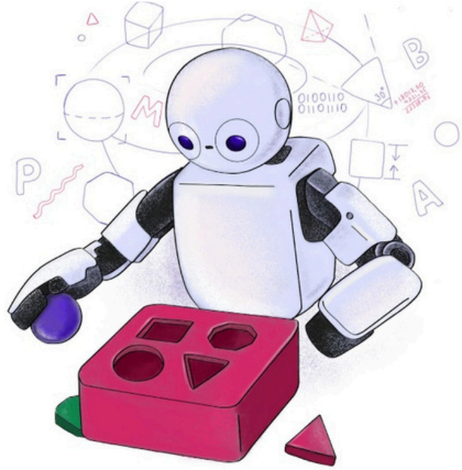
TP558 - Tópicos avançados em Machine Learning: ***AutoGluon-Tabular***



Inatel

Raul Moreno Pereira
raul.pereira@mtel.inatel.br

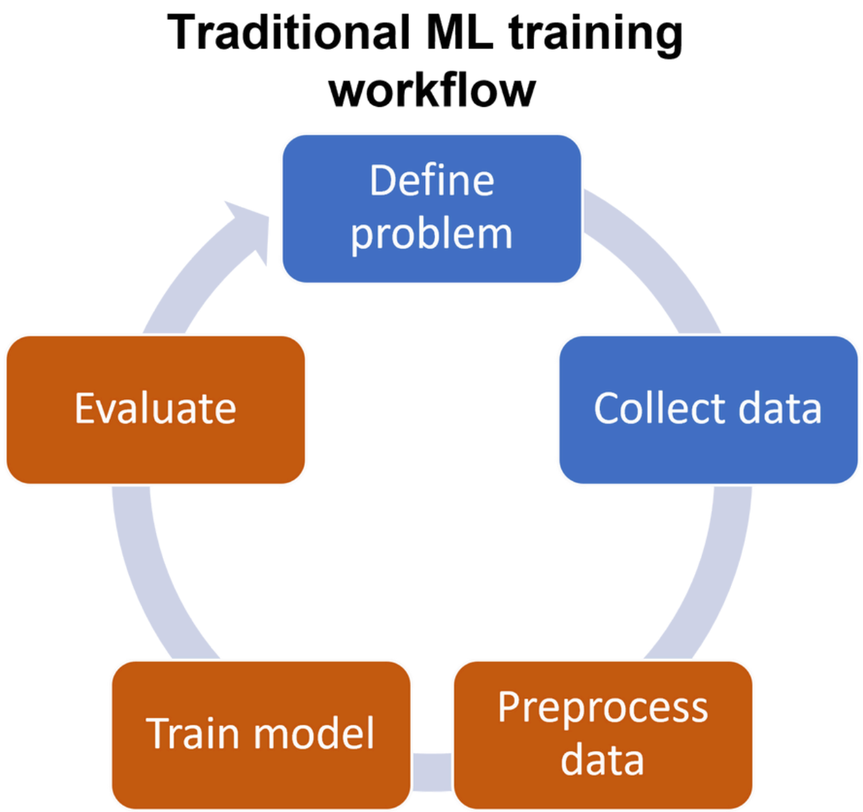
Introdução

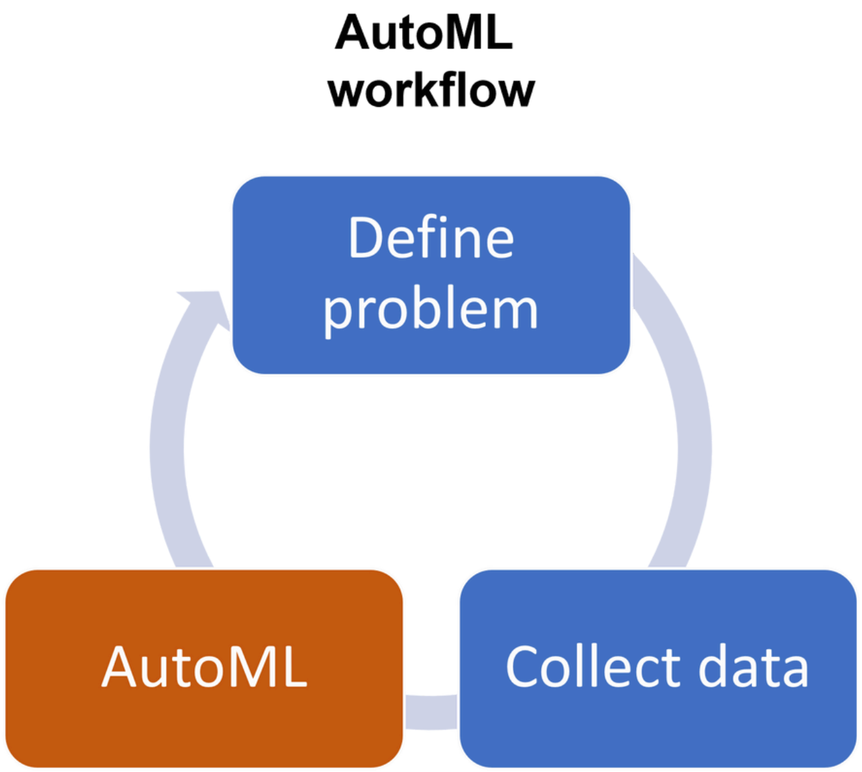
- **Contexto**: Aprendizado de Máquina (ML) cresceu em áreas como saúde, finanças, industria, comércio eletrônico.
- **Desafio**: Complexidade crescente (modelos, pré-processamento, tuning); Até especialistas têm dificuldade em acompanhar todas as técnicas.
- **AutoML**: Automação do pipeline de ML, permitindo treinar modelos de alta qualidade com pouco esforço humano. Isso inclui desde a preparação dos dados e a escolha dos algoritmos até o ajuste de parâmetros e a combinação de modelos.

