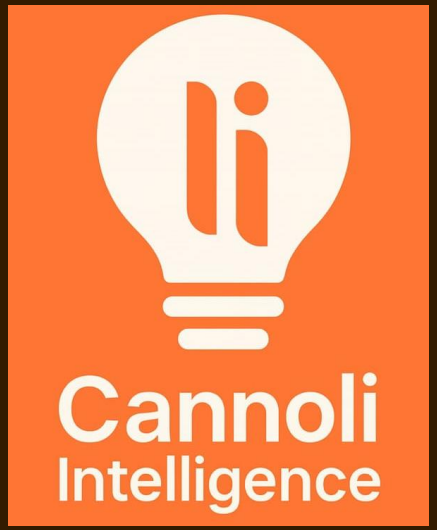
Cannoli Intelligence

Inteligência Artificial e Machine Learning.

Alexandra Christine
Carlos Augusto
Hebert Esteves
José Bento

Cannoli Intelligence



Plataforma de Business Intelligence desenvolvida para a empresa Cannoli, com o objetivo de transformar dados brutos de marketing, vendas e campanhas em informações estratégicas e acionáveis..

- Alertas inteligentes, que detectam quedas e anomalias de desempenho.
- Modelos preditivos, capazes de estimar resultados futuros de campanhas.
- Recomendações automáticas, baseadas em algoritmos como Busca Gulosa e Random Forest, que priorizam campanhas mais eficientes.

Protótipo de IA

Alertas inteligentes

Sistema de detecção automática de quedas de desempenho em campanhas, utilizando média móvel de 7 dias e análise de desvios estatísticos (z-score).

Quando o número de cliques ou conversões cai mais de 30% em relação à média, o modelo gera um alerta automático, permitindo agir antes que o desempenho da campanha se deteriore.

Previsão de conversões

Regressão Linear para prever o número de conversões com base em métricas históricas como impressões, cliques, custo e receita.

Recomendação

Aplicamos um algoritmo de Busca Gulosa (Greedy Search) para ranquear campanhas pela eficiência (conversões ÷ custo).

O sistema recomenda as 3 campanhas mais rentáveis para priorizar investimento e indica as que devem ser ajustadas ou pausadas, exibindo os resultados no card “Sugestões IA” do dashboard.

Bibliotecas Utilizadas

Entrega 1: Validar a viabilidade do uso de IA no projeto.

Modelos implementados:

- IA Reativa (alertas via média móvel e z-score);
- Regressão Linear (previsão de desempenho);
- Busca Gulosa (recomendações rápidas).

Entrega 1: Validar a viabilidade do uso de IA no projeto.

campanhas_simulado.csv

date	campanhaId	nome	impressoes	cliques	conversoes	custo	receita
2025-06-24	1	Campanha_1	24326	3057	391	3544.12	9448.1
2025-06-25	1	Campanha_1	25756	2303	365	3133.44	7081.53
2025-06-26	1	Campanha_1	27619	2986	375	2738.7	8952.55
2025-06-27	1	Campanha_1	26997	3810	562	4787.09	13737.34
2025-06-28	1	Campanha_1	20587	2574	374	2941.54	8195.49
2025-06-29	1	Campanha_1	18847	2077	290	2500.09	8400.6
2025-06-30	1	Campanha_1	17665	2234	436	3165.95	12257.72
2025-07-01	1	Campanha_1	25473	2686	290	3637.32	7055.74
2025-07-02	1	Campanha_1	24952	2663	312	3350.82	9338.61
2025-07-03	1	Campanha_1	30124	3750	580	4526.43	13864.0

alertas_queda.csv

date	campanhaId	nome	motivo
2025-07-04	4	Campanha_4	Cliques 299 abaixo de 329 (média 7d)
2025-07-04	6	Campanha_6	Cliques 437 abaixo de 522 (média 7d)
2025-07-06	3	Campanha_3	Cliques 1011 abaixo de 1393 (média 7d)
2025-07-06	4	Campanha_4	Cliques 309 abaixo de 327 (média 7d)
2025-07-07	6	Campanha_6	Cliques 458 abaixo de 476 (média 7d)
2025-07-14	3	Campanha_3	Cliques 1218 abaixo de 1246 (média 7d)
2025-07-14	5	Campanha_5	Cliques 889 abaixo de 1079 (média 7d)
2025-07-19	6	Campanha_6	Cliques 575 abaixo de 640 (média 7d)
2025-07-26	6	Campanha_6	Cliques 235 abaixo de 649 (média 7d)
2025-07-27	5	Campanha_5	Cliques 987 abaixo de 1163 (média 7d)

