Universidad San Francisco de Quito

Data Mining

Informe: PSet #5 - Adaboost

Integrantes:

Giselle Cevallos (00325549),

Anahí Andrade (00323313),

Jesús Alarcón (00324826),

Erick Suárez (00325769)

I. RESUMEN:

AdaBoost (Adaptive Boosting) es uno de los métodos de ensamble más influyentes en el ámbito del Machine Learning. Su objetivo principal es convertir clasificadores débiles en modelos fuertes mediante un proceso de aprendizaje secuencial que incrementa el peso de las observaciones difíciles de clasificar. Este enfoque permite mejorar significativamente la precisión del modelo, manejar relaciones no lineales con relativa simplicidad y mantener un nivel moderado de interpretabilidad. AdaBoost ofrece un rendimiento competitivo sin requerir modelos complejos; sin embargo, puede ser sensible al ruido, a la presencia de outliers y a problemas derivados del preprocesamiento inadecuado. En conjunto, constituye una técnica eficaz y versátil para tareas de clasificación, especialmente en contextos de tamaño medio y con recursos computacionales limitados.

II. DESARROLLO

I. Formulación matemática:

El método AdaBoost se fundamenta en la idea de construir un modelo aditivo compuesto por varios clasificadores débiles que, al combinarse, conforman un clasificador fuerte. Formalmente, el modelo final puede expresarse como una suma ponderada de modelos débiles:

$$F(x) = \sum_{t=1}^{T} \alpha_t \, h_t(x)$$

donde cada $h_t(x)$es un clasificador débil y $\alpha_t$representa el peso asociado a su desempeño. AdaBoost busca minimizar la **exponential loss**, definida como:

$$L = \sum_{i} \exp\left(-y_i F(x_i)\right),$$

lo cual permite penalizar fuertemente los errores y dar mayor importancia a las observaciones mal clasificadas. Para ello, AdaBoost actualiza de manera iterativa una distribución de pesos sobre los ejemplos. Inicialmente, todos los datos reciben un peso uniforme. Luego, tras entrenar un clasificador débil h_t, se calcula su error ponderado ϵ_t. A partir de este error se determina su peso de contribución:

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right).$$

Posteriormente, los pesos de las observaciones se ajustan de manera exponencial según si fueron clasificadas correctamente o no:

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t}.$$

Si bien AdaBoost trabaja principalmente sobre exponential loss, es importante contextualizarlo dentro de la familia más amplia del boosting. Otros métodos modernos, como Gradient Boosting, XGBoost, LightGBM y CatBoost, utilizan funciones objetivo como MSE, MAE o Huber, y optimizan el modelo mediante descenso del gradiente en el espacio de funciones. Estos incorporan técnicas adicionales como shrinkage (learning rate), regularización L1/L2, submuestreo de filas y columnas, y control avanzado de la estructura del árbol. En contraste, AdaBoost mantiene una estructura más simple, basada en actualizaciones de peso exponenciales.

II. Algoritmo (alto nivel)

El funcionamiento de AdaBoost sigue un proceso secuencial claramente definido. Se inicia asignando el mismo peso a todas las observaciones del conjunto de entrenamiento. En cada iteración, se entrena un clasificador débil sobre los datos ponderados, se calcula su error y, en función de este, se determina su peso en la combinación final. Los ejemplos mal clasificados

aumentan su peso y son enfatizados en la siguiente iteración. Este proceso continúa hasta completar un número preestablecido de iteraciones o hasta que el rendimiento deje de mejorar. El clasificador final se obtiene mediante el signo de la suma ponderada de los clasificadores individuales:

$$H(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t \, h_t(x)).$$

III. Hiperparámetros clave y efectos

El desempeño de AdaBoost depende de varios hiperparámetros fundamentales. El primero es **n_estimators**, que indica cuántos clasificadores débiles se incluyen en el ensamble. Un número mayor de estimadores permite capturar patrones más complejos, aunque también incrementa el riesgo de sobreajuste si el conjunto contiene ruido. Otro hiperparámetro crucial es **learning_rate**, que escala la contribución de cada clasificador débil: valores pequeños (0.01–0.1) producen modelos más estables, pero requieren más iteraciones, mientras que valores grandes pueden conducir a un aprendizaje excesivamente agresivo.

