UFMG

Detecção de Phishing

Yago Aquino

Link do projeto: https://github.com/Yago-Pacheco/ufmg/tree/main/especializacao_ciencia_de_dados/topicos_em_ciencia_de_dados/tp_3

Introdução

O phishing representa uma das principais ameaças à segurança digital, onde atacantes criam websites fraudulentos que imitam empresas legítimas com o objetivo de roubar informações confidenciais como senhas, dados bancários e números de segurança social. Com milhares de novos sites maliciosos sendo criados diariamente, métodos tradicionais baseados em listas negras se tornam ineficazes, uma vez que não conseguem detectar ameaças emergentes em tempo real.

Este trabalho apresenta uma abordagem baseada em machine learning para detecção automatizada de websites de phishing, utilizando um conjunto de 30 características extraídas automaticamente das URLs e propriedades dos sites. Através da análise comparativa entre diferentes algoritmos de classificação, buscou desenvolver um modelo capaz de identificar padrões suspeitos que distinguem sites legítimos de fraudulentos.

UFMG

Objetivos

1. Entendimento dos Dados
2. Pré-processamento e Visualização
3. Modelagem
4. Avaliação dos Resultados
5. Expansão da Modelagem

1. Entendimento dos Dados

Características do Dataset

- Tamanho: 11055 linhas e 31 colunas
- Fonte: PhishTank e dados de websites legítimos
- Variável Target: Result
- Tipo: Todas as features (colunas) são categóricas rotuladas como -1, 0, 1. Sendo que -1 indica a presença de uma características suspeita, 0 indica a presença de uma características de um site legítimo e 1 indica uma características de phishing.

Organização das features

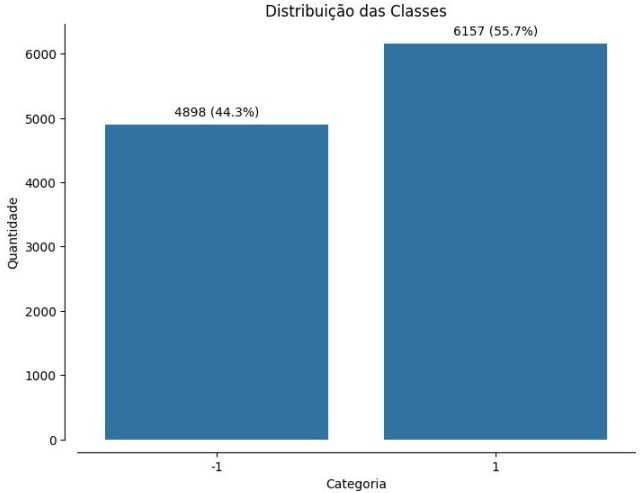
- Baseadas na URL (9 features)
- Segurança e SSL (4 features)
- Conteúdo HTML/JavaScript (8 features)
- Reputação e Tráfego (5 features)
- Características Técnicas (4 features)

Acesso aos dados: https://archive-beta.ics.uci.edu/dataset/327/phishing+websites

2. Pré-processamento e Visualização

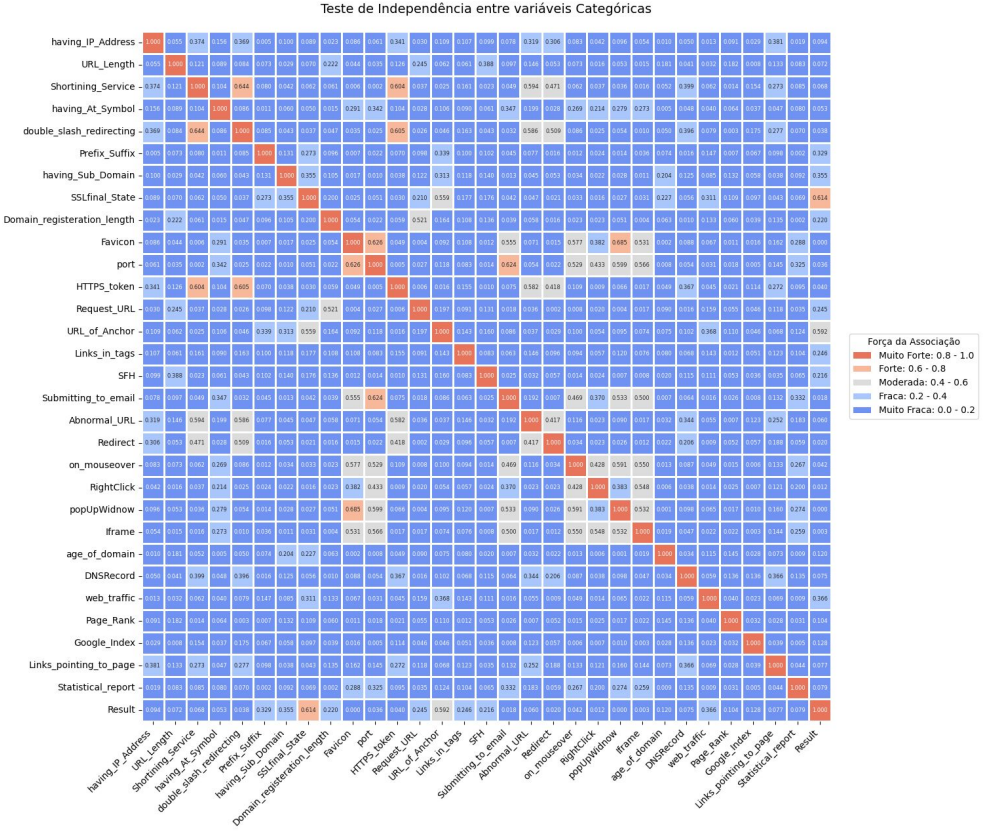
Antes de iniciar a modelagem, se realizou uma verificação dos dados revelou um dataset bem estruturado, sem valores ausentes e com codificação consistente das features categóricas. Todas as 30 características foram adequadamente transformadas em valores numéricos (-1, 0, 1), facilitando o processamento pelos algoritmos de machine learning.