Entrega 1: Validar a viabilidade do uso de IA no projeto.

```
Cannoli Intelligence – IA/ML (dados simulados)
1) Alertas (queda média móvel 7d, 30%):
- Total de alertas gerados: 46
- Exemplo primeira linha:
      date  campanhaId       nome                                motivo
2025-07-04           4 Campanha_4 Cliques 299 abaixo de 329 (média 7d)

2) Regressão Linear – previsão de conversões (features de defasagem 1 dia):
- MAE  : 65.793
- RMSE : 88.305
- R²   : 0.668

3) Busca Gulosa – recomendações (heurística: eficiência = conversões/custo, orçamento R$ 1200):
{
  "data_referencia": "2025-09-21",
  "heuristica": "eficiencia",
  "orcamento_total": 1200.0,
  "priorizar": [
    {
      "campanhaId": 3,
      "nome": "Campanha_3",
      "score": 0.3104852699299716,
      "orcamentoSugerido": 547.53
    },
    {
      "campanhaId": 4,
      "nome": "Campanha_4",
      "score": 0.1474675279481168,
      "orcamentoSugerido": 100.0
    },
```

Entrega 1: Validar a viabilidade do uso de IA no projeto.


```
{
  "data_referencia": "2025-09-21",
  "heuristica": "eficiencia",
  "orcamento_total": 1200.0,
  "priorizar": [
    {
      "campanhaId": 3,
      "nome": "Campanha_3",
      "score": 0.3104852699299716,
      "orcamentoSugerido": 547.53
    },
    {
      "campanhaId": 4,
      "nome": "Campanha_4",
      "score": 0.1474675279481168,
      "orcamentoSugerido": 100.0
    },
    {
      "campanhaId": 6,
      "nome": "Campanha_6",
      "score": 0.08173406976253288,
      "orcamentoSugerido": 152.94
    }
  ],
  "ajustar_ou_pausar": [
    {
      "campanhaId": 1,
      "nome": "Campanha_1",
      "motivo": "baixo score / alto custo"
    },
```

Entrega 2: funcionamento do código

- O código lê e trata o arquivo Campaign_clean.csv, convertendo colunas categóricas em valores numéricos para análise.
- Utiliza o modelo RandomForestClassifier para prever o status das campanhas (Ativa, Concluída, Agendada ou Rascunho).
- Gera o arquivo sugestoes.json, com campanhas priorizadas ou ajustadas segundo o nível de confiança da IA, integrando os resultados ao dashboard Cannoli Intelligence.

Entrega 2: funcionamento do código

```python
import argparse, json, sys
from pathlib import Path

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, classification_report

def load_data(csv_path: Path) -> pd.DataFrame:
    if not csv_path.exists():
        raise FileNotFoundError(f"CSV não encontrado: {csv_path}")
    df = pd.read_csv(csv_path, sep=";")
    return df
```

Entrega 2: funcionamento do código

```python
def prepare_dataframe(df: pd.DataFrame):
    # Inclui `name` para exibição, mas NÃO usa como feature do modelo
    cols = ["storeId", "name", "badge", "type", "isDefault", "createdBy", "updatedBy", "_mes", "status_desc"]
    missing = [c for c in cols if c not in df.columns]
    if missing:
        raise ValueError(f"Faltam colunas no CSV: {missing}")
    data = df[cols].dropna(subset=["status_desc"]).copy()
```

```python
encoders = {}
for col in ["storeId", "badge", "createdBy", "updatedBy", "_mes", "status_desc"]:
    enc = LabelEncoder()
    data[col] = enc.fit_transform(data[col])
    encoders[col] = enc
# Conjunto final para o modelo (removendo name e target)
X = data.drop(columns=["status_desc", "name"])
y = data["status_desc"]
return X, y, names, encoders
```

