Churn_model.py: Criar um **modelo de classificação** que estime a probabilidade de um cliente **dar churn** (parar de comprar / ficar inativo) num horizonte definido

```python
# IA_e_ML/churn_model.py
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.metrics import roc_auc_score, classification_report, confusion_matrix
import joblib
import matplotlib.pyplot as plt

# carregue features (pode usar segmentos e rfm)
df = pd.read_csv("IA_e_ML/segmentos_clientes.csv")  # já contém RFM e cluster
# Suponha que tenhamos target 'churn_90d' (1 = churn nos próximos 90 dias)
# Se não tem, você precisa construir baseado em last_active_date
if 'churn_90d' not in df.columns:
    # exemplo simples: churn se última compra > 90 dias
    df['churn_90d'] = (df['recency'] > 90).astype(int)

# features e target
target = 'churn_90d'
cat_cols = ['gender','region','cluster'] if 'gender' in df.columns else ['cluster']
num_cols = ['recency','frequency','monetary','age'] if 'age' in df.columns else
['recency','frequency','monetary']

X = df[num_cols + cat_cols]
y = df[target]
```

```
# split

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.2,
random_state=42)


# preprocessing

num_pipe = Pipeline([

    ('imputer', SimpleImputer(strategy='median')),

    ('scaler', StandardScaler())

])

cat_pipe = Pipeline([

    ('imputer', SimpleImputer(strategy='most_frequent')),

    ('ohe', OneHotEncoder(handle_unknown='ignore', sparse=False))

])

preproc = ColumnTransformer([

    ('num', num_pipe, num_cols),

    ('cat', cat_pipe, cat_cols)

])


pipe = Pipeline([

    ('preproc', preproc),

    ('clf', RandomForestClassifier(n_estimators=200, random_state=42, n_jobs=-1))

])


# hyperparam tuning (pequeno)

param_grid = {

    'clf__max_depth': [5, 10, None],

    'clf__min_samples_split': [2, 5]

}

cv = StratifiedKFold(n_splits=4, shuffle=True, random_state=42)

gs = GridSearchCV(pipe, param_grid, scoring='roc_auc', cv=cv, n_jobs=-1)

gs.fit(X_train, y_train)
```

```
# avaliar
best = gs.best_estimator_
y_pred_proba = best.predict_proba(X_test)[:,1]
y_pred = best.predict(X_test)

auc = roc_auc_score(y_test, y_pred_proba)
report = classification_report(y_test, y_pred, output_dict=False)
cm = confusion_matrix(y_test, y_pred)

# salvar
joblib.dump(best, "IA_e_ML/churn_model.joblib")

with open("IA_e_ML/churn_metrics.txt", "w") as f:
   f.write(f"AUC: {auc:.4f}\n")
   f.write(report)
   f.write("\nConfusion matrix:\n")
   f.write(str(cm))

# plot ROC (simples)
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
plt.figure()
plt.plot(fpr, tpr)
plt.plot([0,1],[0,1],'--')
plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title(f"ROC curve (AUC={auc:.3f})")
plt.savefig("IA_e_ML/roc_curve.png", dpi=150)
```