El tercer hiperparámetro relevante es **base_estimator**, que suele ser un árbol de decisión con profundidad igual a uno (decision stump). Este tipo de modelo refuerza la filosofía del boosting: combinar modelos muy simples para obtener uno más robusto. Incrementar la profundidad del árbol puede aportar capacidad adicional, pero aumenta considerablemente el riesgo de sobreajuste. También se incluye el hiperparámetro **algorithm**, que permite escoger entre las variantes SAMME y SAMME.R; esta última utiliza probabilidades y ofrece mejor rendimiento en escenarios multiclase. Finalmente, **random_state** garantiza reproducibilidad en los experimentos.

IV. Ventajas y limitaciones

AdaBoost presenta diversas ventajas que explican su impacto histórico y su relevancia actual. Entre las más destacadas se encuentran su capacidad para mejorar modelos débiles, su relativa interpretabilidad y su buen desempeño en datasets pequeños y medianos. La naturaleza secuencial del algoritmo permite corregir iterativamente errores anteriores, lo que contribuye a una mejora acumulativa del rendimiento.

No obstante, también presenta limitaciones importantes. La principal es su alta sensibilidad al ruido y a los outliers, ya que el algoritmo aumenta el peso de los ejemplos mal clasificados, incluso cuando estos corresponden a datos erróneos o atípicos. Además, AdaBoost depende fuertemente de la simplicidad del base learner; modelos más complejos tienden a sobreajustar rápidamente. Por último, aunque es eficiente en términos computacionales para conjuntos pequeños, no compite con algoritmos modernos optimizados para grandes volúmenes de datos.

V. Buenas practicas

Para obtener el mejor rendimiento posible, se recomienda utilizar valores bajos de learning_rate combinados con un número mayor de estimadores, aplicar validación cruzada para evaluar adecuadamente el desempeño y realizar un tratamiento previo adecuado de outliers. En escenarios donde la interpretabilidad sea relevante, es útil examinar las importancias de las características e incluso emplear métodos como SHAP para un análisis más profundo. Asimismo, ajustar la profundidad del árbol base es esencial para controlar la complejidad y evitar el sobreajuste.

VI. Interpretabilidad en AdaBoost

AdaBoost permite interpretar con claridad la importancia de las características porque la estructura basada en stumps facilita: Importancia acumulada según el uso repetido de una variable; y SHAP values, que permiten estimar la contribución marginal de cada característica a la predicción. Esto lo hace adecuado en contextos donde la explicabilidad es crítica, como pricing dinámico, scoring financiero o monitoreo de riesgo.

VII. Casos de uso y pitfalls frecuentes

AdaBoost ha demostrado ser especialmente eficaz en problemas clásicos de clasificación binaria. Uno de sus casos de uso más relevantes es la detección de fraude o anomalías, donde el algoritmo puede identificar patrones sutiles mediante la focalización en ejemplos difíciles. También ha tenido un rol histórico en el reconocimiento facial, específicamente en el algoritmo Viola–Jones, donde la combinación de múltiples clasificadores débiles permitió alcanzar velocidad y precisión en tiempo real. En el ámbito del procesamiento de texto, AdaBoost funciona adecuadamente en tareas de clasificación de spam debido a su capacidad para identificar características discriminantes en espacios de alta dimensionalidad. Asimismo, es útil en sistemas de scoring o evaluación de riesgo, donde se requieren modelos relativamente interpretables.

Sin embargo, su aplicación debe considerar varios riesgos comunes. La sensibilidad inherente del algoritmo al ruido implica que la presencia de outliers puede llevar a un sobreajuste significativo. Además, AdaBoost no es inmune a problemas de **data leakage**, especialmente cuando se utilizan transformaciones globales como normalización antes del train-test split o técnicas de target encoding sin separación por folds. Este tipo de fuga puede inflar artificialmente el rendimiento. También se debe considerar el **target leakage**, que puede ocurrir cuando la información de la etiqueta se filtra indirectamente en el preprocesamiento. Otro riesgo es el **data**

drift: si la distribución de los datos cambia con el tiempo, el modelo puede perder eficacia rápidamente debido a su naturaleza altamente específica. Finalmente, problemas como desbalance de clases, profundidad excesiva del base learner, un número demasiado elevado de estimadores o selecciones inapropiadas de learning_rate pueden deteriorar el desempeño del modelo.