Entrega 2: funcionamento do código

```python
def prepare_dataframe(df: pd.DataFrame):
    # Inclui `name` para exibição, mas NÃO usa como feature do modelo
    cols = ["storeId", "name", "badge", "type", "isDefault", "createdBy", "updatedBy", "_mes", "status_desc"]
    missing = [c for c in cols if c not in df.columns]
    if missing:
        raise ValueError(f"Faltam colunas no CSV: {missing}")
    data = df[cols].dropna(subset=["status_desc"]).copy()
```

```python
encoders = {}
for col in ["storeId", "badge", "createdBy", "updatedBy", "_mes", "status_desc"]:
    enc = LabelEncoder()
    data[col] = enc.fit_transform(data[col])
    encoders[col] = enc
# Conjunto final para o modelo (removendo name e target)
X = data.drop(columns=["status_desc", "name"])
y = data["status_desc"]
return X, y, names, encoders
```

Entrega 2: funcionamento do código

```python
def train_model(X, y, seed=42):
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=seed, stratify=y
    )
    rf = RandomForestClassifier(n_estimators=150, max_depth=10, random_state=seed, n_jobs=-1)
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    f1w = f1_score(y_test, y_pred, average="weighted")
    report = classification_report(y_test, y_pred, output_dict=True)
    return rf, X_test, y_test, y_pred, acc, f1w, report
```

Entrega 2: funcionamento do código

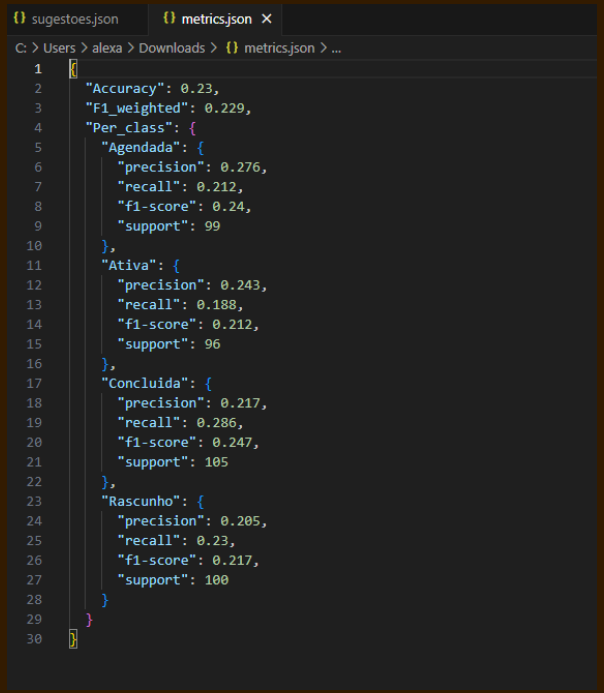
```python
def build_sugestoes(rf, X_test: pd.DataFrame, y_test, encoders: dict, names_test: pd.Series, topk=5):
    proba = rf.predict_proba(X_test)
    conf = proba.max(axis=1)
    y_pred = rf.predict(X_test)
    inv_status = encoders["status_desc"].inverse_transform
    status_prev = inv_status(y_pred)
    status_real = inv_status(y_test)
    def inv(col, arr): return encoders[col].inverse_transform(arr)
    df_pred = pd.DataFrame({
        "storeId": inv("storeId", X_test["storeId"].values),
        "badge": inv("badge", X_test["badge"].values),
        "createdBy": inv("createdBy", X_test["createdBy"].values),
        "updatedBy": inv("updatedBy", X_test["updatedBy"].values),
        "_mes": inv("_mes", X_test["_mes"].values),
        "name": names_test.values,
        "status_real": status_real,
        "status_previsto": status_prev,
        "confianca": conf
    })
    priorizar = df_pred.sort_values("confianca", ascending=False).head(topk).copy()
```

Entrega 2: funcionamento do código

```
metrics = {
    "Accuracy": round(acc, 3),
    "F1_weighted": round(f1w, 3),
    "Per_class": per_class
}
sugestoes = build_sugestoes(rf, X_test, y_test, encoders, names_test, topk=args.topk)
sugestoes["acuracia"] = round(acc, 3)
sugestoes["f1"] = round(f1w, 3)
(out_dir / "metrics.json").write_text(json.dumps(metrics, indent=2, ensure_ascii=False), encoding="utf-8")
(out_dir / "sugestoes.json").write_text(json.dumps(sugestoes, indent=2, ensure_ascii=False), encoding="utf-8")
print(f"[OK] Salvo em: {out_dir.resolve()}")
print(f"- {str((out_dir/'metrics.json').resolve())}")
print(f"- {str((out_dir/'sugestoes.json').resolve())}")
```

Entrega 2: Saídas em JSON

- A acurácia mostra quantas vezes o modelo acertou o status da campanha de forma geral.
- O F1 ponderado avalia se o modelo não está acertando só em uma classe, mas se está conseguindo equilibrar os acertos em todas as categorias.
- A parte de Per_class mostra o desempenho do modelo para cada tipo de status de campanha, permitindo avaliar em qual classe ele está indo melhor ou pior.