Introdução

AutoGluon-Tabular:

- Framework de AutoML para dados tabulares (CSV, planilhas, bancos).
- Foco em ensembles multi-camadas, em vez de apenas buscar o “melhor modelo único”.
- Dedica seus recursos a criar uma orquestra de modelos trabalhando em conjunto, conseguindo entregar predições mais precisas e estáveis.
- Em competições Kaggle, superou até 99% dos participantes em poucas horas.
- Basta uma linha de código para iniciar o treinamento a partir de dados crus.
- Também incorpora boas práticas consolidadas na comunidade de ciência de dados.


Traditional ML training workflow
Define problem
Collect data
Preprocess data
Train model
Evaluate

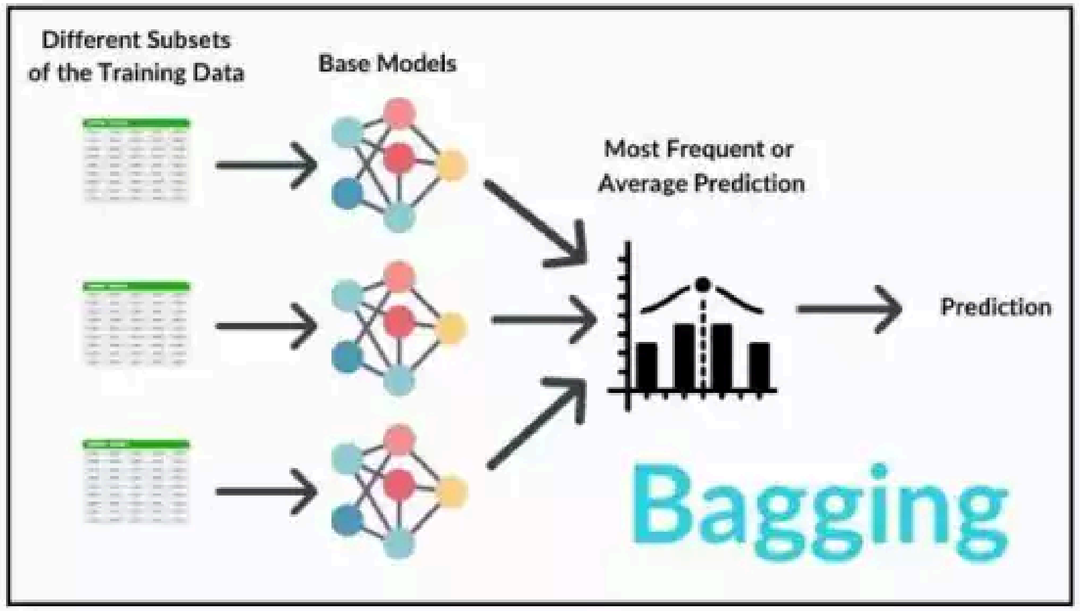

AutoML workflow
Define problem
Collect data
AutoML

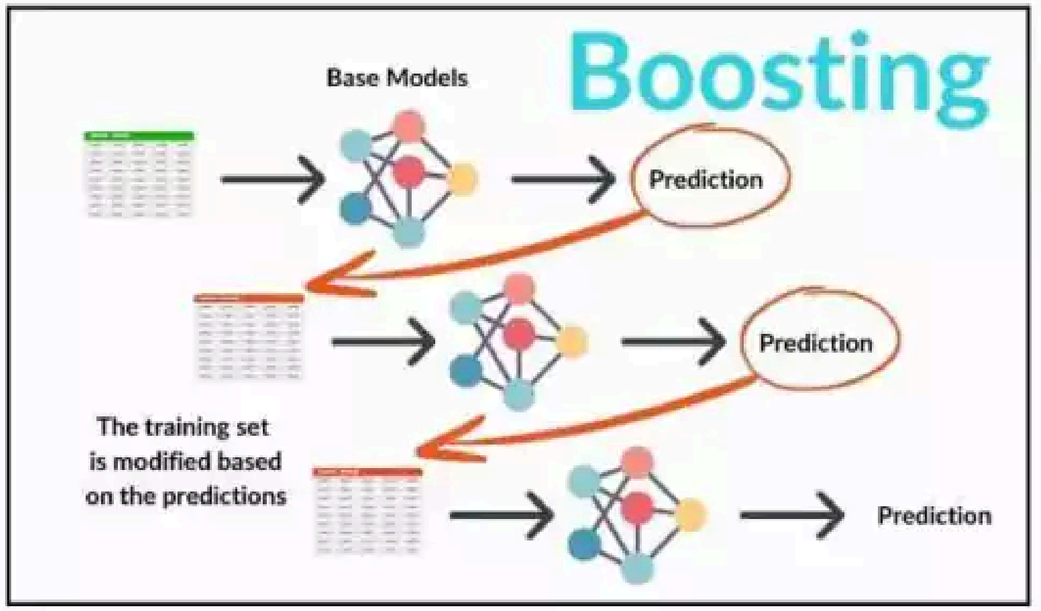
Fundamentação teórica

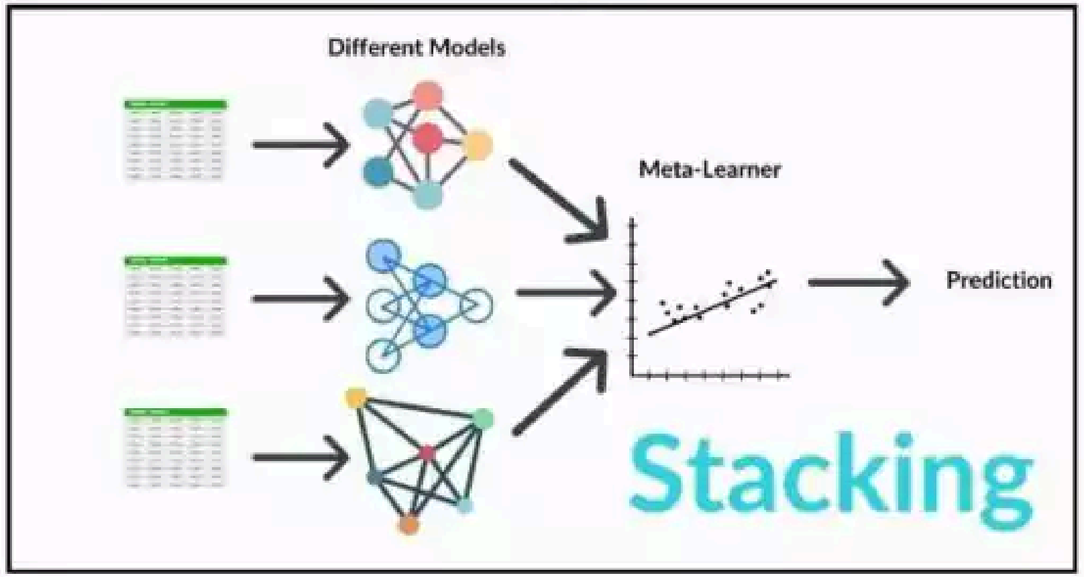
AltoML: Automatiza tarefas de ML (pré-processamento, seleção de modelos, tuning, avaliação).

Ensemble Learning: Combinação de modelos para melhorar performance, onde as falhas de um podem ser compensadas pelos acertos de outro.

- Bagging → cria multiplos subconjuntos dos dados de treino, treina um modelo em cada subconjunto e depois combina os resultados; reduz variância (ex.: Random Forest).
- Boosting → treina modelos de forma sequencial, onde cada novo modelo tenta corrigir os erros cometidos pelos anteriores; reduz viés (ex.: XGBoost, LightGBM, CatBoost).
- Stacking → não apenas junta os resultados dos modelos, mas usa as previsões de vários modelos base como entradas para um novo modelo (meta-modelo), que aprende a combinar essas saídas de forma inteligente.


Different Subsets
of the Training Data
Base Models
Most Frequent or
Average Prediction
Prediction
Bagging


Base Models
Boosting
Prediction
Prediction
The training set
is modified based
on the predictions
Prediction