VIII. Tabla de Hiperparámetros más influyentes:

Hiperparámetro	Rango sugerido	Efecto
learning_rate	0.01 – 1	Controla la agresividad del aprendizaje.
n_estimators	50 – 800	Más iteraciones aumentan capacidad; demasiadas generan sobreajuste.
max_depth del stump	1–3	Mantener amplitud del weak learner.
algorithm (SAMME/SAMME.R)	SAMME.R recomendado	Usa probabilidades; generaliza mejor.
random_state	Entero fijo	Reproducibilidad.

IX. Checklist de tuning

El ajuste de AdaBoost debe seguir un orden lógico que permita controlar progresivamente la complejidad del modelo. En primer lugar, se define el base_estimator; lo más recomendable es emplear un decision stump (max_depth=1), ya que mantiene la esencia del boosting. Luego, se ajusta el learning_rate, que debe oscilar entre 0.01 y 0.5 según el nivel de estabilidad requerido. Posteriormente, se selecciona el número adecuado de estimadores, que puede variar entre 50 y 800 dependiendo del learning rate y del tamaño del conjunto de datos.

A continuación, se analiza el impacto de outliers, ya que AdaBoost les asigna cada vez más peso, lo que puede distorsionar el entrenamiento. Finalmente, se revisan las curvas de entrenamiento y validación para detectar posibles señales de sobreajuste; un incremento progresivo en el error de validación o un error de entrenamiento persistentemente alto puede indicar la

necesidad de ajustar hiperparámetros como learning rate, profundidad del árbol o número de estimadores.

X. Resultados

Para validar la eficacia de los métodos de ensemble, se pusieron a prueba 12 modelos distintos utilizando un dataset de 300,000 registros de taxis de NYC, balanceados a razón de 100,000 muestras por año. El objetivo fue predecir la variable total_amount. Para asegurar que los resultados fueran comparables y reproducibles, se mantuvo una semilla aleatoria fija (42) en todas las divisiones de train/validation/test.

Se evaluaron desde baselines simples (Media, Regresión Lineal) hasta arquitecturas más complejas de Boosting (AdaBoost, Gradient Boosting, XGBoost, LightGBM, CatBoost) y combinaciones mediante Voting y Bagging. La métrica principal de decisión fue el RMSE, aunque también se monitorearon el MAE y el tiempo de cómputo para entender el trade-off entre precisión y costo.

A continuación, se resume el comportamiento de los modelos ordenados por su eficacia en el conjunto de validación:

Modelo	RMSE Val	MAE Val	R^2 Val	Tiempo (s)	Observaciones
GradientBoosting	$2.57	$0.90	0.9680	345.07	Mejor balance general.
VotingRegressor	$2.58	$0.96	0.9677	17.64	Muy eficiente.
Pasting	$2.59	$0.94	0.9676	31.68	-
Bagging	$2.59	$0.94	0.9676	26.72	Similar a Pasting.

CatBoost	$2.60	$0.98	0.9673	24.10	Rápido y preciso.
Baseline-LinearReg	$2.72	$1.12	0.9642	0.00	Referencia base.
LightGBM	$2.81	$0.93	0.9618	59.05	-
XGBoost	$3.39	$0.95	0.9444	12.76	Rendimiento inferior al esperado.
AdaBoost	$3.91	$2.19	0.9258	226.35	Alto error en outliers.
Baseline-Mean	$14.44	$8.25	-0.01	0.00	-

Al llevar el modelo ganador (Gradient Boosting) al entorno de Test, se obtuvo un RMSE de $2.71 y un R^2 de 0.9748. Existe una diferencia del 5.6% entre validación y test; si bien indica cierto sobreajuste, está dentro de rangos aceptables para datos con la varianza típica del transporte urbano.