Entrega 2: Saídas em JSON

sugestoes.json

C: > Users > alexa > Downloads > {} sugestoes.json > ...


```json
{
  "modelo": "RandomForestClassifier",
  "descricao": "Classificação de status de campanhas a partir de atributos categóricos",
  "acuracia": 0.23,
  "f1": 0.229,
  "total_amostras": 400,
  "priorizar": [
    {
      "storeId": "1B7PXJBC8V",
      "badge": "winback",
      "responsavel": "camposaugusto",
      "name": "Campanha Aliquid DRQN",
      "status_previsto": "Rascunho",
      "confianca": 0.479
    },
    {
      "storeId": "42YJMLEQMJ",
      "badge": "winback",
      "responsavel": "gnascimento",
      "name": "Campanha Voluptatum 1SSQ",
      "status_previsto": "Rascunho",
      "confianca": 0.464
    },
    {
      "storeId": "SA12MESQ79",
      "badge": "Nao informado",
      "responsavel": "vicente84",
      "name": "Campanha Aut 0VON",
      "status_previsto": "Concluida",
      "confianca": 0.455
    },
    {
      "storeId": "1TWD96HHWM",
      "badge": "winback",
      "responsavel": "lara85",
      "name": "Campanha Est F1GZ",
      "status_previsto": "Rascunho",
      "confianca": 0.455
    },
    {
      "storeId": "VBM0YX6X3B",
      "badge": "consumption",
      "responsavel": "sarah49",
      "name": "Campanha Nemo 9KYH",
      "status_previsto": "Ativa",
      "confianca": 0.449
```

Pipeline técnico

Dados

São extraídos da base da Cannoli e servem como matéria-prima do sistema.

Limpeza e padronização

Essa etapa garante que o modelo de IA receba dados consistentes.

Modelos de IA/ML

- IA Reativa (alertas inteligentes);
- ML Supervisionado (previsão com Regressão Linear ou Random Forest);
- Busca Gulosa (recomendações rápidas).

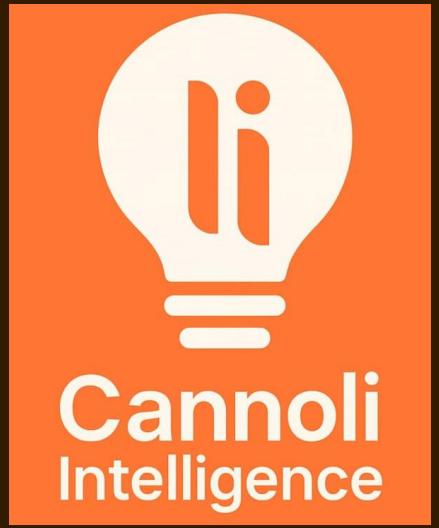
Saídas JSON

Todos os resultados (alertas, previsões, recomendações) são convertidos em arquivos JSON — formato leve e integrável ao front-end.

Dashboard

- “Sugestões IA” → recomendações automáticas.
- “Alertas Inteligentes” → quedas de desempenho.
- O painel passa a se comportar como um assistente de decisão.

Conclusão



- Evolução de um protótipo conceitual para uma aplicação prática e integrada ao dashboard.
- Entrega 1: foi validada a viabilidade do uso de IA através de dados simulados, com modelos que detectaram anomalias, realizaram previsões de desempenho e geraram recomendações automáticas por meio da Busca Gulosa.
- Entrega 2: Aplicação de Machine Learning supervisionado, utilizando o modelo Random Forest sobre a base real Campaign_clean.csv, com o objetivo de prever o status das campanhas e gerar automaticamente arquivos JSON integrados ao dashboard Cannoli Intelligence.

Cannoli Intelligence

Inteligência Artificial e Machine Learning.