Entretanto, a análise exploratória identificou um desafio importante: o desbalanceamento entre as classes target. Essa questão é crítica em problemas de detecção de phishing, pois pode levar o modelo a favorecer a classe majoritária, comprometendo a capacidade de identificar sites maliciosos.



2. Pré-processamento e Visualização

Antes de iniciar a modelagem, se realizou uma verificação dos dados que revelou um dataset bem estruturado, sem valores ausentes e com codificação consistente das features categóricas. Todas as 30 características foram adequadamente codificadas em valores numéricos (-1, 0, 1). Se aplicou o teste Chi-quadrado (χ^2) para identificar as features com maior associação estatística com a variável target. Esta abordagem permitiu focar nas características mais discriminativas, otimizando a análise exploratória.



2. Pré-processamento e Visualização

Quando se analisou a coluna Results e as features selecionadas pelo teste Chi², algumas características chamaram especial atenção, apresentando associações que variam de fortes a moderadas. Features com Maior Poder Discriminativo

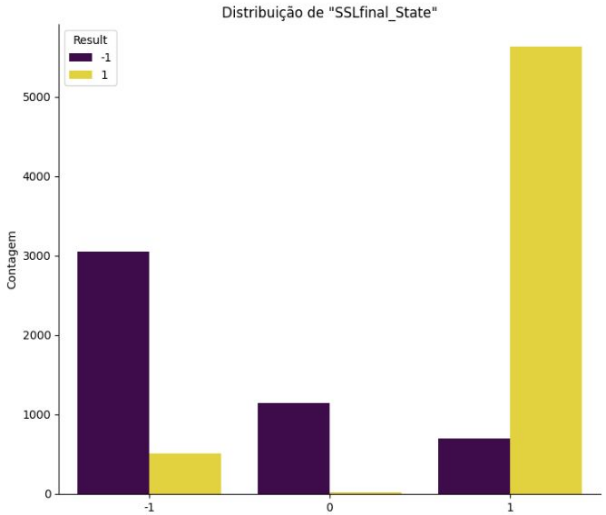
- SSLfinal_State: Avalia a validade do certificado SSL/TLS, verificando não apenas a existência do HTTPS, mas também a autoridade certificadora e idade do certificado.
- Prefix_Suffix: Detecta a presença do caractere hífen (-) no domínio, técnica comum em phishing.
- URL_of_Anchor: Analisa a percentagem de âncoras que apontam para domínios externos ou não levam a lugar algum.
- having_Sub_Domain: Examina múltiplos subdomínios na URL (ex: login.conta.banco.sitefalso.com).
- web_traffic: Mede popularidade através do ranking Alexa - sites sem tráfego são classificados como suspeitos.
- Links_in_tags: Verificação técnica do código-fonte, analisando links dentro das tags HTML.

Para melhor compreensão, se verificou a distribuição específica dessas features selecionadas, focando nas mais discriminativas em vez de analisar todas as 30 colunas.

