Mineração de Dados Aplicada

Nomao Dataset

Ian Trinta
Carlos Clark
Igor Rocha
Douglas Santos

Nomao

- Nomao era conhecido como um motor de busca de lugares, agregando informações sobre estabelecimentos e pontos de interesse.
- Sua base de dados era formada a partir de múltiplas fontes (web, GPS, usuários etc.), resultando em um volume expressivo de registros.

Nomao Challenge

- Competição de Data Mining realizada em 2012, com foco na deduplicação de registros de locais.
- O desafio central era criar modelos capazes de identificar se duas entradas correspondiam ao mesmo lugar, ou seja, consistia nos desafios da deduplicação.

Motivação

Qualidade dos Dados:

A existência de registros duplicados prejudica buscas, relatórios e análises, afetando a confiabilidade do sistema.

Decisões Estratégicas:

Empresas que dependem de dados geoespaciais (logística, mapas digitais, delivery etc.) precisam de dados limpos e precisos para tomar decisões acertadas.

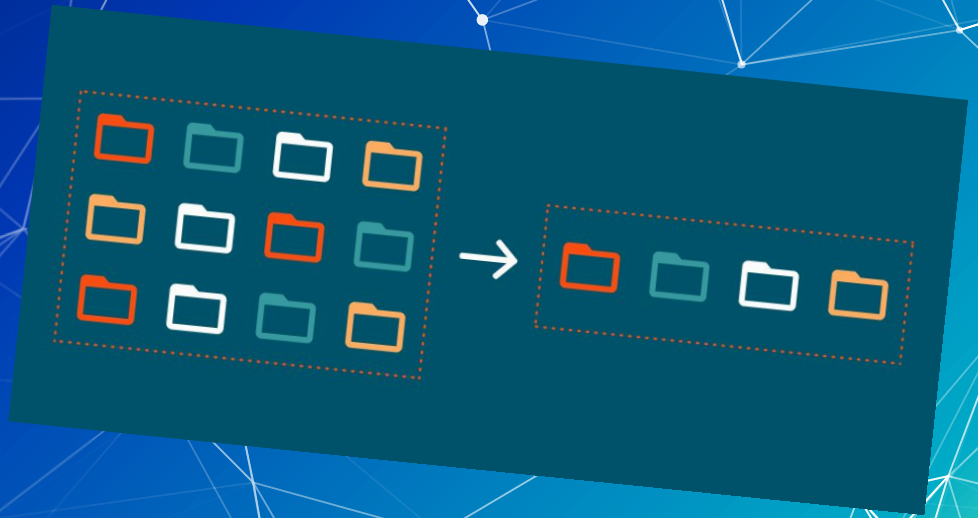
Metas

Deduplicação Eficiente:

Desenvolver um modelo capaz de identificar se dois registros se referem ao mesmo local, reduzindo drasticamente o número de duplicatas.

Melhoria de Indicadores:

Alcançar métricas robustas (por exemplo, F1-score elevado) que atestem a eficácia do modelo.



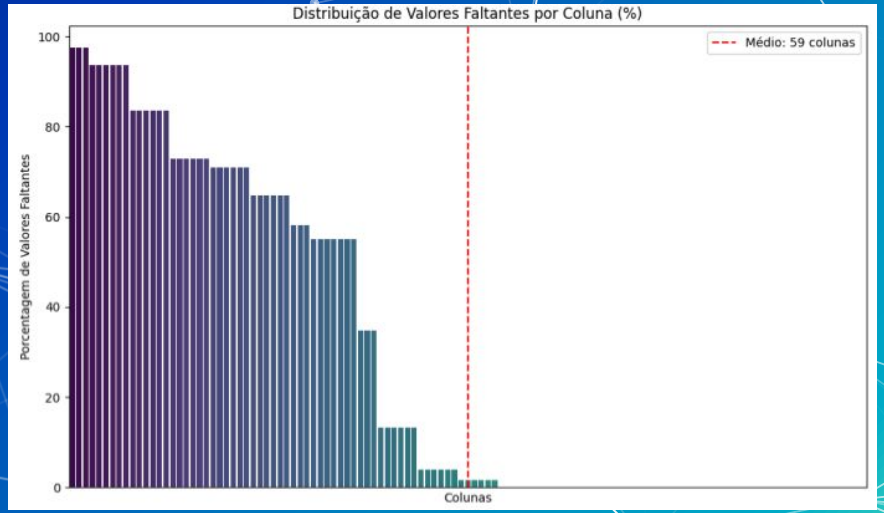
Entendimento dos Dados

Tamanho do Dataset

- o conjunto de dados Nomao contém 34.466 registros e 120 variáveis.
- 31 variáveis são categóricas, enquanto 89 são numéricas.

Característica do Dataset

- A diversidade de fontes de dados utilizadas na construção do dataset acarretou muitos valores faltantes.
- Muitos atributos dependem de dados provenientes de diferentes origens, assim resultando em lacunas.



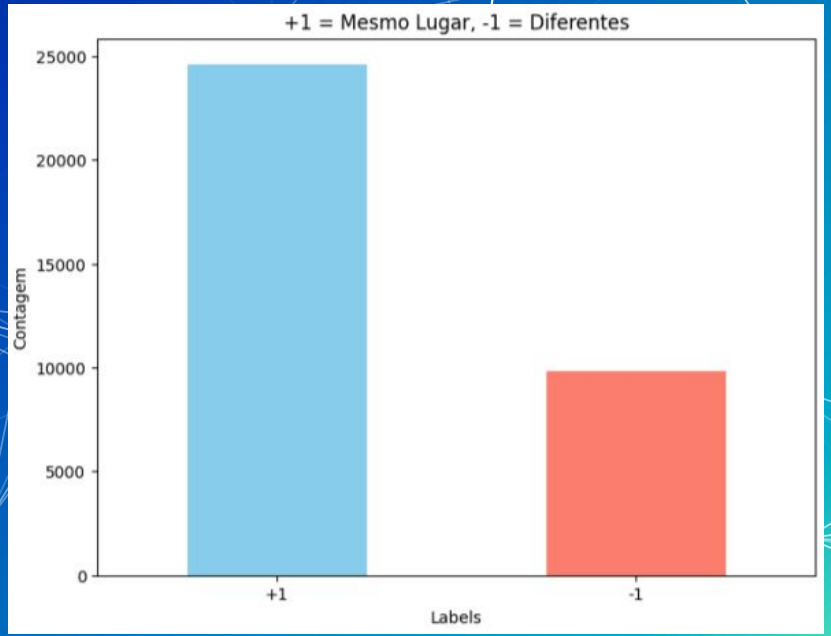
Característica do Dataset

- Dados faltantes são demarcados por um “?”.
- As variáveis contínuas estão no intervalo de 0 a 1.
- As varíaveis categóricas podem ter os valores n,s,m.

```
*************************
6. Number of Attributes

120 attributes: 89 continuous, 31 nominal (including the attributes 'label' and 'id').

The features are separated by comma.

*************************
7. Attribute Information:

Missing data are allowed, represented by question marks '?'.

Labels are +1 if the concerned spots must be merged, -1 if they do not refer to the same entity.

1 id: name is composed of the names of the spots that are compared, separated by a sharp (#).
2 clean_name_intersect_min: continuous.
3 clean_name_intersect_max: continuous.
4 clean_name_levenshtein_sim: continuous.
5 clean_name_trigram_sim: continuous.
6 clean_name_levenshtein_term: continuous.
7 clean_name_trigram_term: continuous.
8 clean_name_including: n,s,m.
9 clean_name_equality: n,s,m.
10  city_intersect_min: continuous.
```

Label do nosso Dataset

- No dataset, a variável "label" indica se dois pontos correspondem ao mesmo local ou não.
- Foi fundamental avaliar o balanceamento dessa variável.



Pré-processamento

Etapas

Divisão do conjunto de Dados

Separação dos dados em conjuntos de treino (80%) e teste (20%), garantindo a consistência das classes com a técnica de estratificação.

Remoção de Colunas Irrelevantes

Eliminação da coluna 'id' por não contribuir para o modelo preditivo.

