

50
anos



Centro de
Informática
UFPE

SAINT

Self-Attention and Intersample Transformer

Uma abordagem inovadora para dados tabulares



Por que SAINT? Motivação e Objetivos

Motivação

Superar as limitações dos modelos de deep learning existentes para dados tabulares

Explorar o potencial dos mecanismos de atenção para capturar relações complexas

Desenvolver uma abordagem que supere métodos clássicos como Gradient Boosting

Criar um modelo que funcione bem mesmo com dados rotulados escassos

Publicado em 2021 por Somepalli, o SAINT busca trazer para dados tabulares o mesmo avanço que os Transformers trouxeram para o PLN.

Objetivos e inovações



Atenção dual

Aplicar atenção tanto sobre colunas (self-attention) quanto sobre linhas (intersample attention)



Embeddings aprimorados

Método especializado para processar features categóricas e numéricas de forma integrada



Pré-treinamento contrastivo

Técnica auto-supervisionada para melhorar o desempenho quando há escassez de dados rotulados



Diferencial

O SAINT não apenas adapta o Transformer para dados tabulares, mas introduz uma abordagem híbrida que combina o poder dos mecanismos de atenção com técnicas específicas para lidar com as particularidades desse tipo de dado, buscando superar tanto métodos clássicos quanto abordagens de deep learning anteriores.

O Desafio dos Dados Tabulares

Características únicas



Features heterogêneas (categóricas e numéricas)



Ausência de informação posicional (ordem arbitrária das colunas)



Correlações complexas entre features sem dependência espacial



Valores ausentes e distribuições variadas

Abordagens tradicionais

Métodos Clássicos

Gradient Boosting (XGBoost, CatBoost, LightGBM), Random Forests



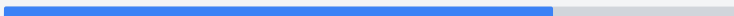
Redes Neurais Convencionais

MLPs, Redes Profundas Convencionais



Modelos Específicos

TabNet, NODE, TabTransformer



O problema

Apesar do sucesso dos métodos clássicos, modelos de deep learning para dados tabulares ainda enfrentam desafios para superar consistentemente abordagens baseadas em árvores, diferentemente do que ocorreu em visão computacional e processamento de linguagem natural.

Arquitetura do SAINT: Visão Geral

Estrutura geral

O SAINT é composto por um empilhamento de L estágios idênticos, cada um contendo:

1 Bloco de Self-Attention

Foca nas interações entre features (colunas) dentro de uma mesma amostra

2 Bloco de Intersample Attention

Foca nas relações entre diferentes amostras (linhas) do conjunto de dados

Cada bloco inclui camadas de normalização (Layer Norm), conexões residuais (skip connections) e redes feed-forward, seguindo a estrutura básica do Transformer original, mas adaptada para dados tabulares.

Dados tabulares de entrada
(Features categóricas e numéricas)

Camada de embedding
Processamento especializado para cada tipo de feature

L Estágios SAINT

Bloco Self-Attention
MSA + FF + LN + Skip Connection

Bloco Intersample Attention
MISA + FF + LN + Skip Connection



Camada de saída (MLP)
Classificação ou Regressão

Hiperparâmetros Principais:

L: Número de estágios empilhados

d: Dimensão dos embeddings

h: Número de cabeças de atenção

Dropout: Taxa para regularização

Processamento: Embedding e Self-Attention

Processamento de features

Features categóricas

- ✓ Cada categoria é mapeada para um vetor de embedding único
- ✓ Valores ausentes são tratados como uma categoria especial (SAINT_NAN)
- ✓ Similar ao processamento de tokens em PLN

Features numéricas

- ✓ Normalização via StandardScaler (por padrão)
- ✓ Projeção para espaço d-dimensional via camada linear com ReLU
- ✓ Valores ausentes são preenchidos com zeros





Token [CLS]

Similar ao BERT, um token especial [CLS] é adicionado ao início da sequência de embeddings. Sua representação final é usada para classificação/regressão.