Different Models
Meta-Learner
Prediction
Stacking

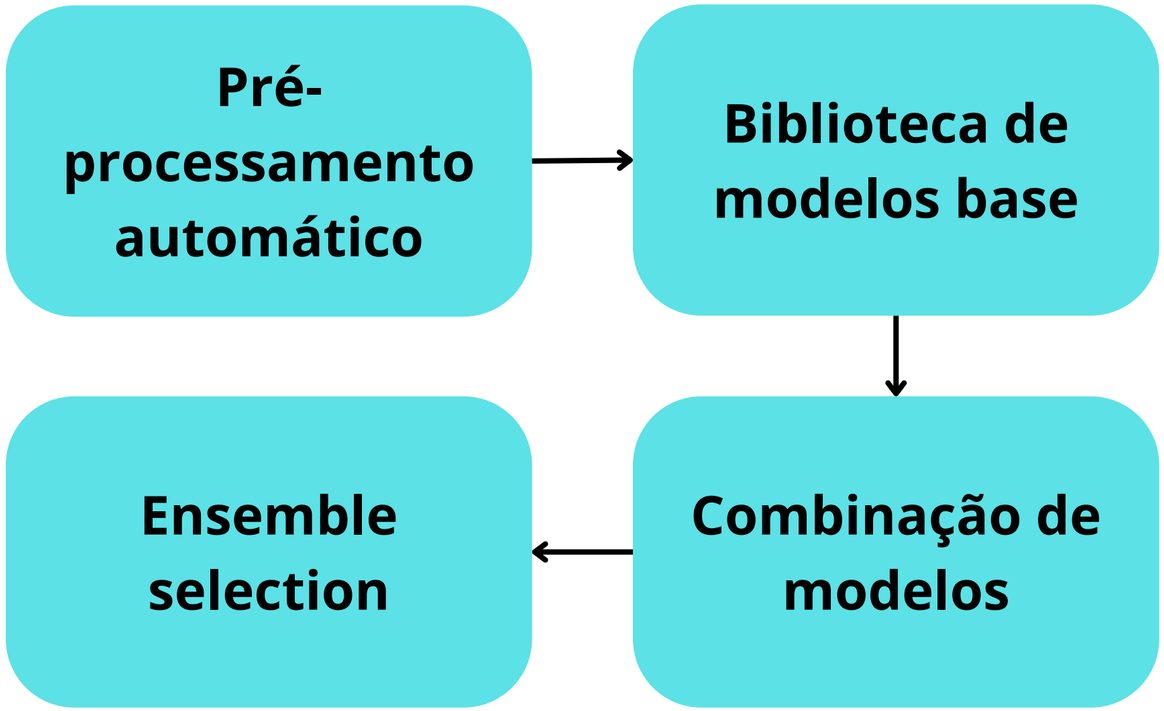
Fundamentação teórica

Diferencial do AutoGluon:

- Introduz o conceito de multi-layer stacking, em que os ensembles não se limitam a uma única camada, mas são construidos em várias camadas sucessivas.
- Em cada camada, as previsões são combinadas com os dados originais (skip connections).
- Isso resulta em uma estrutura parecida com uma rede neural profunda, mas formada por modelos de diferentes naturezas (árvores, redes neurais, KNN, etc.).
- Essa estratégia aumenta a capacidade preditiva e reduz erros sistemáticos.

Arquitetura e funcionamento

Principais etapas do pipeline do AutoGluon:



Arquitetura e funcionamento

Pré-processamento automático:

- Identificação de tipos de variáveis (numéricas, categóricas, texto, datas).
- Tratamento de valores faltantes (ex.: categoria “Unknown”).
- Extração de atributos derivados de textos e datas.

Modelos base:

- Árvores de decisão avançadas (LightGBM, CatBoost), eficientes para lidar com dados heterogêneos.
- Florestas aleatórias e árvores extremamente randomizadas, que trazem diversidade pro ensemble.
- Redes neurais próprias, que aproveitam embeddings para representar variáveis categóricas e camadas densas para processar atributos numéricos.
- Modelos mais simples, como kNN para diversidade.

Arquitetura e funcionamento

Rede Neural própria do AutoGluon:

- Utiliza embeddings, uma representação vetorial aprendida automaticamente, para representar variáveis categóricas de forma compacta.
- Concatena embeddings + variáveis numéricas em camadas totalmente conectadas.
- Passa por uma rede feedforward com camadas densas, Batch Normalization, Dropout e Skip-connections, que tornam o treinamento mais estável e ajuda a evitar problema de overfitting, facilitando também o fluxo do graduente.

Essa rede é integrada como um dos modelos base no ensemble, trazendo maior diversidade e poder preditivo.