Tratamento de Valores Faltantes

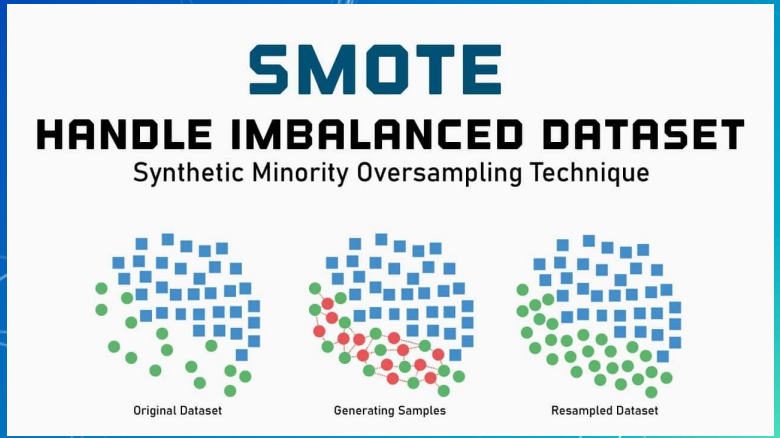
Preenchimento de valores nulos em colunas numéricas com -1, estratégia utilizada devido a referencial teórico que estudamos de artigos sobre a Nomao Challenge.

Codificação de Variáveis Categóricas

Identificação de colunas categóricas e aplicação de One-Hot Encoding para conversão em variáveis numéricas.

Balanceamento das Classes

Utilização da técnica SMOTE para balancear as classes no conjunto de treino



Modelagem e Avaliação

Modelos de Classificação escolhidos

- KNN
- LVQ
- Decision Tree
- SVM
- Random Forest
- MLP
- Comitê de Redes Neurais
- Comitê Heterogêneo (Stacking)
- XG-Boost
- LightGBM

Configuração para busca dos hiperparâmetros.

Utilizamos o RandomizedSearchCV para otimizar hiperparâmetros sem testar todas as combinações.

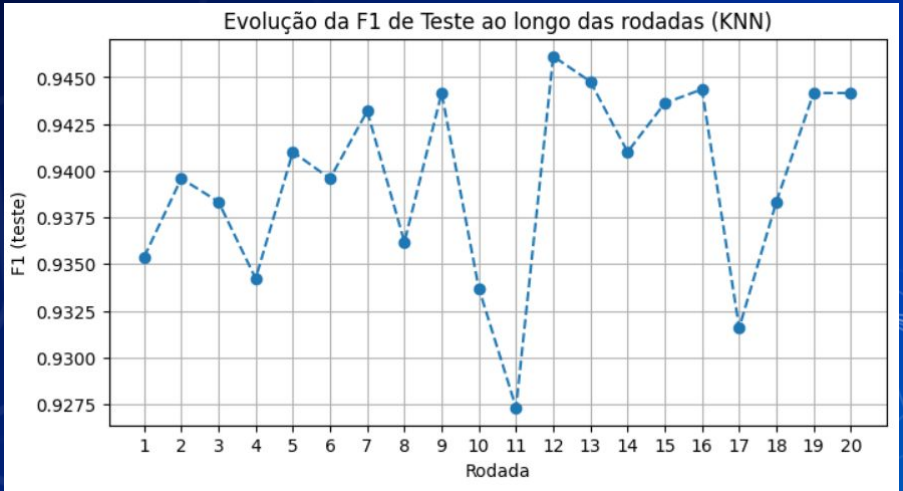
Para cada configuração:

1. Validação Cruzada Estratificada: Dividimos os dados em k folds, mantendo a proporção das classes.
2. Treinamento & Avaliação: O modelo é treinado nos k-1 blocos e testado no restante. A métrica (ex: f1_macro) é calculada em cada fold.
3. Média & Desvio-Padrão: A pontuação final é a média das métricas dos k folds.

KNN

```
# Espaço de busca dos hiperparâmetros do KNN
param_dist_knn = {
    'knn__n_neighbors': list(range(1, 31)),  # 1 a 30
    'knn__weights': ['uniform', 'distance'],
    'knn__p': [1, 2]  # 1=Manhattan, 2=Euclidiana
}
```

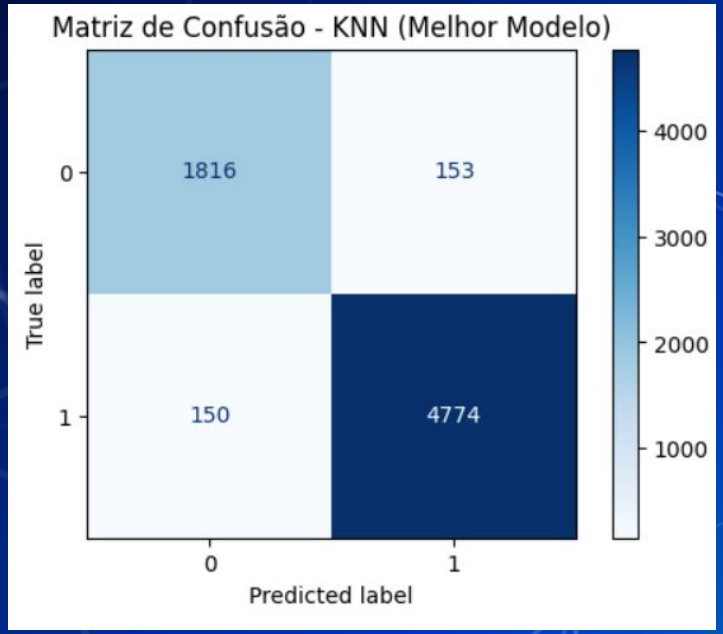
KNN

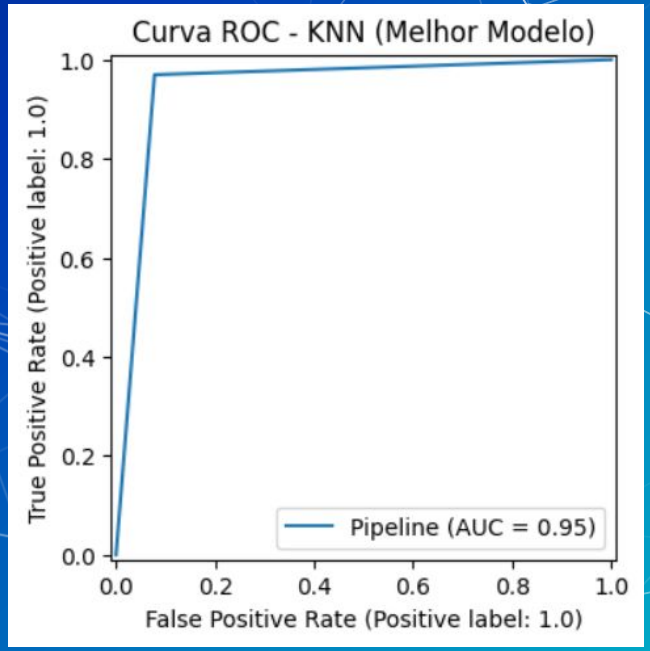



```
Desempenho no Conjunto de Teste:
Acurácia  : 0.9560
Precisão  : 0.9463
Recall    : 0.9459
F1-score  : 0.9461
AUC       : 0.9459
```

```
Melhor rodada: 12
F1 nessa rodada: 0.9461202153510431
Melhores hiperparâmetros: {'knn__weights': 'uniform', 'knn__p': 1, 'knn__n_neighbors': 1}
```

KNN

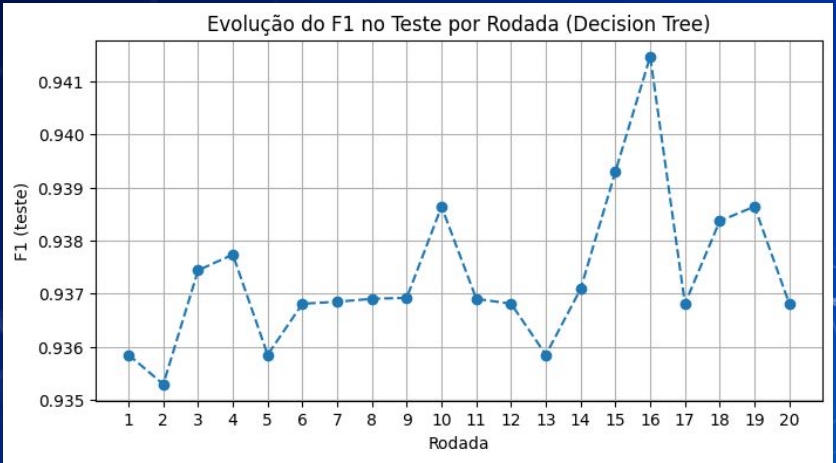

Matriz de Confusão - KNN (Melhor Modelo)
1816
153
150
4774
0
1
True label
Predicted label
4000
3000
2000
1000