Self-Attention (MSA)

O bloco de Self-Attention opera sobre os embeddings das features de uma única amostra, permitindo ao modelo aprender quais features são mais relevantes umas para as outras.

Componentes:

-  Multi-Head Self-Attention (MSA) com h cabeças
-  Feed-Forward Network (FFN) com GELU
-  Skip connections (conexões residuais)
-  Layer Normalization (LN)

Diferencial do SAINT:

Ao contrário de outros modelos Transformer para dados tabulares, o SAINT processa tanto features categóricas quanto numéricas de forma integrada através do mecanismo de self-attention, permitindo capturar interações complexas entre todos os tipos de features.

Inovação: Intersample Attention

O conceito

A Intersample Attention (MISA) é uma das principais inovações do SAINT. Enquanto a self-attention tradicional opera sobre features dentro de uma mesma amostra, a intersample attention computa atenção entre diferentes amostras (linhas) do conjunto de dados.

Como funciona:

Os embeddings de todas as amostras no batch são concatenados por feature

A atenção multi-cabeça é aplicada sobre essas amostras para cada dimensão de feature

Isso permite que cada amostra "observe" outras amostras similares

O resultado é uma representação contextual que incorpora informações de outras amostras relevantes



Analogia:

Pode ser vista como uma forma de k-Nearest Neighbors (k-NN) aprendida e integrada à rede neural, permitindo que amostras similares contribuam para a representação umas das outras.

Benefícios



Melhor generalização

Permite que o modelo aprenda padrões mais robustos ao considerar múltiplas amostras simultaneamente



Tratamento de ruído

Quando uma feature é ruidosa ou ausente em uma amostra, o modelo pode "emprestar" informações de amostras similares



Captura de padrões complexos

Identifica relações entre amostras que seriam difíceis de modelar com abordagens tradicionais

Diferencial em relação a outros modelos

Modelo

Tipo de Atenção

TabTransformer Apenas self-attention em features categóricas

TabNet Atenção sequencial para seleção de features

VIME Sem mecanismo de atenção entre amostras

SAINT Self-attention + Intersample attention

Aprendendo Padrões: Pré-treinamento Auto-Supervisionado

Pré-treinamento Contrastivo

Uma das contribuições chave do SAINT é a introdução de um pipeline de pré-treinamento auto-supervisionado projetado especificamente para dados tabulares, especialmente útil quando os dados rotulados são escassos.

Desafio:

Criar "augmentations" (transformações) significativas para dados tabulares é mais desafiador do que para imagens ou texto, onde rotações, cortes ou substituições de palavras são naturais.

Solução do SAINT:



CutMix

Mistura subconjuntos de features entre duas amostras diferentes



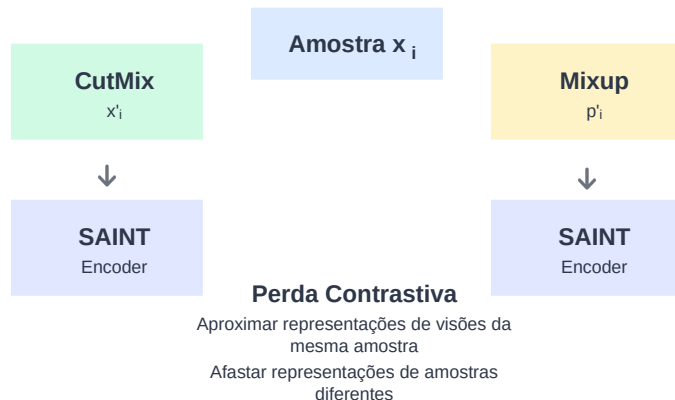
Mixup

Cria novas amostras virtuais através da combinação linear convexa dos embeddings

Objetivo:

Criar duas "visões" ligeiramente diferentes, mas semanticamente relacionadas, da mesma amostra ou de uma mistura de amostras, para treinar o modelo a reconhecer padrões invariantes.