Arquitetura e funcionamento

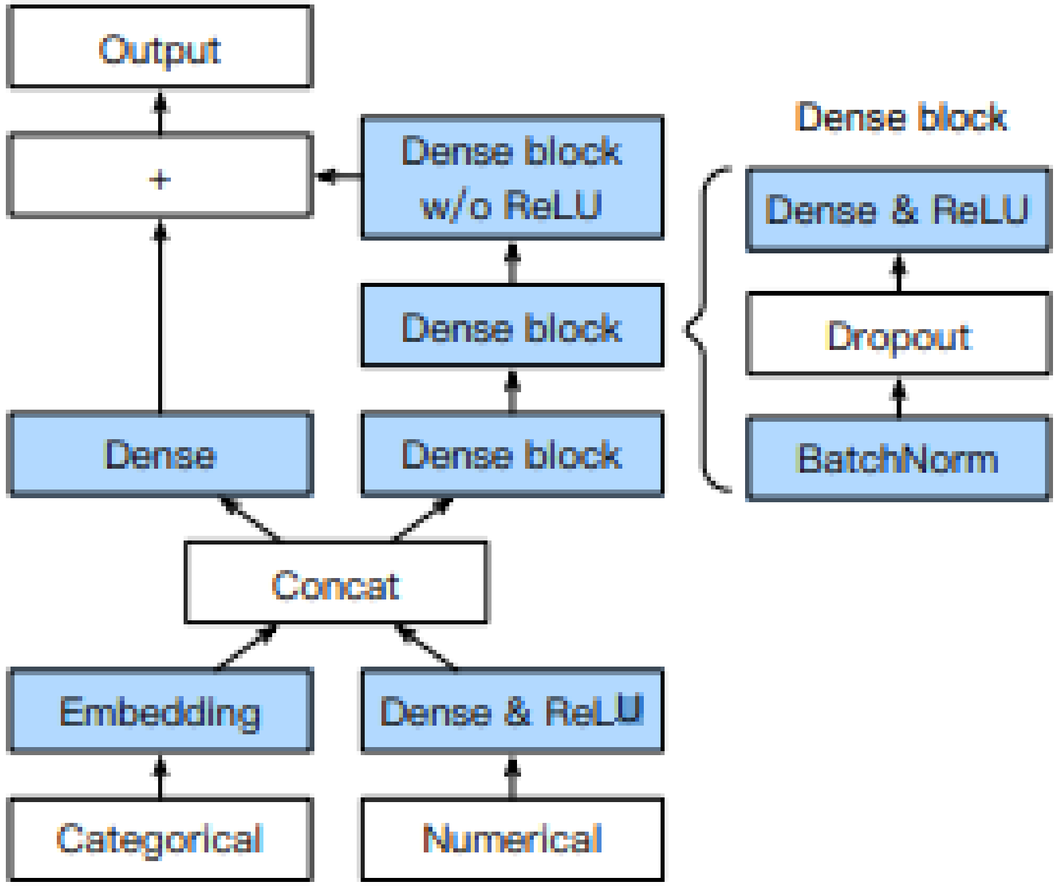


Figure 1. Architecture of AutoGluon's neural network for tabular data composed of numerical and categorical features. Layers with learnable parameters are marked as blue.

Arquitetura e funcionamento

Multi-layer stacking:

- Previsões alimentam camadas seguintes juntos com os dados originais.
- Camada 1 → modelos base treinados em paralelo. Suas previsões são concatenadas e usadas como entrada para a proxima camada (stackers).
- Camada 2 → stackers recebem tanto as previsões quanto os dados originais, funcionando como uma espécie de skip connection.

Ensemble Selection:

- Seleção e combinação ponderada dos melhores modelos para maximizar a acurácia.

Arquitetura e funcionamento

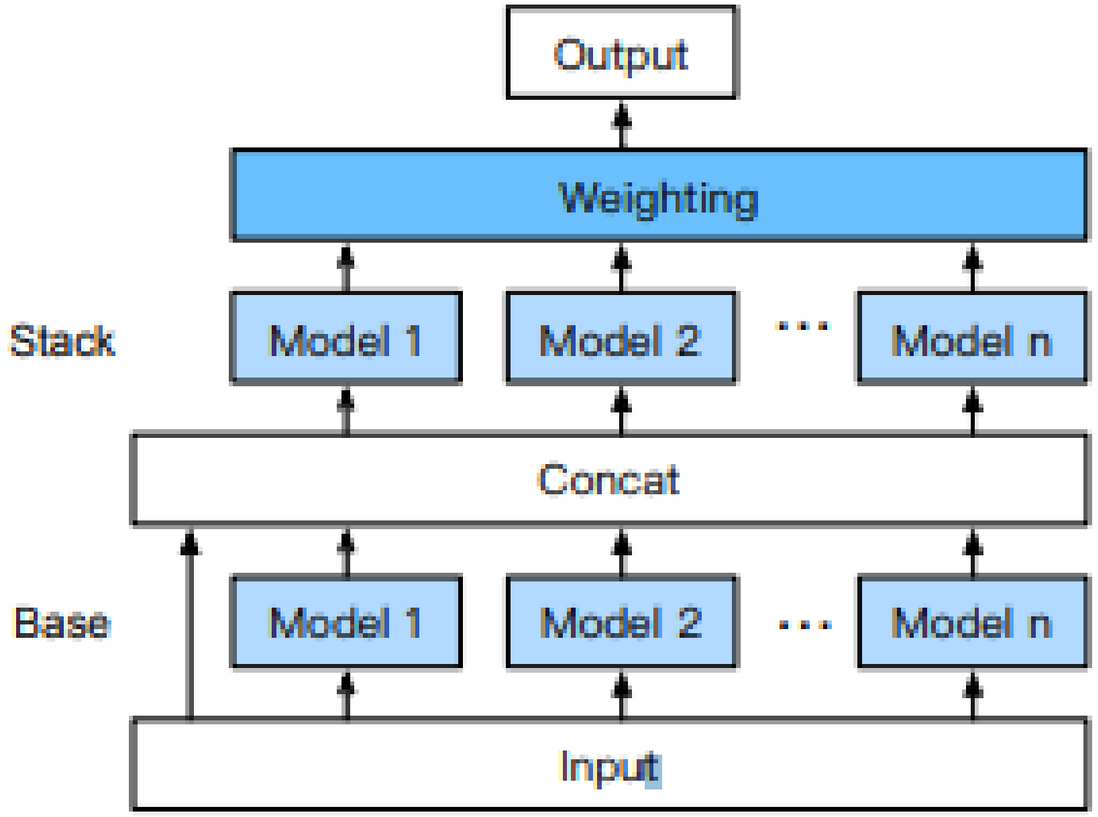


Figure 2. AutoGluon's multi-layer stacking strategy, shown here using two stacking layers and n types of base learners.

Treinamento e otimização

Processo de treinamento foi projetado para equilibrar três objetivos: eficiência computacional, robustez dos modelos e máxima acurácia possível. Busca explorar e combinar modelos diversos de forma inteligente.

Principais mecanismos utilizados:

- K-fold bagging: divide os dados em subconjuntos (folds) e treina multiplas versões de cada modelo, usados para treino e validação.
- Repeated bagging: repete rodadas, mudando as divisões dos dados, para reduzir overffiting e aumentar diversidade.
- Early stopping: interrompe treinos quando não há ganho → economiza tempo e evita overfitting.
- Gestão de tempo e memória: Execução sequencial das camadas (menos falhas de memória que execução paralela) e tempo total distribuído adaptativamente entre camadas.