Curva ROC - KNN (Melhor Modelo)
True Positive Rate (Positive label: 1.0)
False Positive Rate (Positive label: 1.0)
Pipeline (AUC = 0.95)
0.0
0.2
0.4
0.6
0.8
1.0

Decision Tree

```
# 1) Definindo o espaço de hiperparâmetros para a Decision
Tree

param_dist_dt = {

    'criterion': ['gini', 'entropy', 'log_loss'],

    'max_depth': range(1, 100),

    'min_samples_split': range(2, 11),

    'min_samples_leaf': range(1, 11)

}
```

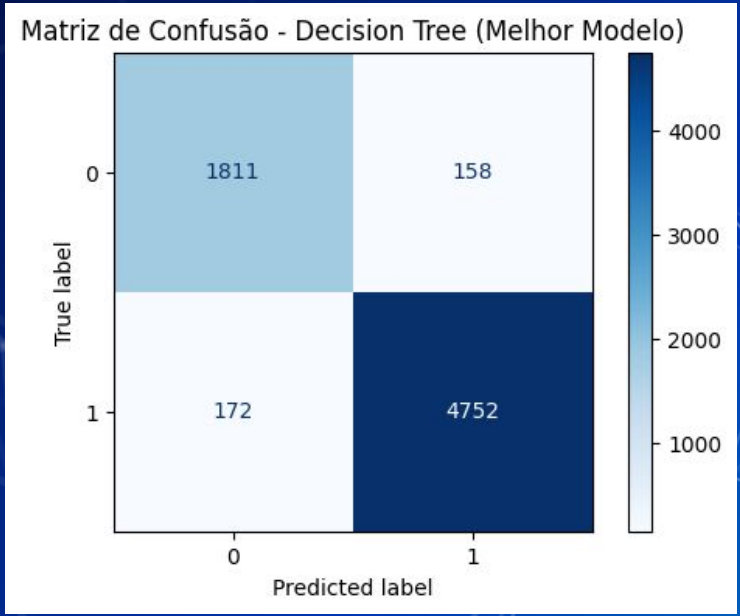
Decision Tree

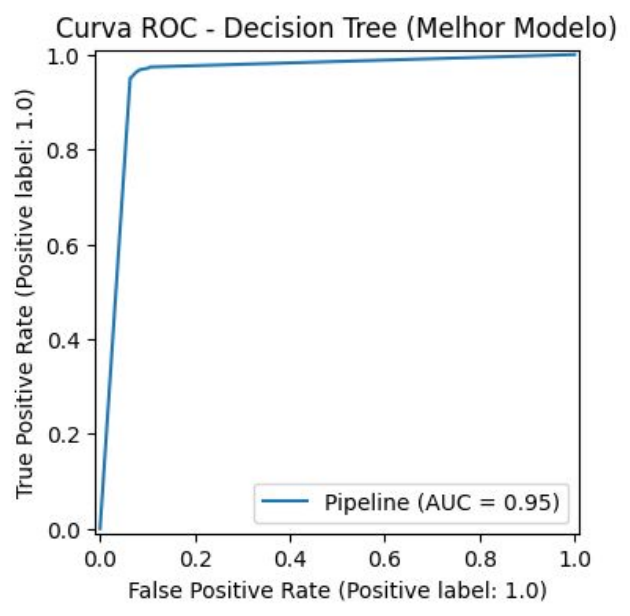



```
Desempenho no Conjunto de Teste:
Acurácia : 0.9521
Precisão : 0.9405
Recall   : 0.9424
F1-score : 0.9415
AUC      : 0.9535
```

```
Melhor rodada: 16
F1 nessa rodada: 0.9414704643643181
Melhores hiperparâmetros: {'dt__min_samples_split': 4, 'dt__min_samples_leaf': 3, 'dt__max_depth': 35, 'dt__criterion': 'entropy'}
```

Decision Tree

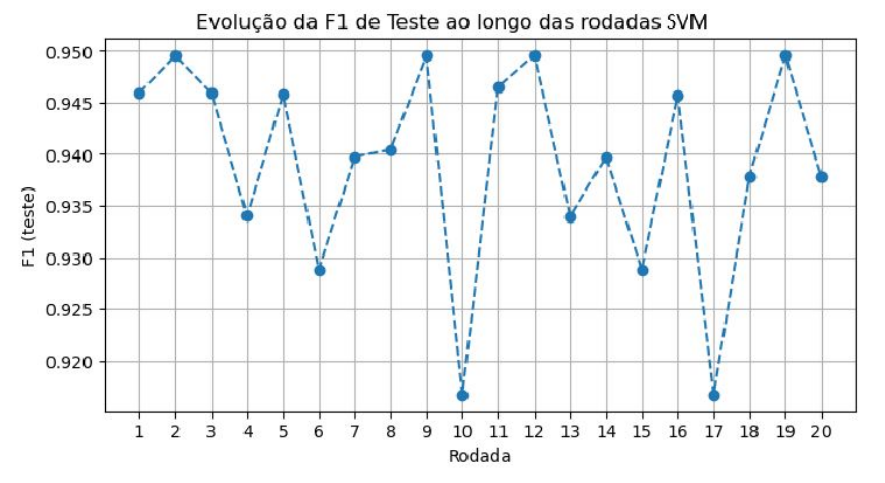




SVM

```python
# Espaço de busca dos hiperparâmetros do SVM
param_dist_svm = {
    'svm__C': [0.1, 1, 10, 100],
    'svm__kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
    'svm__degree': [2, 3, 4],
    'svm__gamma': ['scale', 'auto'],
    'svm__class_weight': ['balanced', None]
}
```

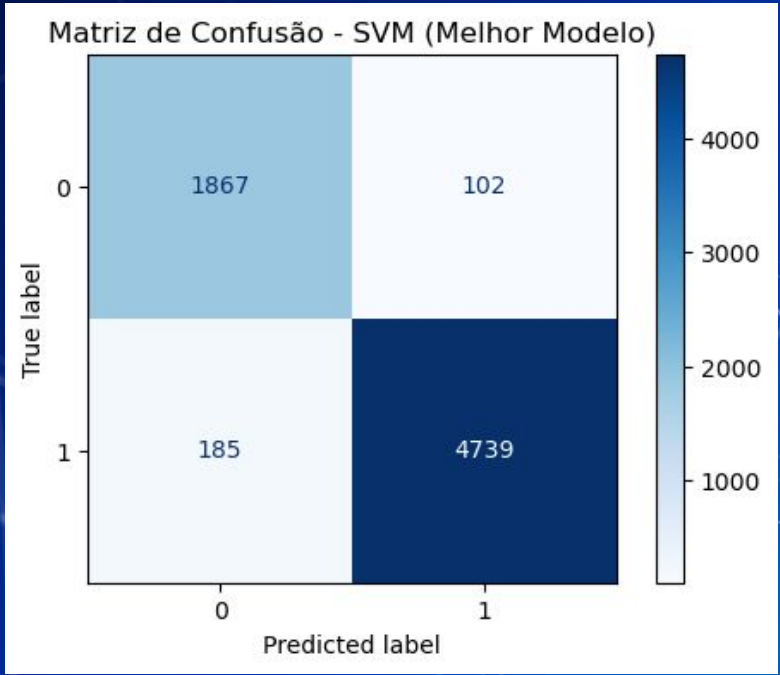
SVM

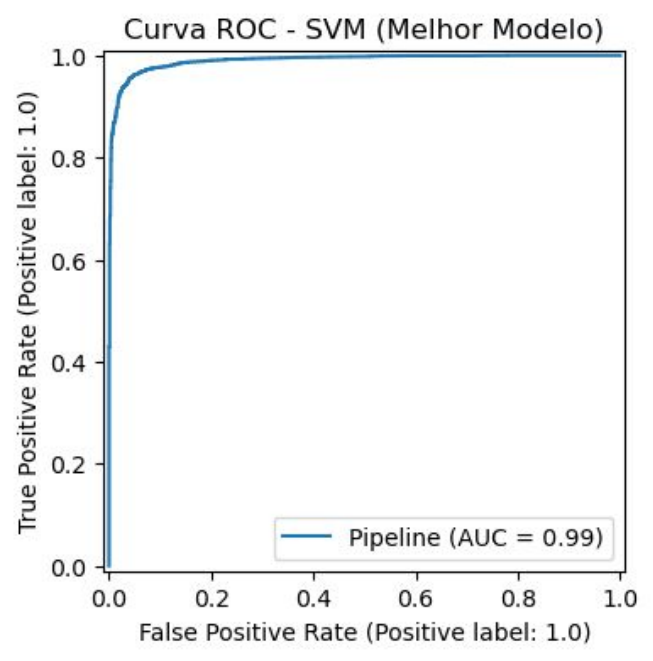



```
Desempenho no Conjunto de Teste:
Acurácia  : 0.9584
Precisão  : 0.9444
Recall    : 0.9553
F1-score  : 0.9496
AUC       : 0.9903
```

```
Melhores hiperparâmetros (validação cruzada): {'svm__kernel': 'rbf', 'svm__gamma': 'auto', 'svm__degree': 3, 'svm__class_weight': 'balanced', 'svm__C': 10}
Melhor F1 (validação cruzada - média): 0.9424643017928819
F1 no Teste: 0.9378704434016374
```