Processo de Aprendizado Contrastivo



Benefícios do Pré-treinamento:

- ✓ Aproveita dados não rotulados para melhorar a representação
- ✓ Melhora significativamente o desempenho em cenários com poucos dados rotulados
- ✓ Força o modelo a focar em características essenciais e invariantes dos dados
- ✓ Inicializa os pesos do modelo de forma mais eficiente antes do fine-tuning

Aprendendo Padrões: Fine-tuning e Representação Final

Fine-tuning Supervisionado

Após o pré-treinamento (ou diretamente, se houver dados rotulados suficientes), o SAINT é treinado de ponta a ponta para minimizar uma função de perda específica da tarefa:

Processo:

Os pesos do modelo são inicializados com os valores do pré-treinamento (se disponível)


A entrada consiste nas features tabulares após embedding


Os dados passam por todos os estágios L do SAINT

A representação final do token [CLS] é usada para a predição

O modelo é otimizado para minimizar o erro na tarefa específica

Funções de Perda:

 **Classificação:** Cross-Entropy, Focal Loss

 **Regressão:** MSE, MAE, Huber Loss

Formação da Representação

Durante o treinamento, os blocos de atenção do SAINT refinam progressivamente os embeddings das features. As matrizes de atenção aprendidas em cada camada refletem os padrões descobertos:

Padrões Intra-Amostra

Os pesos de self-attention indicam quais features são co-dependentes ou preditivas umas das outras dentro de uma linha.

Padrões Inter-Amostras

Os pesos de intersample attention revelam quais amostras são consideradas similares ou relevantes entre si.

Visualização de Atenção:

Os autores do SAINT visualizaram as matrizes de atenção para entender como o modelo aprende:



Representação simplificada de um mapa de atenção do SAINT

Onde Usar o SAINT? Aplicações Práticas



Detecção de Fraudes

Em setores financeiros e e-commerce, o SAINT pode identificar padrões sutis de fraude analisando transações e comportamentos de usuários.

A capacidade de capturar relações entre diferentes amostras via intersample attention permite identificar comportamentos anômalos com maior precisão.



Saúde e Genômica

Análise de prontuários eletrônicos, dados de ensaios clínicos ou dados genômicos para diagnóstico e prognóstico.

O SAINTENS, uma variante do SAINT, foi desenvolvido especificamente para biomarcadores digitais usando dados tabulares de saúde do mundo real.



Marketing e CRM

Previsão de churn, segmentação de clientes e personalização de campanhas baseadas em dados comportamentais.

O pré-treinamento contrastivo permite extrair insights mesmo de bases de clientes com poucos dados rotulados sobre conversões.



Finanças e Investimentos

Análise de risco de crédito, previsão de mercado e detecção de anomalias em dados financeiros.

A capacidade de modelar interações complexas entre diferentes indicadores financeiros pode levar a previsões mais acuradas.



Ciência de Materiais

Descoberta de novos materiais ou moléculas através da análise de propriedades químicas e físicas tabulares.

O SAINT pode prever propriedades de materiais com base em suas características estruturais e descritores químicos.



Manufatura e IoT

Manutenção preditiva, controle de qualidade e otimização de processos industriais.

A capacidade de lidar com dados heterogêneos de sensores e equipamentos torna o SAINT adequado para aplicações industriais.

