** Clasificación de tipos de terreno a partir de imágenes satelitales.

8. Aplicaciones Prácticas y Casos de Uso

- **Finanzas:** Detección de transacciones fraudulentas, evaluación de riesgo crediticio (*credit scoring*).
- **Biomedicina:** Clasificación de tumores a partir de datos genéticos, diagnóstico de enfermedades.
- **E-commerce y Marketing:** Sistemas de recomendación, predicción de la tasa de cancelación de clientes (*churn rate*).
- **Visión por Computadora:** Detección de objetos en imágenes (fue un uso histórico clave para AdaBoost).
- **Ecología:** Clasificación de tipos de terreno a partir de imágenes satelitales.

Resumen y Próximos Pasos

- Los **ensambles** combinan modelos débiles para crear modelos fuertes y estables.

Resumen y Próximos Pasos

- Los **ensambles** combinan modelos débiles para crear modelos fuertes y estables.
- **Random Forest** utiliza bagging y subconjuntos de características para crear árboles decorrelacionados, reduciendo principalmente la **varianza**.

Resumen y Próximos Pasos

- Los **ensambles** combinan modelos débiles para crear modelos fuertes y estables.
- **Random Forest** utiliza bagging y subconjuntos de características para crear árboles decorrelacionados, reduciendo principalmente la **varianza**.
- **Boosting** construye modelos secuencialmente, donde cada uno corrige los errores del anterior, reduciendo principalmente el **sesgo**.

Resumen y Próximos Pasos

- Los **ensambles** combinan modelos débiles para crear modelos fuertes y estables.
- **Random Forest** utiliza bagging y subconjuntos de características para crear árboles decorrelacionados, reduciendo principalmente la **varianza**.
- **Boosting** construye modelos secuencialmente, donde cada uno corrige los errores del anterior, reduciendo principalmente el **sesgo**.
- Ambos métodos ofrecen formas de medir la **importancia de las variables**.

Resumen y Próximos Pasos

- Los **ensambles** combinan modelos débiles para crear modelos fuertes y estables.
- **Random Forest** utiliza bagging y subconjuntos de características para crear árboles decorrelacionados, reduciendo principalmente la **varianza**.
- **Boosting** construye modelos secuencialmente, donde cada uno corrige los errores del anterior, reduciendo principalmente el **sesgo**.
- Ambos métodos ofrecen formas de medir la **importancia de las variables**.
- La elección entre RF y Boosting depende del balance entre rendimiento, tiempo de desarrollo y riesgo de sobreajuste.

Resumen y Próximos Pasos

- Los **ensambles** combinan modelos débiles para crear modelos fuertes y estables.
- **Random Forest** utiliza bagging y subconjuntos de características para crear árboles decorrelacionados, reduciendo principalmente la **varianza**.
- **Boosting** construye modelos secuencialmente, donde cada uno corrige los errores del anterior, reduciendo principalmente el **sesgo**.
- Ambos métodos ofrecen formas de medir la **importancia de las variables**.
- La elección entre RF y Boosting depende del balance entre rendimiento, tiempo de desarrollo y riesgo de sobreajuste.
- **A continuación:** ¡Vamos a aplicar estos conceptos en el taller práctico con Jupyter Notebook!