XI. Discusiones

El Gradient Boosting logró el mejor desempeño ($2.57 RMSE), superando al baseline lineal por un margen del 5.52%. Lo interesante no es solo la métrica, sino cómo el modelo gestionó la estructura de los datos. Al analizar los residuos, se observó que este algoritmo fue particularmente hábil corrigiendo los errores en tarifas de media distancia, donde la regresión lineal tendía a infravalorar el costo. Aunque fue el modelo más lento de entrenar (casi 6 minutos), la ganancia en precisión justifica el costo para un sistema de precios donde cada centavo cuenta.

AdaBoost fue el modelo de boosting asignado al equipo, pero curiosamente terminó en la parte baja de la tabla (Posición 9), superando apenas a la media simple. Tras revisar los casos donde AdaBoost falló, se identificó que su alta sensibilidad a los valores atípicos fue su talón de

Aquiles. El dataset de taxis contiene viajes con tarifas inusualmente altas (probablemente errores de registro o viajes muy largos) y, dada la naturaleza de AdaBoost de asignar pesos exponenciales a los errores, el modelo tendió a sesgarse intentando corregir estos casos perdidos, degradando su rendimiento general.

Un hallazgo relevante fue el desempeño del VotingRegressor. Logró un RMSE técnicamente idéntico al mejor modelo ($2.58 vs $2.57) pero en una fracción del tiempo (17s vs 345s). Esto sugiere que, para una implementación en tiempo real donde la latencia importa, un ensamble simple de modelos heterogéneos (Lineal + Árbol + Boosting) constituye una alternativa más pragmática que un Gradient Boosting puro y pesado.

Por otro lado, CatBoost merece una mención especial: obtuvo resultados muy competitivos casi sin ajuste de hiperparámetros, demostrando ser el algoritmo más robusto "out-of-the-box".

El modelo confirmó lo que la intuición sugiere: la tarifa base (base_fare_components) es el predictor dominante. Sin embargo, se encontraron patrones interesantes en las variables rush_hour y borough. Esto valida que el algoritmo no solo memoriza tarifas, sino que entiende el contexto temporal y geográfico (tarifas dinámicas).

Desde una perspectiva de negocio, reducir el error absoluto medio (MAE) a $0.90 implica que es posible predecir el costo de un viaje con una precisión de ±1 dólar en la gran mayoría de los casos, lo cual es suficiente para ofrecer estimaciones de precio cerradas al usuario antes de abordar.

XII. Conclusiones

AdaBoost es un método de boosting fundamental en el desarrollo del Machine Learning moderno debido a su capacidad para transformar clasificadores débiles en modelos robustos mediante un proceso iterativo que enfatiza los ejemplos difíciles de aprender. Su simplicidad conceptual, combinada con un desempeño competitivo, lo convierte en una técnica adecuada para problemas de clasificación binaria y escenarios con conjuntos de datos pequeños o medianos. Sin embargo, su sensibilidad al ruido, la presencia de outliers y los riesgos asociados al preprocesamiento, como el data leakage o el target leakage, requieren una aplicación cuidadosa y un adecuado control de hiperparámetros. En conjunto, AdaBoost sigue siendo una herramienta valiosa dentro del Data Mining, especialmente cuando se busca un equilibrio entre interpretabilidad, eficiencia y capacidad predictiva.

En conclusión, los experimentos confirman que los métodos de ensemble —especialmente Gradient Boosting— ofrecen la mejor precisión para predecir tarifas. Aunque existen diferencias en tiempo de cómputo y sensibilidad a outliers, modelos como VotingRegressor demuestran que es posible lograr un buen equilibrio entre rendimiento y eficiencia. Además, el bajo MAE obtenido evidencia que el sistema puede entregar estimaciones de precio confiables para la mayoría de los viajes. En general, los resultados validan que los ensambles bien seleccionados superan ampliamente a los baselines tradicionales.

XIII. Referencias

Freund, Y., & Schapire, R. E. (1997). *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of Computer and System Sciences, 55(1), 119–139. https://doi.org/10.1006/jcss.1997.1504

Schapire, R. E. (2003). *The boosting approach to machine learning: An overview*. In Nonlinear Estimation and Classification (pp. 149–171). Springer. https://doi.org/10.1007/978-0-387-21579-2_9

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (2nd ed.). Springer.

Zhang, T., & Yu, B. (2005). *Boosting with early stopping: Convergence and consistency*. The Annals of Statistics, 33(4), 1538–1579.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.