Onde Usar o SAINT? Aplicações Práticas



Quando Considerar o SAINT

- ✓ Quando há interações complexas entre features que métodos tradicionais não capturam
- ✓ Quando há poucos dados rotulados disponíveis (aproveitando o pré-treinamento)
- ✓ Quando a interpretabilidade não é o requisito principal
- ✓ Quando há recursos computacionais suficientes para treinamento

Relatório - Benchmarks

Benchmarks: modelos de gradient boosting

- ✓ **LightGBM:** velocidade e eficiência extremas. Utiliza algoritmos baseados em histogramas e uma estratégia de crescimento de árvore "leaf-wise" (folha a folha), que resulta em convergência mais rápida e menor uso de memória.
- ✓ **XGBoost:** uma das bibliotecas de gradient boosting mais populares e bem-sucedidas, famosa por seu desempenho vencedor em competições. Oferece alta performance, opções avançadas de regularização para evitar over fitting e grande escalabilidade.
- ✓ **CatBoost:** tratamento sofisticado e nativo de features categóricas. Utiliza a "ordered boosting" e um método eficiente de "target encoding" para lidar com variáveis categóricas sem a necessidade de pré-processamento manual extensivo, como o one-hot encoding.

O que é AutoML?

Processo de automação das tarefas complexas e iterativas envolvidas na construção de um modelo de machine learning.

✓ **AutoGluon:**

Biblioteca de AutoML da Amazon (AWS) focada em simplicidade e robustez. O AutoGluon se destaca por treinar múltiplos modelos em paralelo e criar automaticamente ensembles avançados (empilhamento de modelos), que frequentemente resultam em desempenho de ponta com poucas linhas de código.

✓ **auto-sklearn 2.0 (ASKL 2.0):**

Ferramenta clássica de AutoML construída sobre o ecossistema **Scikit-learn**. Ela utiliza otimização Bayesiana para explorar de forma inteligente o vasto espaço de possíveis algoritmos e suas configurações de hiperparâmetros, construindo automaticamente um ensemble com os melhores modelos encontrados durante a busca.

Relatório - Framework de Execução

Prevenção de vazamento de dados (Data Leakage)

O princípio fundamental da metodologia foi o isolamento rigoroso do conjunto de teste. Para cada um dos 30 datasets, os dados foram divididos **uma única vez** no início do processo. O conjunto de teste (30%) foi imediatamente separado e não foi utilizado em nenhuma etapa de treinamento, pré-processamento ou otimização de hiperparâmetros, sendo acessado apenas na avaliação final.

Framework de execução: passo a passo

Esta etapa inicial é idêntica para os três scripts.

1. **Divisão 70/30:** Os dados são separados em 70% para treino e 30% para teste, utilizando a semente aleatória fixa para garantir que todos os modelos vejam as mesmas partições.

```
# Divisao de dados inicial e unificada
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.3,
    random_state=42,
    stratify=y
)
```

Listing 2: Divisão de dados unificada (70/30, seed 42).

Framework de execução: passo a passo

2. **Criação do Pipeline:** Um pipeline do Scikit-learn é construído para tratar os dados, com etapas de imputação de valores ausentes e normalização/codificação de features. Crucialmente, este pipeline é **ajustado (fit) apenas com os dados de treino**.

```
# Define transformadores para colunas numericas e categoricas
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# O preprocessor e ajustado apenas em X_train
preprocessor = ColumnTransformer(transformers=[
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])
```

As constantes a seguir foram aplicadas de forma consistente em todos os frameworks para assegurar que as condições de teste fossem idênticas.

```
# Constantes aplicadas a todos os frameworks
TEST_SIZE = 0.3
RANDOM_SEED = 42
N_TRIALS = 30      # Para otimizacao com Optuna
NUM_FOLDS = 10     # Para validacao cruzada
```

Relatório - Treinamento e otimização

Modelos manuais

(total_hyper_comparator.py)

Para o SAINT, LightGBM, XGBoost e CatBoost, o script utiliza a biblioteca Optuna para a busca de hiperparâmetros. A função objetivo do Optuna avalia cada configuração candidata através de uma validação cruzada de 10 folds.

```
def objective(trial, X, y):  
    # Sugestao de hiperparametros pelo Optuna  
    params = {'learning_rate': trial.suggest_float('learning_rate',  
0.01, 0.3), ...}  
  
    # Validacao cruzada (10 folds) APENAS no conjunto de treino  
    skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)  
    scores = []  
    for train_idx, val_idx in skf.split(X, y):  
        # ... treina e avalia no fold de validacao interno  
        scores.append(score)  
  
    return np.mean(scores)  
  
# Otimizacao e executada passando apenas os dados de treino  
study.optimize(lambda trial: objective(trial, X_train, y_train),  
n_trials=30)
```

Listing 4: Otimização com Optuna e validação cruzada de 10 folds.