Treinamento e otimização

Principais mecanismos utilizados:

- Checkpoint: salva progresso → retoma após falhas.
- Hiperparâmetros pré-definidos: conjuntos robustos já validados, dispensando busca exaustiva.

Algorithm 1 AutoGluon-Tabular Training Strategy (multi-layer stack ensembling + n-repeated k-fold bagging).

Require: data (X, Y), family of models $\mathcal{M}$, # of layers L

Preprocess data to extract features
for $l = 1$ **to** L **do** {Stacking}
  for $i = 1$ **to** n **do** {n-repeated}
    Randomly split data into k chunks $\{X^j, Y^j\}_{j=1}^k$
    for $j = 1$ **to** k **do** {k-fold bagging}
      for each model type m in $\mathcal{M}$ **do**
        Train a type-m model on X^{-j}, Y^{-j}
        Make predictions $\hat{Y}_{m,i}^j$ on OOF data X^j
      end for
    end for
  end for
  Average OOF predictions $\hat{Y}_m = \{\frac{1}{n}\sum_i \hat{Y}_{m,i}^j\}_{j=1}^k$
  $X \leftarrow \text{concatenate}(X, \{\hat{Y}_m\}_{m \in \mathcal{M}})$
end for

Vantagens

- **Vantagens**:
  - Simplicidade: em 1 linha de código, identifica os tipos de atributos, trata valores ausentes, aplica técnicas adequadas de pré-processamento e inicia o treinamento;
  - Robustez: lida bem com dados heterogêneos e valores ausentes;
  - Auta acurácia: melhor desempenho que H20 AutoML, Auto-sklearn, TPOT e Google AutoML Tables;
  - Eficiência computacional: evita falhas de memória com execução sequencial.

Desvantagens

- **Desvantagens**:
  - Custo computacional: ensembles multi-camadas demandam mais tempo e espaço em disco do que um modelo único bem otimizado;
  - Baixa interpretabilidade: modelo final funciona como “caixa-preta”, ou seja, é muito bom em prever, mas dificil de interpretar;
  - Limitações em datasets enormes: risco de falhas de memória.

Exemplo(s) de aplicação

Link da aplicação

```
test_data = TabularDataset(f'{data_url}test.csv')

y_pred = predictor.predict(test_data.drop(columns=[label]))
y_pred.head()
```

```
Loaded data from: https://raw.githubusercontent.com/mli/ag-docs/main/knot_theory/test.csv | Columns = 19 / 19 | Rows = 5000 -> 5000
```

	signature
0	-4
1	0
2	0
3	4
4	2

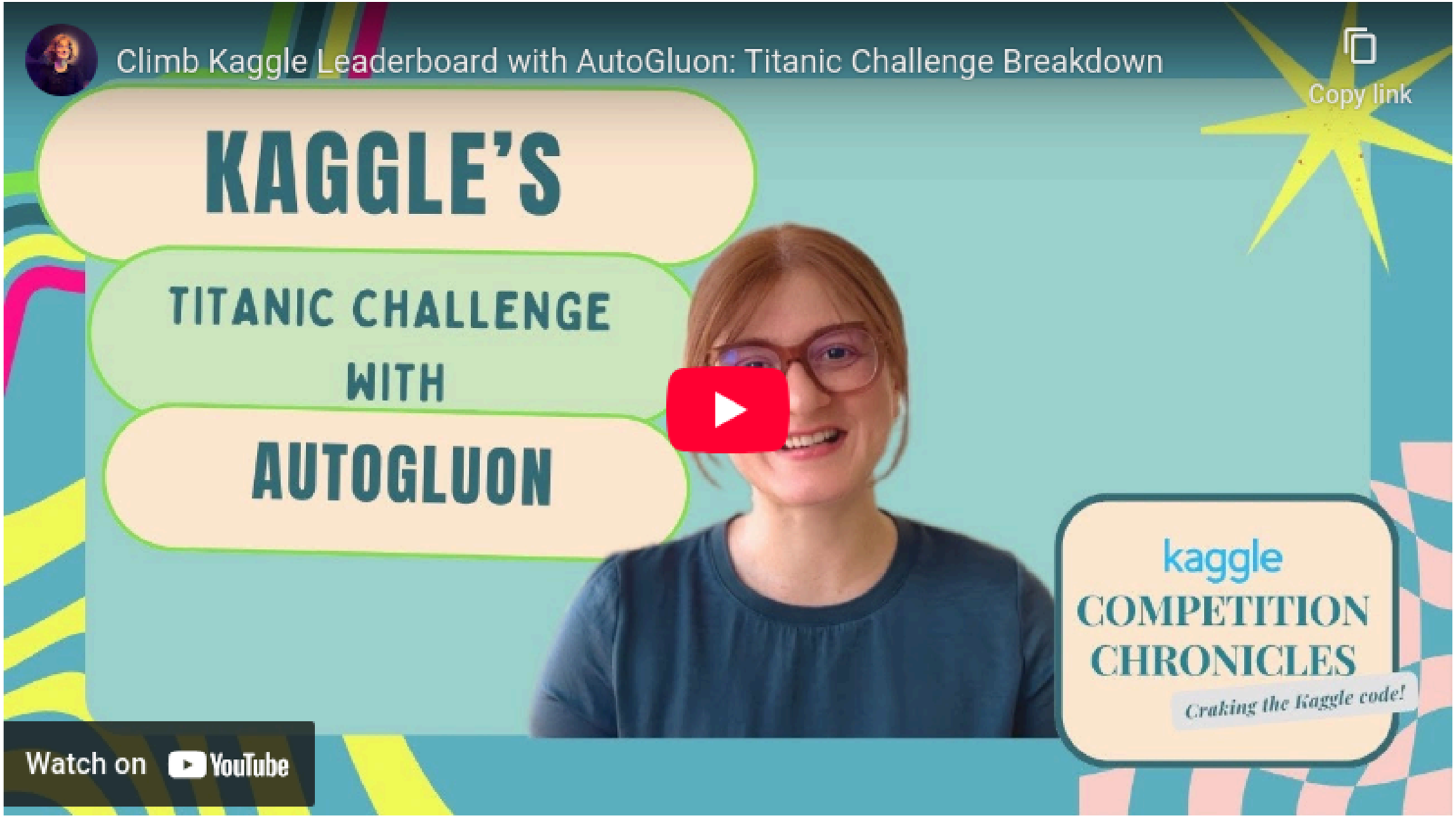
dtype: int64

```
predictor.evaluate(test_data, silent=True)
```

```
{'accuracy': 0.949,
 'balanced_accuracy': np.float64(0.7549527462860195),
 'mcc': np.float64(0.9375019210477166)}
```

```
predictor.leaderboard(test_data)
```

