UNIVERSIDADE DE BRASÍLIA
Faculdade do Gama

Aprendizado de Máquina

Mini Trabalho 6

Otimização e ajuste fino do sistema

Grupo - 1

Ana Clara Barbosa Borges
André Emanuel Bispo da Silva
Artur Handow Krauspenhar
Gabriel Moura dos Santos
João Artur Leles Ferreira Pinheiro
João Pedro Anacleto Ferreira Machado

Brasília, DF
2025

Introdução

No último minitrabalho da disciplina, testamos a tarefa de classificação em diferentes modelos, avaliando o desempenho de cada um deles. Os modelos avaliados foram: K Nearest Neighbours, XGBoost, Random Forest, Support Vector Machine e Categorical Boosting. Dentre eles, os que geraram os melhores resultados foram Categorical Boosting e Random Forest.

Selecionados os modelos, partimos para a etapa de otimização e ajuste fino dos modelos selecionados, visando os melhores resultados. Pensando nisso, exploramos diferentes combinações de hiperparâmetros usando o RandomizedSearch, que dados algumas opções de parâmetros, seleciona aquela que fará o modelo ter o melhor desempenho. Além disso, aplicamos técnicas de validação cruzada para evitar o sobreajuste, como o StratifiedKFold, que divide os dados em *k* subconjuntos de forma a manter em cada um deles a mesma proporção de rótulos observada no conjunto completo; assim, se há 70 % de instâncias de uma classe e 30 % de outra, cada fold também apresentará essa distribuição, evitando que algum conjunto de treino ou teste seja enviesado, e ao fazer *k* rodadas—cada vez usando um fold distinto para teste e os demais para treino—e depois calcular a média e o desvio das métricas (precisão, recall, F1 etc.).

Nas seções abaixo descrevemos como o processo foi realizado nos modelos Random Forest e Categorical Boosting.

Random Forest

No modelo de random forest optamos para uma busca aleatória com intervalos específicos de hiperparâmetros, rodando em uma das máquinas disponíveis na FCTE, a hu1 com as seguintes especificações.

- Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz com 48 cores e 2 threads por core
- 755 Gib de memória RAM

A alta capacidade de paralelização permitiu a execução de vários testes rapidamente por parte da biblioteca.

Os parâmetros escolhidos para a otimização foram:

- **`n_estimators`**: número de árvores na floresta, variando entre **50 e 200**.
- **`max_depth`**: profundidade máxima de cada árvore, variando entre **1 e 20**.
- **`min_samples_split`**: número mínimo de amostras necessário para dividir um nó interno, variando entre **2 e 20**.

- **`min_samples_leaf`**: número mínimo de amostras necessário em uma folha, variando entre **1 e 20**.

- **`criterion`**: função usada para medir a qualidade da divisão:

    - `gini`: índice de Gini.

    - `entropy`: ganho de informação (entropia).

    - `log_loss`: perda logarítmica.

- **`class_weight`**: estratégia para tratar classes desbalanceadas, com valor fixo:

    - `balanced`: ajusta pesos inversamente proporcionais à frequência das classes.

- **`n_jobs`**: número de jobs em paralelo, com valor fixo:

    - `None`: Usa um único núcleo de CPU (escolhido para que o algoritmo de busca aleatória possa tirar proveito dos núcleos).

Os parâmetros para a busca aleatória foram:

- **n_iter = 20**: número de combinações de parâmetros que serão testadas aleatoriamente (20 buscas diferentes).

- **cv = 5**: número de divisões na validação cruzada .

- **random_state = 42**: semente aleatória usada para garantir reprodutibilidade dos resultados.