SVM

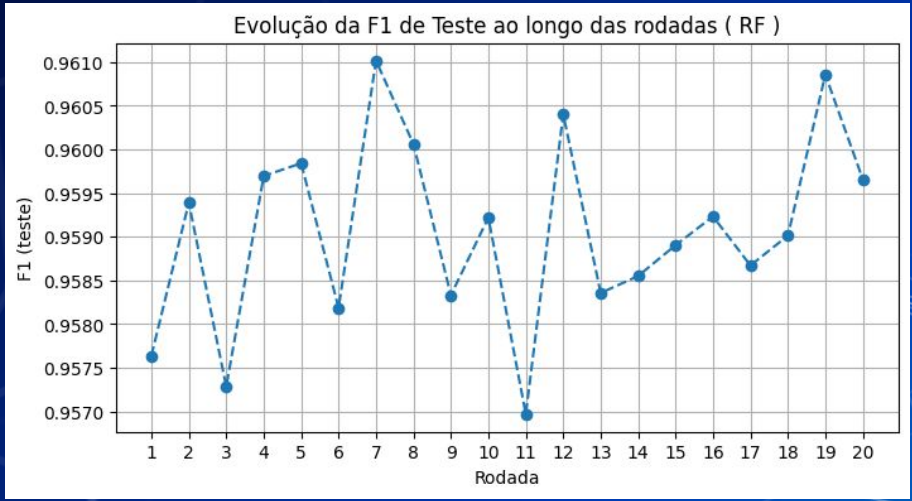

Matriz de Confusão - SVM (Melhor Modelo)
1867
102
185
4739
0
1
True label
0
1
Predicted label
4000
3000
2000
1000


Curva ROC - SVM (Melhor Modelo)
True Positive Rate (Positive label: 1.0)
False Positive Rate (Positive label: 1.0)
Pipeline (AUC = 0.99)
0.0
0.2
0.4
0.6
0.8
1.0

Random Forest

```
param_dist_rf = {
    'rf__n_estimators': range(50, 301, 50),            # Valores: 50, 100, 150, 200, 250, 300
    'rf__max_depth': [None] + list(range(5, 26, 5)),     # None, 5, 10, 15, 20, 25
    'rf__min_samples_split': range(2, 11),              # 2 até 10
    'rf__min_samples_leaf': range(1, 5),                # 1 até 4
    'rf__bootstrap': [True, False]                      # Uso de bootstrap: True ou False
}
```

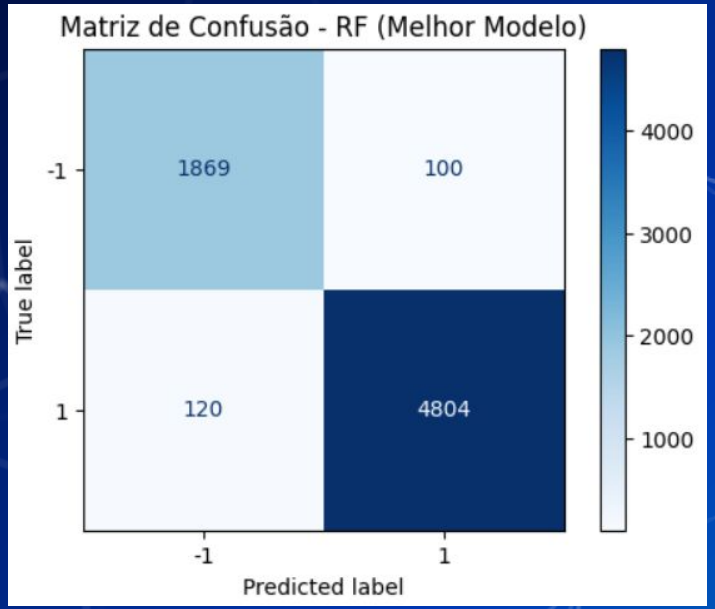
Random Forest

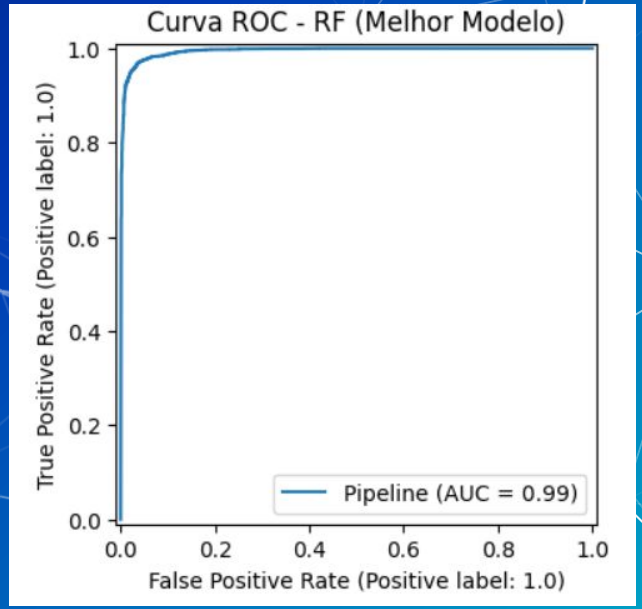



```
Desempenho no Conjunto de Teste:
Acurácia   : 0.9681
Precisão   : 0.9596
Recall     : 0.9624
F1-score   : 0.9610
AUC        : 0.9943
```

```
Melhor rodada: 7
F1 nessa rodada: 0.9610156747599901
Melhores hiperparâmetros: {'rf__n_estimators': 100, 'rf__min_samples_split': 5, 'rf__min_samples_leaf': 1, 'rf__max_depth':
25, 'rf__bootstrap': False}
```

Random Forest

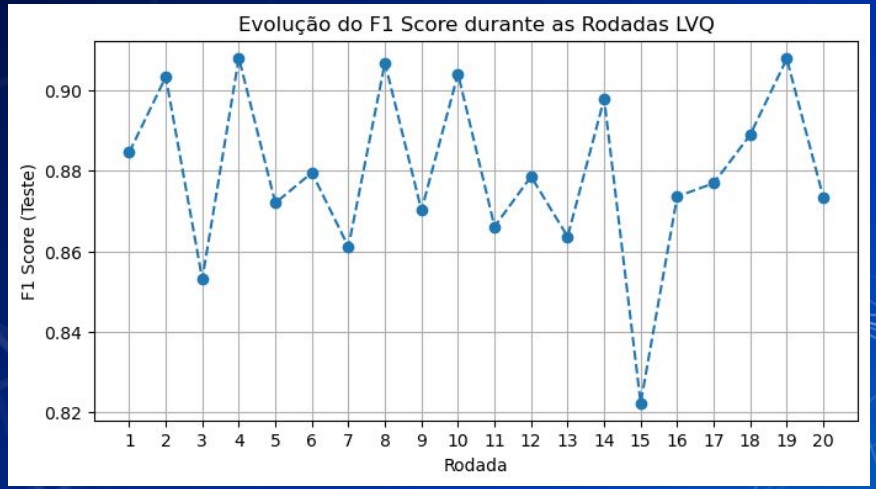




LVQ

```
param_dist = {
    'n_codebooks': [5, 10, 15, 20, 25],
    'lrate': (0.001, 0.1),  # Intervalo contínuo
    'epochs': [1,2,3]
}
```