	model	score_test	score_val	eval_metric	pred_time_test	pred_time_val	fit_time	pred_time_test_marginal	pred_ti
0	WeightedEnsemble_L2	0.9490	0.961962	accuracy	2.976920	0.439154	103.052541	0.012196	
1	LightGBM	0.9456	0.955956	accuracy	0.758117	0.144594	8.217093	0.758117	
2	XGBoost	0.9448	0.956957	accuracy	2.780253	0.405104	14.087608	2.780253	
3	LightGBMLarge	0.9444	0.949950	accuracy	2.768586	0.465523	15.391197	2.768586	
4	CatBoost	0.9432	0.955956	accuracy	0.081445	0.012383	70.802043	0.081445	


Climb Kaggle Leaderboard with AutoGluon: Titanic Challenge Breakdown
Copy link
KAGGLE'S
TITANIC CHALLENGE
WITH
AUTOGLUON
kaggle
COMPETITION
CHRONICLES
Craking the Kaggle code!
Watch on YouTube

Exemplo(s) de aplicação

- **Finanças**: Detecção de fraudes em transações e Análise de risco de crédito.
- **Negócios**: Churn prediction (perver perda de clientes).
- **Saúde**:Predição de diagnósticos com base em exames tabulares.
- **E-commerce**: Recomendação de produtos com base em dados de clientes.

Comparação com outros algoritmos

- **Auto-WEKA(2013)**: pioneiro, mas limitado a CASH (algoritmo + hiperparâmetros).
- **Auto-sklearn(2015)**: vencedor de competição AutoML, mas ensembles simples.
- **TPOT(2019)**: usa algoritmos genéticos, mas gera pipelines inválidos → pouco eficiente.
- **H20 AutoML(2019)**: combina modelos e stacking, mas desempenho inferior ao AutoGluon.
- **Google AutoML Tables(2019)**: bom desempenho, mas dependente da nuvem GCP e caro.

Table 2. Comparing each AutoML framework against AutoGluon on the 39 AutoML Benchmark datasets (with 4h training time). Listed are the number of datasets where each framework produced: better predictions than AutoGluon (Wins), worse predictions (Losses), a system failure during training (Failures), or more accurate predictions than all of the other 5 frameworks (Champion). The latter 3 columns show the average: rank of the framework (among the 6 AutoML frameworks applied to each dataset), (rescaled) loss on the test data, and actual training time. Averages are computed over only the subset of datasets/folds where all methods ran successfully. To ensure averaging over datasets remains meaningful, we rescale the loss values for each dataset such that they span $[0, 1]$ among our AutoML frameworks.

Framework	Wins	Losses	Failures	Champion	Avg. Rank	Avg. Rescaled Loss	Avg. Time (min)
AutoGluon	-	-	**1**	**23**	**1.8438**	**0.1385**	201
H2O AutoML	4	26	8	2	3.1250	0.2447	220
TPOT	6	27	5	5	3.3750	0.2034	235
GCP-Tables	5	20	14	4	3.7500	0.3336	**195**
auto-sklearn	6	27	6	3	3.8125	0.3197	240
Auto-WEKA	4	28	6	1	5.0938	0.8001	244

