Documento Explicativo Treinamento

Este documento descreve a **implementação técnica detalhada** dos scripts de limpeza, tratamento, engenharia de features e modelagem preditiva utilizados no **projeto Daruma**, desenvolvido para o desafio de previsão de *targets* de jogadores.

O objetivo desta documentação é explicar **as decisões técnicas e arquiteturais** adotadas, bem como justificar por que elas tornam o pipeline mais **eficiente, robusto e escalável**.

---

1. Visão Geral do Pipeline

O pipeline segue uma estrutura modular e escalável, separando claramente as fases de preparação, modelagem e avaliação:

[Importação e limpeza] → [Engenharia de features] →

[Seleção de variáveis] → [Otimização de hiperparâmetros] →

[Treinamento com ensemble] → [Validação cruzada e métricas] → [Exportação dos modelos]

A execução é feita em Python, aproveitando bibliotecas otimizadas para manipulação vetorizada e aprendizado de máquina, como:

- **Pandas / NumPy** → Processamento vetorial de alta performance.
- **Scikit-learn** → Padronização, métricas e cross-validation.
- **Optuna** → Otimização eficiente de hiperparâmetros via *Bayesian Optimization*.
- **CatBoost** → Modelo de boosting de gradiente de alta performance e tolerante a ruído.
- **Matplotlib / Seaborn** → Visualização e diagnóstico.
- **Joblib / Pickle** → Serialização dos modelos e artefatos.

---

2. Limpeza e Tratamento de Dados

A primeira etapa foca na integridade e consistência dos dados.
O script realiza:

- **Conversão de tipos:** colunas numéricas convertidas explicitamente (to_numeric) com errors='coerce'.
- **Padronização de separadores decimais:** substituição de vírgula por ponto para compatibilidade com operações matemáticas.
- **Tratamento de valores ausentes:** imputação por mediana, reduzindo o impacto de *outliers*.

- **Correção de valores inválidos:** valores como -1 (indicativos de ausência) são tratados como NaN.

Justificativa técnica:
O uso de **imputação por mediana** é mais robusto em distribuições assimétricas, e a coerção automática evita falhas de execução.
Toda limpeza é vetorizada, dispensando loops explícitos e otimizando o desempenho.

---

3. Engenharia de Features

Após a normalização, o script constrói *features derivadas* para capturar padrões temporais e relacionais:

- **Estatísticas por grupo:** médias (mean), desvios padrão (std) e máximos (max).
- **Features temporais compostas:** médias de períodos iniciais e tardios (P_early, P_late).
- **Interações multiplicativas:** F1103_X_P_mean, combinando variáveis relevantes.
- **Engenharia condicional:** cálculos feitos apenas se as colunas existirem.

Justificativa técnica:
A criação de *features estatísticas e interativas* melhora o aprendizado de modelos de boosting, capturando relações não lineares e reduzindo *underfitting*.

---

4. Seleção de Features

O filtro de seleção baseia-se em **correlação com o target**:

```
corr = abs(df[col].corr(df[TARGET]))

if corr > 0.35:

   feature_pool.append(col)
```

Somente as 15 variáveis mais correlacionadas são mantidas.

Justificativa técnica:

- Reduz dimensionalidade e custo computacional.
- Evita colinearidade e *overfitting*.
- Melhora a interpretabilidade e a velocidade de inferência.

---

5. Modelagem e Otimização

A modelagem é baseada no **CatBoostRegressor**, um algoritmo de *gradient boosting* otimizado para dados tabulares.

Configurações Técnicas:

- **Otimização com Optuna:** até 100 *trials* para buscar hiperparâmetros ideais.
- **Validação cruzada k-fold (k=3):** métrica base da otimização.
- **Escalonamento com RobustScaler:** mais resistente a *outliers*.
- **Ensemble de 3 modelos** com *seeds* diferentes (42, 123, 456) para reduzir variância.

Justificativa técnica:

- CatBoost elimina *one-hot encoding*, economizando memória e tempo.
- Optuna usa otimização bayesiana, mais eficiente que busca em grade.
- Ensemble estabiliza previsões e melhora generalização.

---

6. Validação e Métricas

Métricas usadas:

- **R² (R-squared):** proporção da variância explicada.
- **MAE (Erro Absoluto Médio):** erro médio em unidades reais.
- **MAPE Seguro:** variação modificada do *Mean Absolute Percentage Error*:

```
def safe_mape(y_true, y_pred):
    return np.mean(np.abs((y_true - y_pred) / np.maximum(np.abs(y_true), 1e-10))) * 100
```

Outras técnicas:

- **Leave-One-Out Cross Validation (LOO-CV)** → avaliação individual por amostra.
- **Controle de overfitting:** diferença < 15% entre treino e teste.

Justificativa técnica:
O *safe MAPE* evita divisões por zero e resultados distorcidos, enquanto o LOO-CV garante avaliação confiável e independente.

---

7. Exportação e Integração

Após o treinamento:

- Modelos salvos como .cbm (formato nativo CatBoost).
- Scalers e listas de features exportados via pickle.
- Integrado ao backend **FastAPI**, que carrega e serve as previsões via REST.

Justificativa técnica:
O formato .cbm reduz tempo de carregamento e permite integração direta em ambientes Docker com latência mínima.

---

8. Eficiência e Escalabilidade

Técnica	Benefício
Vetorização com Pandas/Numpy	Processamento até 50× mais rápido que iteração.
CatBoost + Optuna	Busca inteligente de parâmetros, com alta performance.
Ensemble de modelos	Reduz variância e aumenta estabilidade.
Persistência de artefatos	Evita reprocessamento e acelera inicialização.

Conclusão: o pipeline é projetado para **treinar em minutos e inferir em milissegundos**, sendo ideal para ambientes de produção.

---

9. Reprodutibilidade e Manutenção

- **Seeds fixos** garantem reprodutibilidade.
- **Estrutura modular** facilita ajustes isolados.
- **Logging detalhado** e métricas registradas em tempo real.
- **Documentação integrada** para facilitar manutenção futura.

🏁 Conclusão

O pipeline desenvolvido combina **robustez estatística, desempenho computacional e integração fluida**, resultando em uma solução de Machine Learning pronta para produção.

Em resumo:

- **CatBoost + Optuna** → eficiência e precisão.
- **Ensemble + LOO-CV** → estabilidade e confiabilidade.
- **Safe MAPE + RobustScaler** → métricas consistentes.
- **Exportação modular** → integração ágil com APIs e dashboards.

Este pipeline representa um padrão de referência em engenharia de Machine Learning para dados tabulares — eficiente, escalável e reprodutível.