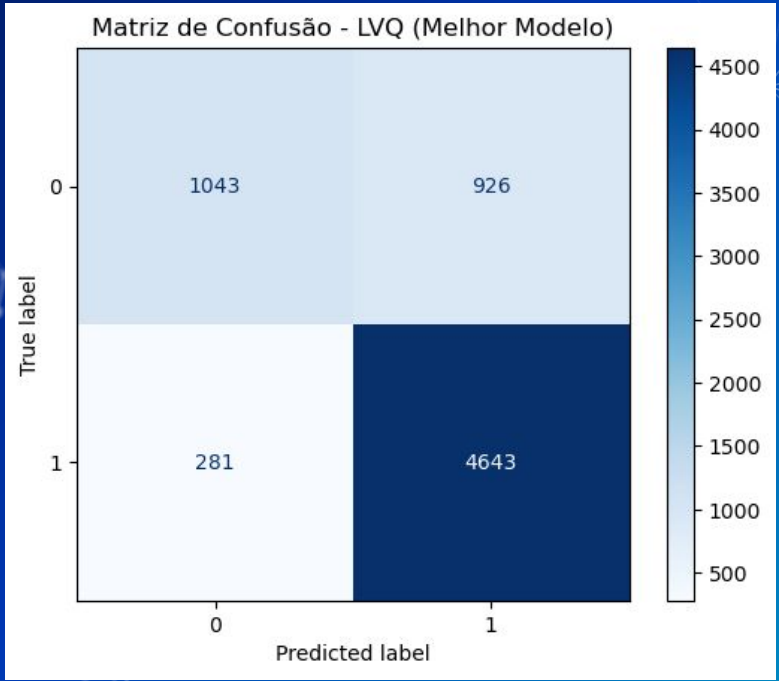
LVQ




```
Desempenho no Conjunto de Teste - LVQ:
Acurácia  : 0.8249
Precisão  : 0.8107
Recall    : 0.7363
F1-score  : 0.7592
```

```
Melhor rodada: 4
F1 essa rodada: 0.90802265460604 54
Melhores hiperparâmetros encontrados: {'n_codebooks=20, 'lrate' = 0.05342005303613508, 'epochs' = 2}
```

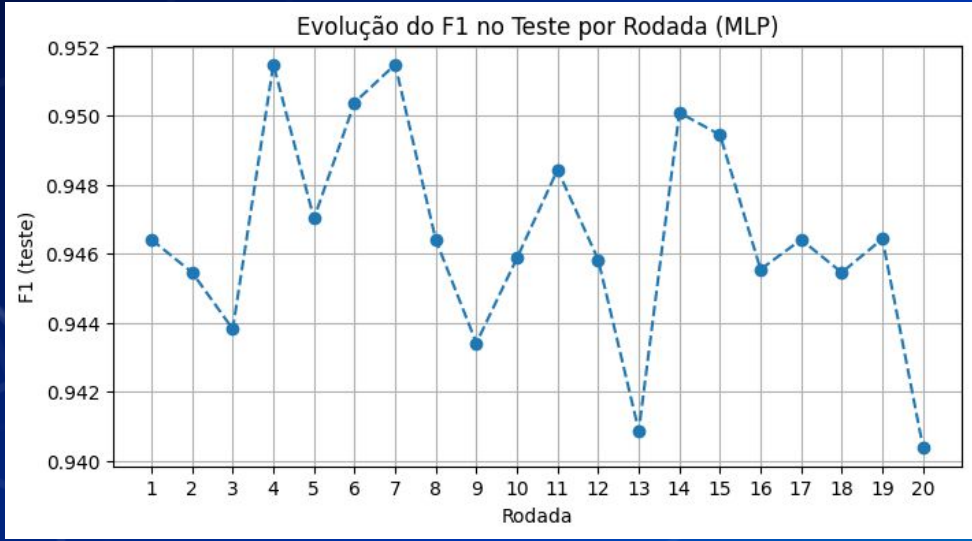
LVQ


Matriz de Confusão - LVQ (Melhor Modelo)
1043
926
281
4643
0
1
True label
0
1
Predicted label
4500
4000
3500
3000
2500
2000
1500
1000
500

MLP

```
param_dist_mlp = {

    'mlp__hidden_layer_sizes': [(50,), (100,), (50,50), (100,50),
(100,100)],

    'mlp__activation': ['relu', 'tanh'],

    'mlp__solver': ['adam', 'sgd'],

    'mlp__alpha': [1e-4, 1e-3, 1e-2],

    'mlp__learning_rate_init': [0.001, 0.01, 0.1]

}
```

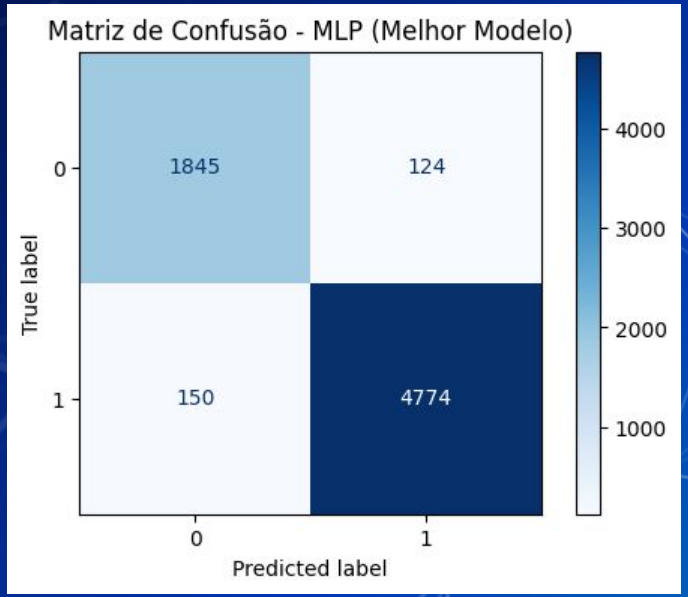
MLP

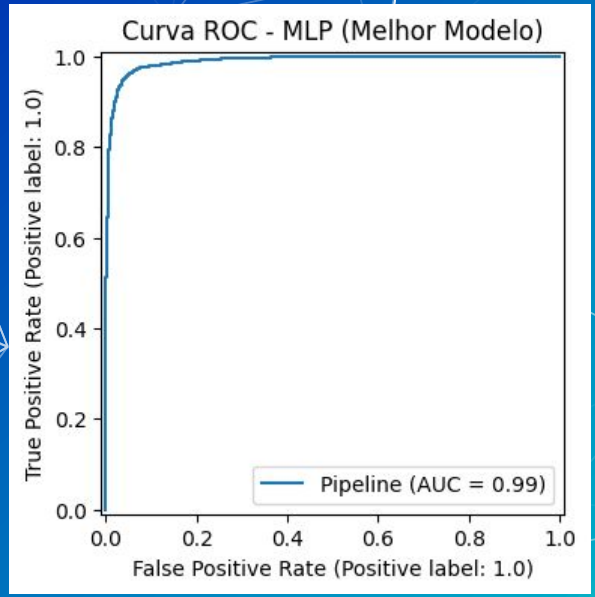



```
Desempenho no Conjunto de Teste:
Acurácia : 0.9602
Precisão : 0.9497
Recall   : 0.9533
F1-score : 0.9515
AUC      : 0.9897
```

```
Melhor rodada: 4
F1 na melhor rodada: 0.9514906711821585
Melhores hiperparâmetros: {'mlp__solver': 'adam', 'mlp__learning_rate_init': 0.001, 'mlp__hidden_layer_sizes': (100, 100), 'mlp__alpha': 0.01, 'mlp__activation': 'tanh'}
```

