Respuestas a los Ejercicios 16,17,18

Ejercicio 1: Regularización (Ridge, Lasso y ElasticNet)

¿Cómo afecta la regularización a los coeficientes?

Después de probar los tres métodos, vi diferencias bastante claras:

Ridge mantuvo todos los coeficientes pero los hizo más pequeños. Es como si los "suavizara" sin eliminar ninguno completamente. Todas las variables siguieron en el modelo.

Lasso fue más agresivo - eliminó directamente la variable NOX (óxidos nítricos) poniéndola en cero exacto. Esto significa que decidió que esa variable no era útil para predecir precios de casas.

ElasticNet tomó un enfoque intermedio. Redujo los coeficientes pero no fue tan drástico como Lasso. Curiosamente, este método tuvo el mejor rendimiento con MSE de 24.44.

La regularización básicamente evita que el modelo se "obsesione" con los datos de entrenamiento, controlando qué tan grandes pueden ser los coeficientes.

¿Cuándo usar cada método?

Por experiencia en este ejercicio:

Lasso funciona bien cuando:

- Tienes muchas variables y sospechas que varias no sirven
- Se quiere un modelo simple de entender
- Se necesita que el algoritmo seleccione automáticamente las mejores características

Ridge es mejor cuando:

- Se cree que todas tus variables aportan algo
- Se tiene variables que están relacionadas entre sí
- Se quiere estabilidad en las predicciones

ElasticNet resultó ser el ganador porque combina ambos enfoques. Lo recomendaría como primera opción.

Ejercicio 2: Overfitting

¿Cómo detectar overfitting?

En la gráfica quedó bastante claro:

1. **El error de entrenamiento siguió bajando** - el modelo cada vez "aprendía" mejor los datos de entrenamiento
2. **El error de prueba empezó a subir** después de cierto punto - especialmente en grado 10
3. **La diferencia entre ambos creció** - cuando llegamos a polinomio grado 10, el gap fue de unos 38 puntos

Esto es la señal clásica: el modelo memoriza en lugar de aprender patrones generales.

¿Qué hacer para prevenirlo?

Hay varias estrategias que funcionan:

- **Usar regularización** (como vimos en el ejercicio anterior)
- **Validación cruzada** para elegir los mejores parámetros
- **Conseguir más datos** si es posible
- **Simplificar el modelo** - a veces menos es más
- **Early stopping** - parar de entrenar cuando el error de validación empiece a subir

En nuestro caso, un polinomio de grado 2 o 5 habría sido mejor que uno de grado 10.

Ejercicio 3: Gradient Boosting

¿Por qué Gradient Boosting vs Bagging?

Los resultados hablan por sí solos - conseguí 88% de precisión con LightGBM y CatBoost, comparado con XGBoost que dio 85%.

Gradient Boosting tiene estas ventajas:

- Cada modelo nuevo aprende específicamente de los errores del anterior
- Se enfoca en los casos más difíciles
- Generalmente da mejor precisión (como confirmé en Titanic)

Bagging es más simple:

- Los modelos se entrenan en paralelo
- Es más rápido
- Menos riesgo de overfitting

Para este dataset, Gradient Boosting definitivamente fue superior.

¿Qué parámetros son importantes?

Interesantemente, los valores por defecto funcionaron muy bien (88% de precisión), pero si tuviera que ajustar:

XGBoost:

- Número de árboles (n_estimators)
- Profundidad máxima (max_depth)
- Velocidad de aprendizaje (learning_rate)

LightGBM (mi ganador):

- Número de hojas por árbol
- Cantidad mínima de datos por hoja
- Qué fracción de características usar

CatBoost (el otro ganador):

- Número de iteraciones
- Profundidad de árboles
- Velocidad de aprendizaje

Lo bueno es que estos algoritmos vienen bien configurados de fábrica, así que no necesitas ser un experto para obtener buenos resultados.