Relatório - AutoGluon e Auto-sklearn

AutoML - AutoGluon (autogluon_comparator.py): Este script invoca o AutoGluon, que recebe apenas os dados de treino e gerencia todo o processo internamente.

```
# Instancia e treina o preditor apenas com os dados de treino
predictor = TabularPredictor(label=label_column, eval_metric='
roc_auc_ovo')
predictor.fit(train_data) # train_data contem apenas os 70%
```

Listing 5: Execução do AutoGluon.

AutoML - auto-sklearn 2.0 (askllm.py): De forma similar, o script do auto-sklearn instancia o classificador e o treina somente nos dados de treino.

```
import autosklearn.classification

automl = autosklearn.classification.AutoSklearnClassifier(
    time_left_for_this_task=3600,
    seed=42
)
# O ajuste (fit) e feito apenas nos dados de treino
automl.fit(X_train_processed, y_train)
```

Listing 6: Execução do auto-sklearn 2.0.

Relatório - Tempo médio de execução

Table 1: Tempo médio de execução (em segundos) por dataset.

Modelo	Tempo Médio (s)
LightGBM	0.45
XGBoost	0.61
CatBoost	1.01
SAINT	5.86
auto-sklearn 2.0	35.87
AutoGluon	42.16

Relatório - Resultados detalhados

Table 3: Resultados detalhados com todos os competidores (Conjunto de Teste)