MLP

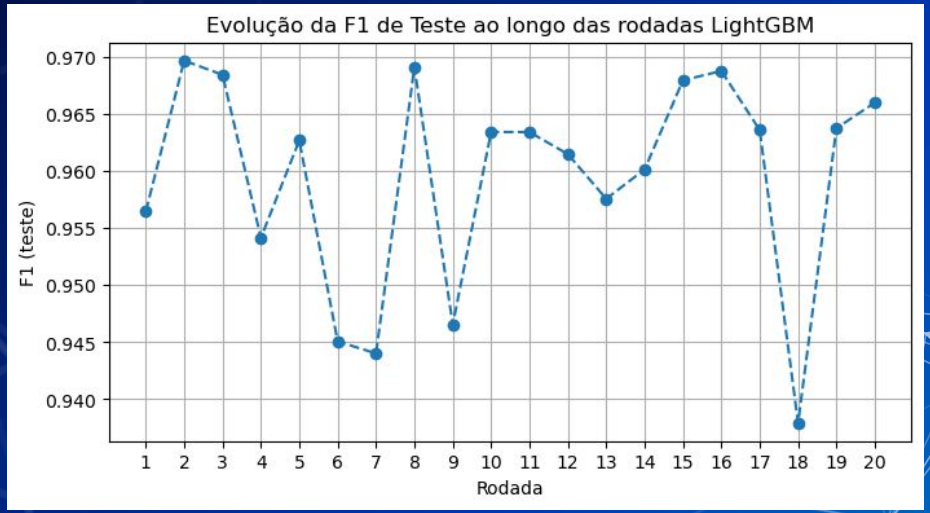

Matriz de Confusão - MLP (Melhor Modelo)
1845
124
150
4774
0
1
True label
Predicted label
4000
3000
2000
1000


Curva ROC - MLP (Melhor Modelo)
True Positive Rate (Positive label: 1.0)
False Positive Rate (Positive label: 1.0)
Pipeline (AUC = 0.99)
0.0
0.2
0.4
0.6
0.8
1.0

LightGBM

```
param_dist_lgbm = {
    'lgbm__n_estimators': [50, 100, 200, 500],
    'lgbm__learning_rate': [0.01, 0.05, 0.1, 0.2],
    'lgbm__max_depth': [-1, 3, 5, 10],
    'lgbm__num_leaves': [10, 20, 31, 40],
    'lgbm__boosting_type': ['gbdt', 'dart'],
    'lgbm__reg_alpha': [0, 0.1, 0.5, 1.0],
    'lgbm__reg_lambda': [0, 0.1, 0.5, 1.0]
}
```

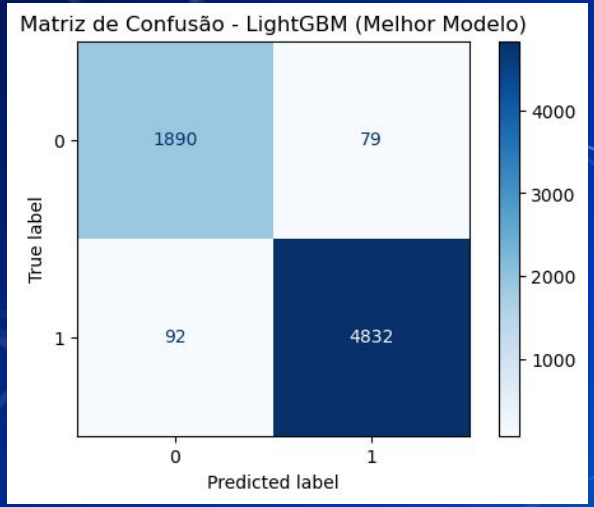
LightGBM

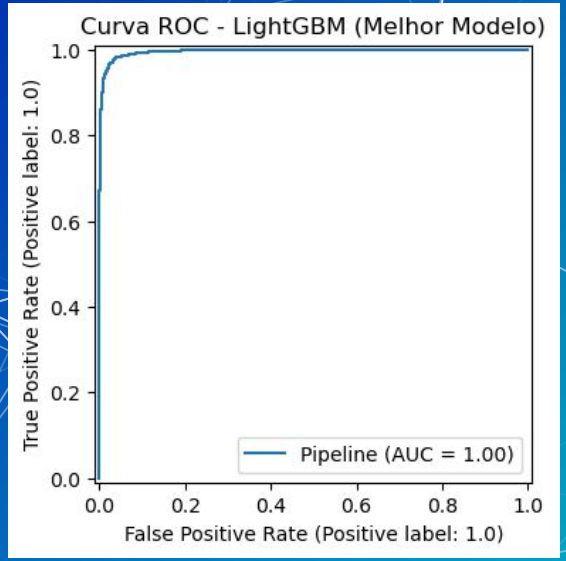



```
Desempenho no Conjunto de Teste:
Acurácia   : 0.9752
Precisão   : 0.9687
Recall     : 0.9706
F1-score   : 0.9697
AUC        : 0.9963
```

```
Melhores hiperparâmetros (validação cruzada): {'lgbm__reg_lambda': 0.5, 'lgbm__reg_alpha': 0, 'lgbm__num_leaves': 31, 'lgbm__n_estimators': 200, 'lgbm__max_depth': 10, 'lgbm__learning_rate': 0.2, 'lgbm__boosting_type': 'gbdt'}
Melhor F1 (validação cruzada - média): 0.966667361578714
F1 no Teste: 0.9660284927618135
```

LightGBM

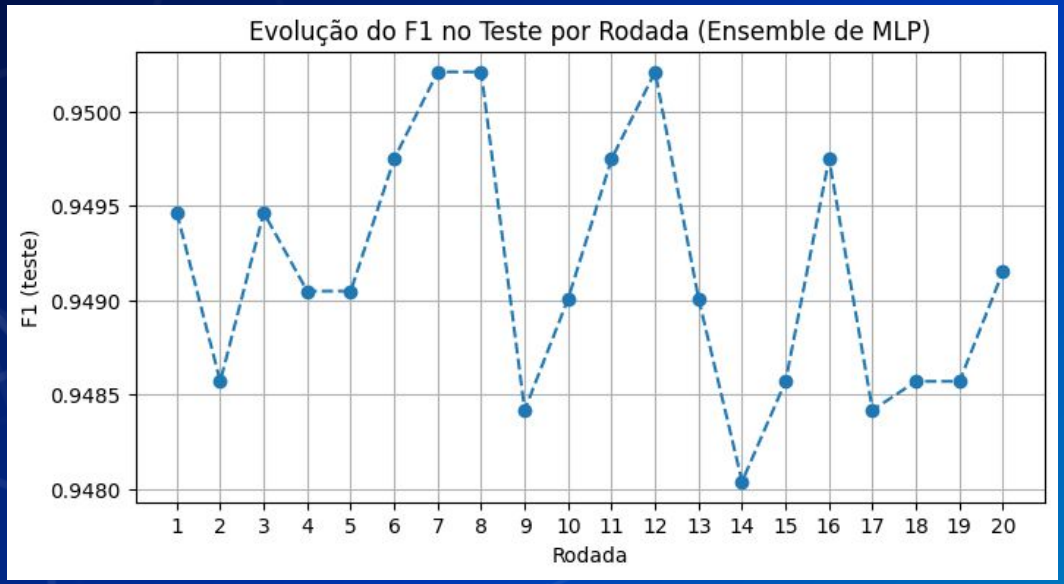

Matriz de Confusão - LightGBM (Melhor Modelo)
1890
79
92
4832
0
1
True label
0
1
Predicted label
4000
3000
2000
1000


Curva ROC - LightGBM (Melhor Modelo)
True Positive Rate (Positive label: 1.0)
False Positive Rate (Positive label: 1.0)
Pipeline (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

Comitê de redes neurais

```
                                                ('mlp1',
MLPClassifier(hidden_layer_sizes=(50,50), random_state=42)),

                                                ('mlp2',
MLPClassifier(hidden_layer_sizes=(100,), random_state=42)),

                                                ('mlp3',
MLPClassifier(hidden_layer_sizes=(50,30), random_state=42))

        ],

             voting='soft'   # 'soft' utiliza médias das
probabilidades; 'hard' utiliza voto majoritário

    ))

])



# Espaço de hiperparâmetros para o VotingClassifier

param_dist_ensemble = {

    'mlp_ensemble__mlp1__alpha': [1e-3, 1e-2],

    'mlp_ensemble__mlp2__alpha': [1e-3, 1e-2],

    'mlp_ensemble__mlp3__alpha': [1e-3, 1e-2],

    'mlp_ensemble__voting': ['hard', 'soft']

}
```