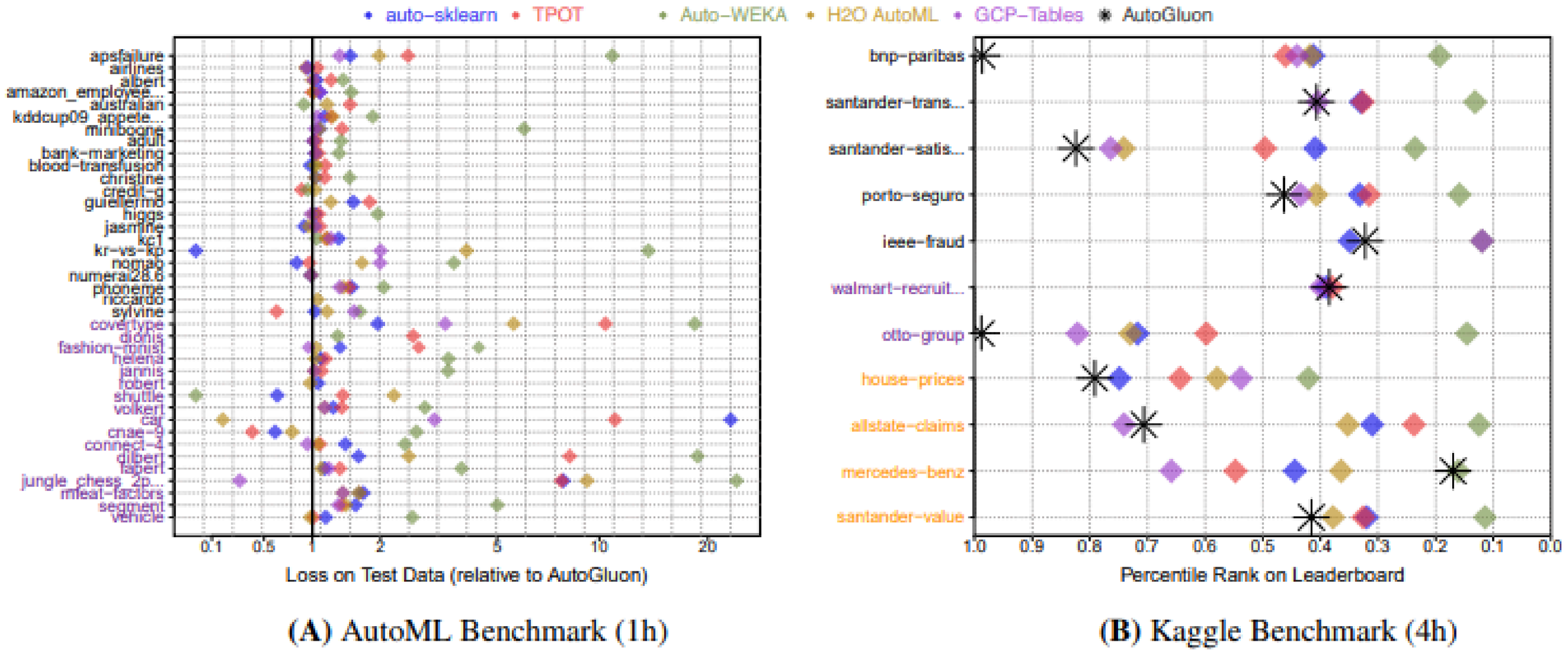
Table 3. Comparing each AutoML framework against AutoGluon on the 11 Kaggle competitions (under 4h time limit). Columns are defined as in Table 2. Instead of loss, we report the percentile rank, i.e. the proportion of teams beaten by AutoML on the competition leaderboard (higher is better). Averages are computed only over the subset of 7 competitions where all methods ran successfully.

Framework	Wins	Losses	Failures	Champion	Avg. Rank	Avg. Percentile	Avg. Time (min)
AutoGluon	-	-	**0**	**7**	**1.7143**	**0.7041**	**202**
GCP-Tables	3	7	1	3	2.2857	0.6281	222
H2O AutoML	1	7	3	0	3.4286	0.5129	227
TPOT	1	9	1	0	3.7143	0.4711	380
auto-sklearn	3	8	**0**	1	3.8571	0.4819	240
Auto-WEKA	0	10	1	0	6.0000	0.2056	221

Comparação com outros algoritmos

- Resultados experimentais:
  - OpenML (39 datasets): AutoGluon foi o melhor em 23 casos com apenas 4h de treino.
  - Kaggle (11 competições): AutoGluon superou rivais em 7.

- Diferenciais:
  - Mais robusto (menos falhas de execução).
  - Melhor aproveitamento do tempo limite.
  - Ensembles multi-camadas únicos entre frameworks.



(A) AutoML Benchmark (1h)

(B) Kaggle Benchmark (4h)

Figure 3. **(A)** Performance of AutoML frameworks relative to AutoGluon on the AutoML Benchmark (with 1h training time). **(B)** Proportion of teams in each Kaggle competition whose scores were beat by each AutoML framework (with 4h training time). Failed runs are not shown in these plots (and we omit a massive loss of Auto-WEKA on `cars` as an outlier). The color of each dataset name indicates the task: binary classification (black), multi-class classification (purple), regression (orange).

Perguntas?

Quiz

Escanear o QRCode para acessar

Link

Referências

- Erickson, Nick & Mueller, Jonas & Shirkov, Alexander & Zhang, Hang & Larroy, Pedro & Li, Mu & Smola, Alexander. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. 10.48550/arXiv.2003.06505.
- GEWARREN. O que é o AutoML (Machine Learning Automatizado)? - ML.NET. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/machine-learning/automated-machine-learning-mlnet>. Acesso em: 19 set. 2025.
- AutoGluon Tabular - Quick Start. Disponível em: <https://auto.gluon.ai/stable/tutorials/tabular/tabular-quick-start.html>. Acesso em: 15 set. 2025.

Obrigado!