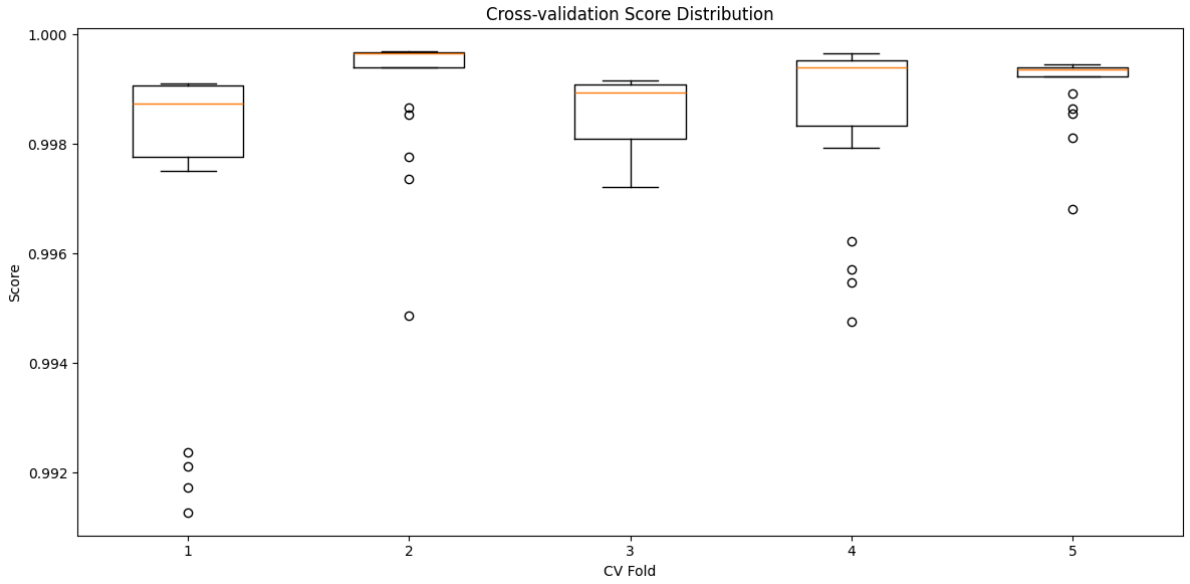
- **n_jobs = -1**: utiliza todos os núcleos disponíveis da CPU para acelerar a busca.

Os resultados foram agregados em um relatório contendo um arquivo em markdown com um resumo da validação cruzada, um gráfico demonstrando a distrbuição da validação cruzada e o modelo serializado e comprimido através do módulo pickle do python. Dessa forma os melhores parâmetros encontrados foram:

- class_weight = 'balanced'

- criterion = 'entropy'
- max_depth = 17
- min_samples_leaf = 4
- min_samples_split = 19
- n_estimators = 153

E aqui os resultados da validação cruzada:



Fonte: André Silva, 2025.

O relatório completo com o modelo serializado e os códigos utilizados podem ser encontrados em conjunto com este documento. O modelo foi comprimido via bzip2 e precisa ser descomprimido antes do uso.

Categorical Boosting

No CatBoost optamos por fornecer intervalos de parâmetros, para que o RandomizedSearch selecionasse parâmetros dentro desses intervalos. A partir disso, o RandomizedSearch testou o modelo com 5 combinações aleatórias dos parâmetros e selecionou a melhor combinação. Após a avaliação, a melhor combinação de parâmetros testada foi:

- Bagging Temperature (Bootstrap Bayesiano): 0.27864646423661144
-
- Border Count (número de divisões): 225

- Depth (profundidade): 4
- Iterations (número de árvores): 412
- L2 Regularization Term (termo de regularização): 2
- Learning Rate (taxa de aprendizado): 0.16928037499514093
- Random Strength (aleatoriedade das divisões): 0.44778316457309164