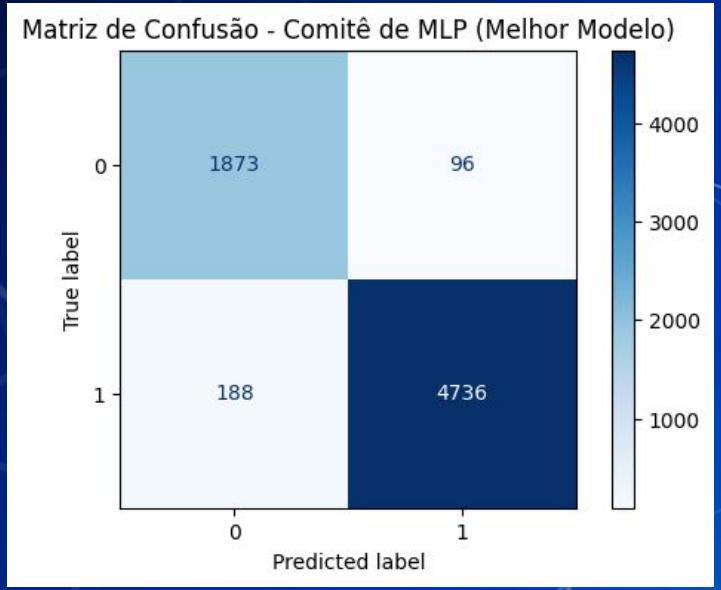
Comitê de redes neurais

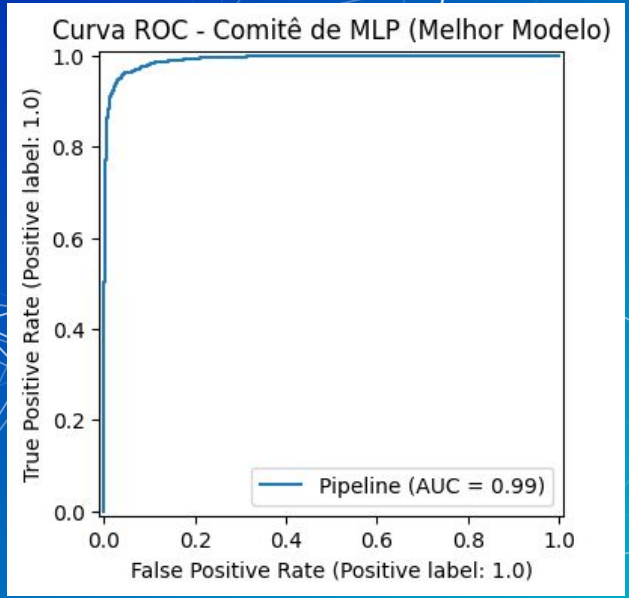



```
Desempenho no Conjunto de Teste:
Acurácia : 0.9588
Precisão : 0.9445
Recall   : 0.9565
F1-score : 0.9502
AUC      : 0.9920
```

```
Melhor rodada: 7
F1 nessa rodada: 0.9502091224386189
Melhores hiperparâmetros: {'mlp_ensemble__voting': 'soft', 'mlp_ensemble__mlp3__alpha': 0.01, 'mlp_ensemble__mlp2__alpha': 0.01, 'mlp_ensemble__mlp1__alpha': 0.001}
```

Comitê de redes neurais

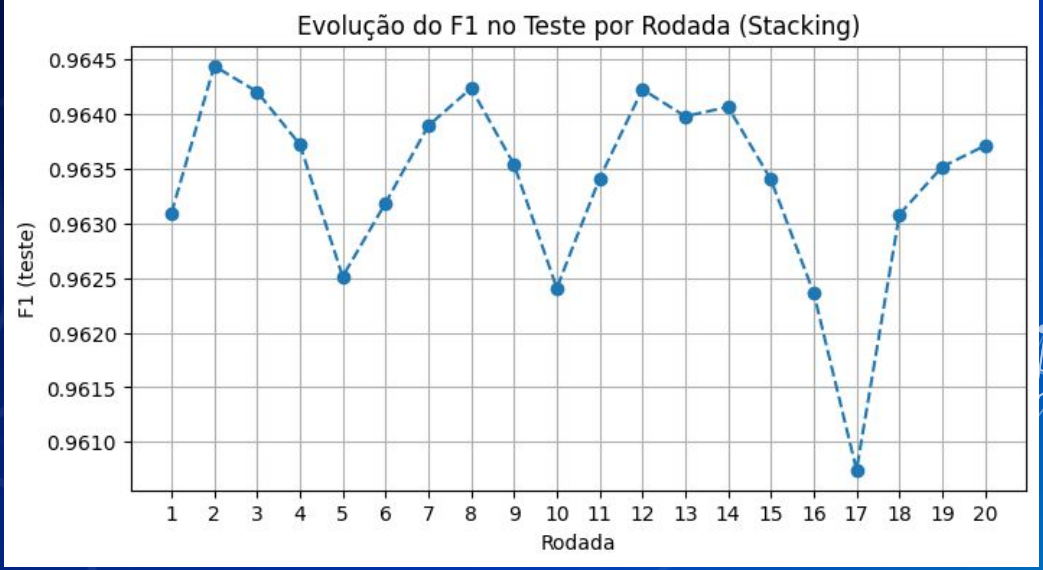




Stacking

```
pipeline_stack = Pipeline([
    ('smote', SMOTE(random_state=42)),
    ('stack', StackingClassifier(
        estimators=[
            ('knn', KNeighborsClassifier()),
            ('rf', RandomForestClassifier(random_state=42)),
            ('lgbm', LGBMClassifier(random_state=42))
        ],
        final_estimator=LogisticRegression(random_state=42)
    ))
])

param_dist_stack = {
            #   Hiperparâmetros   do   estimador   final
(LogisticRegression)
    'stack__final_estimator__C': [0.01, 0.1, 1, 10, 100],
        'stack__final_estimator__penalty':   ['l2'],     #
LogisticRegression utiliza L2 por padrão

    # Hiperparâmetros do KNN base
    'stack__knn__n_neighbors': list(range(1, 21)),

    # Hiperparâmetros da Random Forest base
    'stack__rf__n_estimators': [50, 100, 150],

     # Hiperparâmetros do LGBM base (por exemplo, número de
folhas)
    'stack__lgbm__num_leaves': [31, 50, 70]
}
```

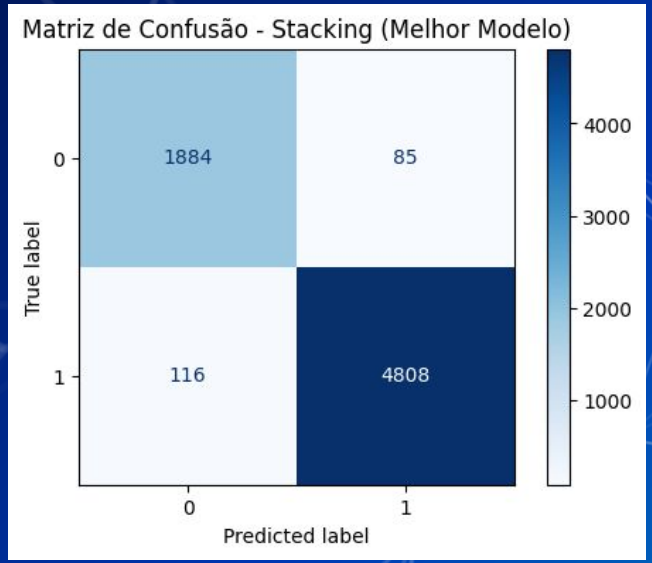
Stacking

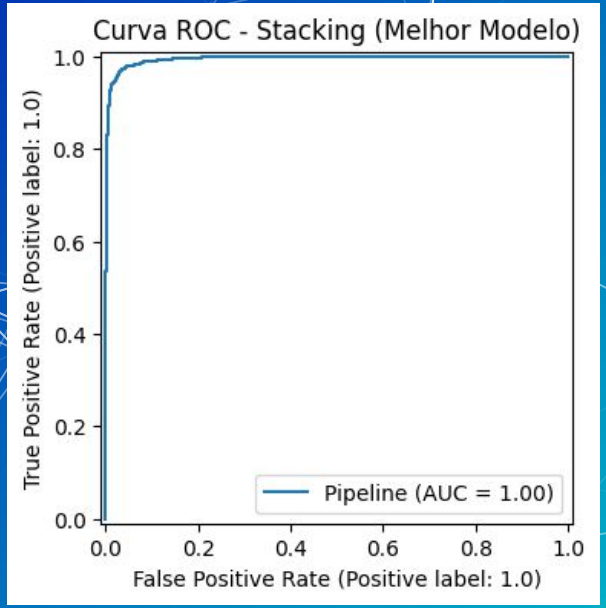



```
Desempenho no Conjunto de Teste:
Acurácia : 0.9708
Precisão : 0.9623
Recall   : 0.9666
F1-score : 0.9644
AUC      : 0.9951
```

```
Melhor rodada: 2
F1 nessa rodada: 0.9644414170091202
Melhores hiperparâmetros: {'stack__rf__n_estimators': 150, 'stack__lgbm__num_leaves': 50, 'stack__knn__n_neighbors': 16, 'stack__final_estimator__penalty': 'l2', 'stack__final_estimator__C': 0.1}
```