Dataset	SAINT			AutoGluon			auto-sklearn 2.0			CatBoost			LightGBM			XGBoost		
	ACC	AUC	CE	ACC	AUC	CE	ACC	AUC	CE	ACC	AUC	CE	ACC	AUC	CE	ACC	AUC	CE
11	.944	.014	.154	.944	.160	.202	.928	.091	.230	.460	.322	.955	.832	.214	.426	.912	.107	.302
15	.957	.993	.122	.964	.997	.077	.971	.996	.085	.957	.991	.116	.964	.992	.104	.971	.991	.106
23	.583	.255	.884	.590	.280	.872	.566	.254	.899	.364	.257	.928	.532	.272	.935	.559	.246	.878
29	.870	.928	.376	.891	.962	.256	.891	.957	.267	.884	.933	.554	.884	.946	.290	.884	.949	.278
31	.700	.804	.575	.755	.842	.458	.735	.825	.472	.700	.785	.667	.705	.784	.511	.735	.813	.486
37	.766	.840	.480	.832	.902	.385	.818	.899	.400	.799	.876	.472	.792	.856	.456	.812	.878	.427
50	.990	.998	.047	.990	1.000	.027	.990	.998	.049	.677	.817	.588	.990	1.000	.022	.979	.990	.117
54	.776	.060	.452	.765	.134	.468	.765	.090	.478	.249	.090	.945	.706	.084	.529	.747	.081	.532
188	.649	.127	1.579	.676	.243	.723	.649	.168	.773	.265	.177	1.298	.622	.102	.774	.622	.114	.806
307	.758	.023	.826	.970	.077	.117	.939	.020	.205	.091	.033	1.341	.944	.002	.199	.859	.010	.498
458	.994	.000	.016	.994	.500	.018	.988	.000	.027	.320	.000	.736	.982	.000	.034	.959	.002	.144
469	.194	.437	2.007	.192	.454	1.898	.183	.441	1.964	.169	.439	1.813	.181	.436	1.811	.175	.452	2.207
1049	.928	.931	.209	.918	.959	.165	.918	.957	.172	.877	.857	.405	.904	.945	.187	.911	.947	.186
1050	.898	.856	.246	.904	.904	.220	.901	.891	.224	.898	.862	.254	.898	.855	.285	.895	.873	.235
1063	.886	.878	.314	.914	.927	.254	.895	.903	.279	.895	.877	.344	.895	.896	.310	.886	.895	.282
1068	.928	.862	.201	.932	.865	.187	.932	.844	.192	.932	.705	.586	.932	.840	.202	.932	.843	.193
1462	1.000	1.000	.000	1.000	1.000	.000	1.000	1.000	.001	.858	.945	.557	.996	1.000	.010	.993	1.000	.069
1464	.753	.740	.518	.760	.781	.468	.747	.752	.487	.760	.701	.629	.753	.731	.498	.747	.723	.503
1468	.866	.014	.479	.880	.063	.324	.873	.029	.343	.111	.027	1.946	.801	.025	.708	.847	.021	.471
1480	.718	.734	.527	.709	.781	.495	.701	.772	.506	.684	.751	.510	.667	.716	.539	.684	.692	.575
1494	.872	.928	.548	.872	.939	.283	.872	.932	.294	.801	.891	.650	.853	.926	.332	.863	.911	.350
1501	.919	.004	.339	.956	.043	.125	.944	.007	.173	.100	.003	.361	.944	.002	.187	.884	.006	.458
1510	.974	.995	.101	.982	.999	.062	.974	.996	.091	.912	.988	.585	.939	.994	.119	.965	.991	.112
6332	.769	.824	1.127	.796	.871	.385	.759	.851	.434	.722	.789	.564	.759	.844	.498	.722	.814	.506
23381	.530	.610	1.726	.587	.600	.733	.587	.645	.667	.580	.655	.690	.570	.626	.663	.550	.607	.729
40966	.995	.000	.012	.995	.500	.011	.995	.000	.014	.127	.008	.749	.991	.000	.031	.958	.002	.266
40975	.986	.000	.039	.991	.090	.033	.991	.029	.032	.560	.026	.348	.988	.001	.036	.977	.004	.100
40982	.738	.044	.682	.798	.171	.491	.776	.093	.519	.316	.079	1.509	.764	.040	.540	.787	.036	.570
40994	.954	.976	.108	.977	.996	.071	.968	.992	.083	.917	.947	.629	.954	.992	.084	.972	.985	.100

Considerações: Limitações e Conclusão

Limitações do SAINT



Custo Computacional

Modelos Transformer, incluindo o SAINT, são computacionalmente mais intensivos que métodos clássicos como Gradient Boosting, especialmente com datasets grandes.



Interpretabilidade

Embora os mecanismos de atenção ofereçam alguma visão sobre a importância das features, a interpretabilidade global é menor que a de modelos baseados em árvores.



Sensibilidade a Hiperparâmetros

O desempenho pode ser sensível à escolha de hiperparâmetros, exigindo um processo de ajuste cuidadoso e computacionalmente custoso.



Necessidade de Dados

Embora o pré-treinamento ajude em cenários com poucos dados rotulados, a eficácia geral pode ainda depender de uma quantidade razoável de dados.

Conclusão

O SAINT representa um avanço significativo no uso de deep learning para dados tabulares, combinando:

Mecanismos de atenção dual (self-attention e intersample attention)

Embeddings aprimorados para features heterogêneas

Pré-treinamento contrastivo auto-supervisionado

Estas inovações permitem ao SAINT:

Capturar interações complexas entre features

Modelar relações entre diferentes amostras

Superar métodos anteriores de deep learning para dados tabulares

Competir e frequentemente superar métodos clássicos como Gradient Boosting

O SAINT demonstra que, com arquiteturas especializadas, o deep learning pode alcançar e superar métodos tradicionais em dados tabulares.