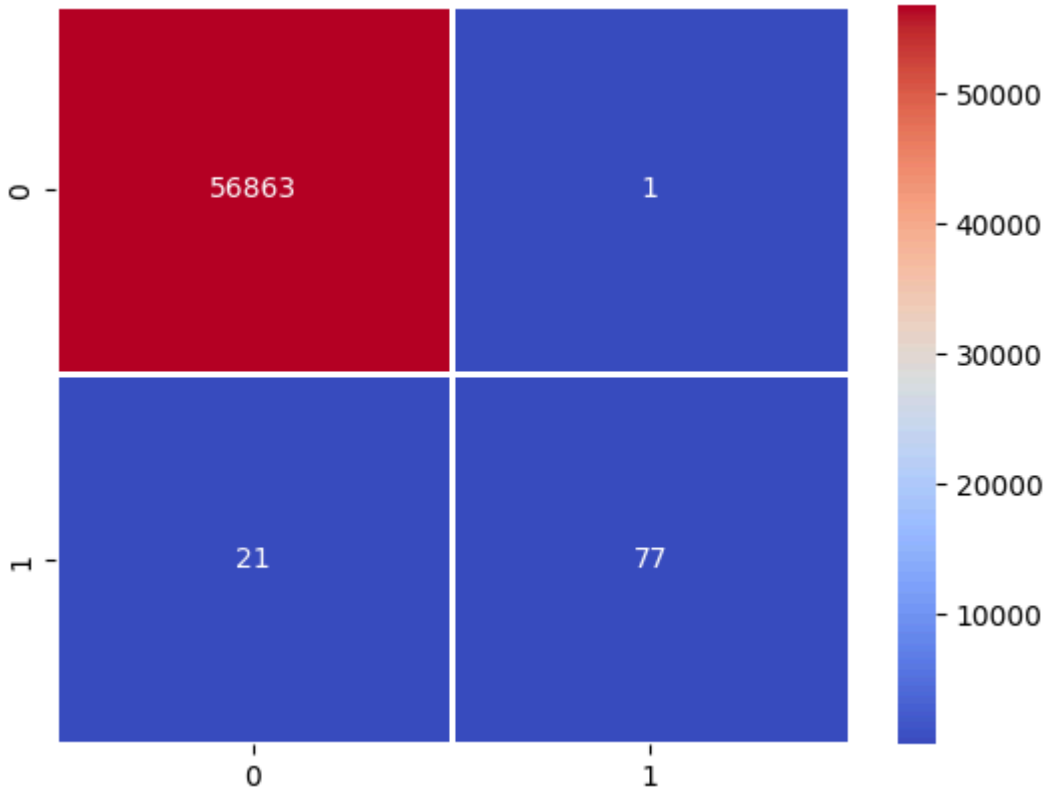
Após rodar os experimentos, obtivemos como F1-score macro 0.94. Sendo o F1-score da classe 0 (transações não fraudulentas) 1.0 e o F1-score da classe 1 (transações fraudulentas) 0.88. Acreditamos que a diferença entre os F1-scores se dá pelo desbalanceamento do nosso dataset, onde a classe 0 aparece diversas vezes. Abaixo é possível ver o relatório do modelo treinado:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	56864
1	0.99	0.79	0.88	98
accuracy			1.00	56962
macro avg	0.99	0.89	0.94	56962
weighted avg	1.00	1.00	1.00	56962

Fonte: Notebook CV_CatBoostCreditCard

Observa-se que a Matriz de Confusão manteve um resultado consideravelmente similar com a entrega anterior tendo uma leve alteração nos falsos negativos, tivemos:

- **Verdadeiros Negativos**: 56.863 transações foram corretamente classificadas como "Não Fraude".
- **Falsos Positivos :** 1 transações legítimas foram erroneamente marcadas como "Fraude" (alarmes falsos).
- **Falsos Negativos (FN):** 21 fraudes reais foram classificadas como "Não Fraude" (erro grave, pois passaram despercebidas).
- **Verdadeiros Positivos (TP):** 77 fraudes foram detectadas corretamente.



Fonte: Notebook CV_CatBoostCreditCard

Conclusão

Por meio da otimização de hiperparâmetros utilizando RandomizedSearchCV e técnicas de validação cruzada, como StratifiedKFold, conseguimos melhorar significativamente a robustez e a acurácia dos modelos. O Random Forest apresentou ótimos resultados com os parâmetros ajustados, como *max_depth = 17* e criterion = 'entropy', enquanto o CatBoost alcançou um F1-score macro de 0.94, Desse modo demonstrando a importância do ajuste fino e da seleção adequada de hiperparâmetros para otimizar o desempenho dos algoritmos.