Stacking

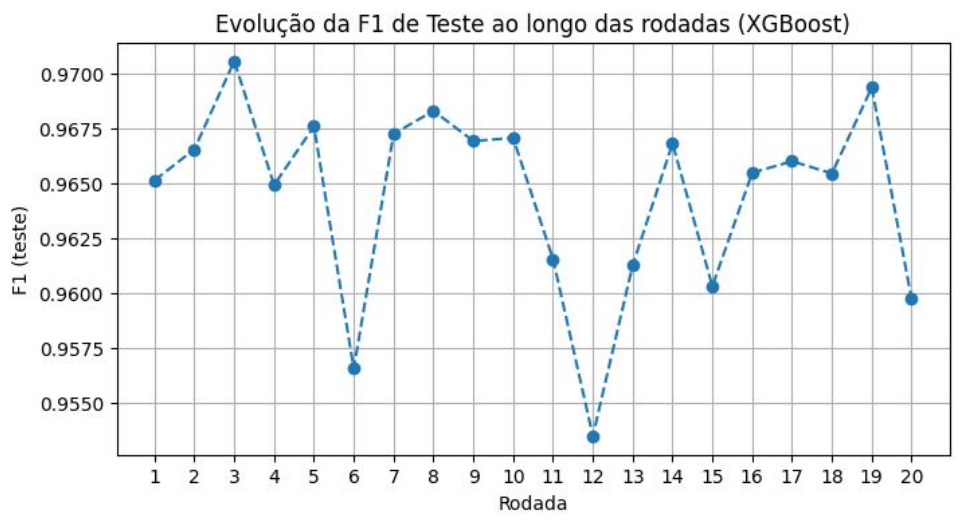

Matriz de Confusão - Stacking (Melhor Modelo)
1884
85
116
4808
0
1
True label
0
1
Predicted label
4000
3000
2000
1000


Curva ROC - Stacking (Melhor Modelo)
True Positive Rate (Positive label: 1.0)
False Positive Rate (Positive label: 1.0)
Pipeline (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

XGBOOST

```python
# Espaço de busca dos hiperparâmetros para o XGBoost
param_dist_xgb = {
    'xgb__n_estimators': range(50, 301, 50),      # Número de árvores: 50, 100, 150, 200, 250, 300
    'xgb__max_depth': range(3, 11, 2),            # Profundidade máxima: 3, 5, 7, 9
    'xgb__learning_rate': [0.01, 0.05, 0.1, 0.2], # Taxa de aprendizado
    'xgb__subsample': [0.6, 0.8, 1.0],           # Amostragem das instâncias
    'xgb__colsample_bytree': [0.6, 0.8, 1.0],    # Amostragem das colunas
    'xgb__gamma': [0, 0.1, 0.2, 0.3],            # Regularização gamma
}

# Configurando a validação cruzada estratificada (k=5)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

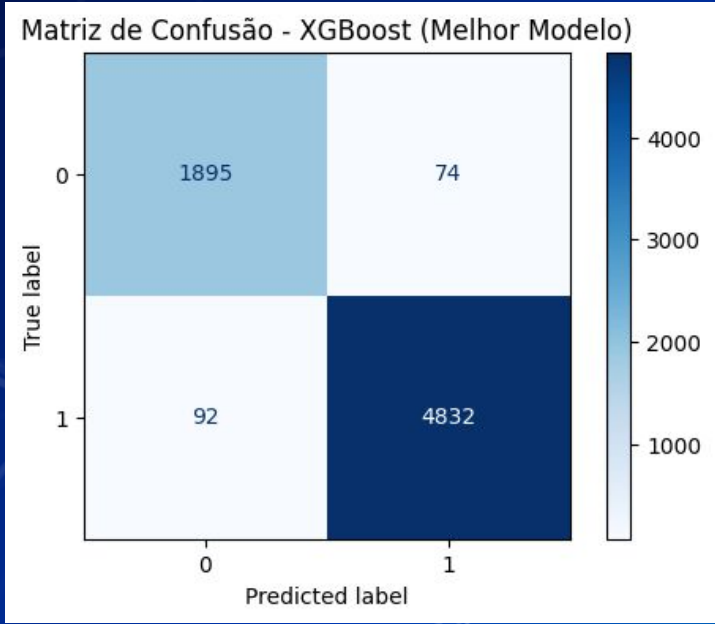
XGBOOST

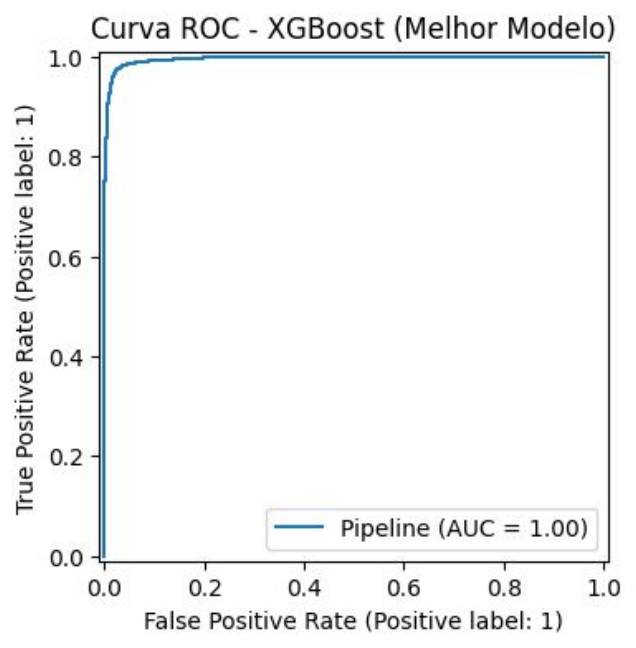



```
Desempenho no Conjunto de Teste:
Acurácia  : 0.9759
Precisão  : 0.9693
Recall    : 0.9719
F1-score  : 0.9706
AUC       : 0.9961
```

```
Melhor rodada: 3
F1 nessa rodada: 0.9705756711414573
Melhores hiperparâmetros: {'xgb__subsample': 0.8, 'xgb__n_estimators': 250, 'xgb__max_depth': 9, 'xgb__learning_rate': 0.2, 'xgb__gamma': 0.1, 'xgb__colsample_bytree': 0.6}
```

XGBOOST


Matriz de Confusão - XGBoost (Melhor Modelo)
1895
74
92
4832
0
1
True label
0
1
Predicted label
4000
3000
2000
1000


Curva ROC - XGBoost (Melhor Modelo)
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
Pipeline (AUC = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

Deployment e Conclusões

Deployment

- Significa levar o modelo para uso real.
- Pode ser integrado a um sistema ou usado para gerar insights.
- Exemplos: APIs, automação, dashboards, relatórios.

Onde esse modelo pode ser usado?

- Identificação de registros duplicados em sistemas.
- Plataformas de mapas, delivery, logística, redes varejistas.
- Redução de inconsistências e melhoria na tomada de decisão.

Qual foi o melhor modelo?

- XGBoost → Acurácia 97.59%, F1-score 97.06%, AUC 99.61%.
- LightGBM→ Acurácia 97.52%, F1-score 96.97%, AUC 99.63%.
- Ambos modelos conseguiram excelentes desempenhos

XGBoost

```
Desempenho no Conjunto de Teste:
Acurácia   : 0.9759
Precisão   : 0.9693
Recall     : 0.9719
F1-score   : 0.9706
AUC        : 0.9961
```

LightGBM

```
Desempenho no Conjunto de Teste:
Acurácia   : 0.9752
Precisão   : 0.9687
Recall     : 0.9706
F1-score   : 0.9697
AUC        : 0.9963
```

O que pode ser melhorado?

- Coletar mais dados para melhorar generalização.
- Testar técnicas de feature engineering avançadas.
- Criar uma API ou sistema de automação para validar o modelo no mundo real.

Obrigado!