2. Pré-processamento e Visualização

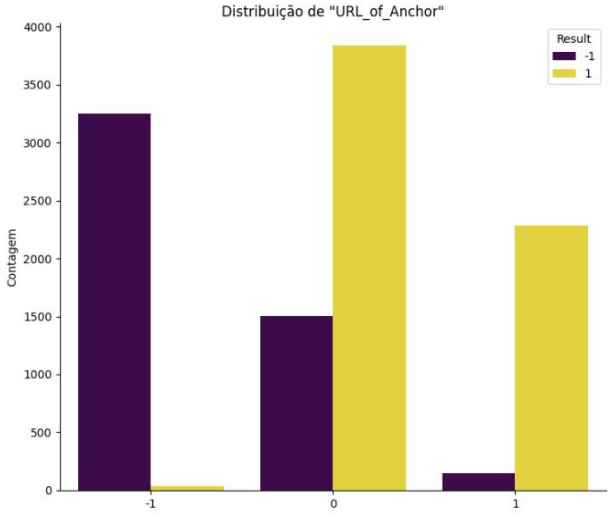
SSLfinal_State - Feature mais discriminativa:

- Valor 1 (SSL confiável): Massivamente legítimo (~5700 vs ~700)
- Valor -1 (sem SSL): Predominantemente phishing (~3100 vs ~500)
- Padrão claro: Sites legítimos investem em certificados SSL válidos



URL_of_Anchor - Segundo melhor discriminador:

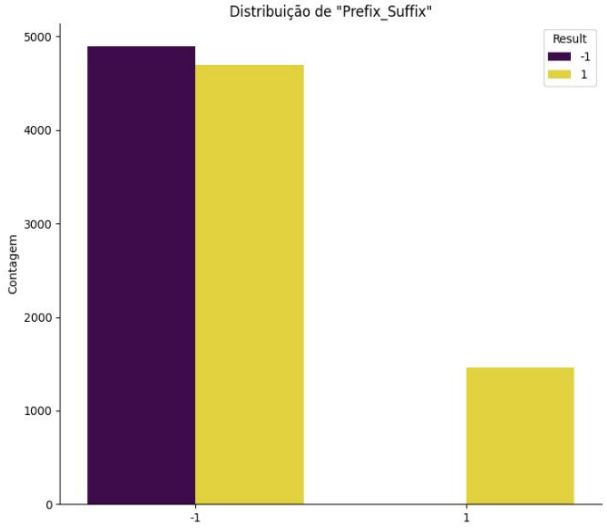
- Valor -1 (âncoras externas): Quase exclusivamente phishing (~3300 vs ~50)
- Valor 0/1: Distribuição favorece sites legítimos
- Phishers frequentemente direcionam para domínios externos



2. Pré-processamento e Visualização

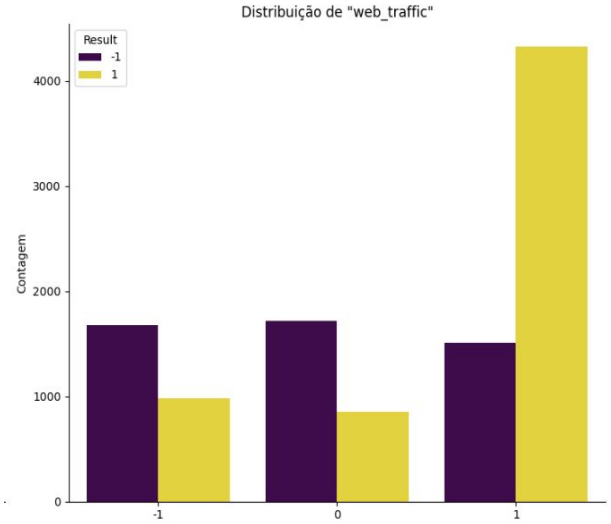
Prefix_Suffix - Comportamento interessante:

- Valor -1 (sem hífen): Distribuição equilibrada entre as classes
- Valor 1 (com hífen): Surpreendentemente, mais comum em sites legítimos
- Contraria a expectativa inicial - hífens não são indicadores fortes de phishing



web_traffic - Indicador de legitimidade:

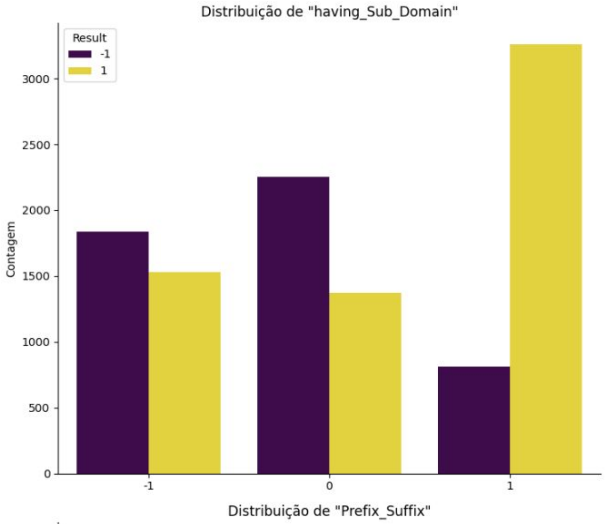
- Valor 1 (alto tráfego): Forte tendência para sites legítimos (~4400 vs ~1500)
- Sites phishing raramente alcançam rankings elevados de tráfego

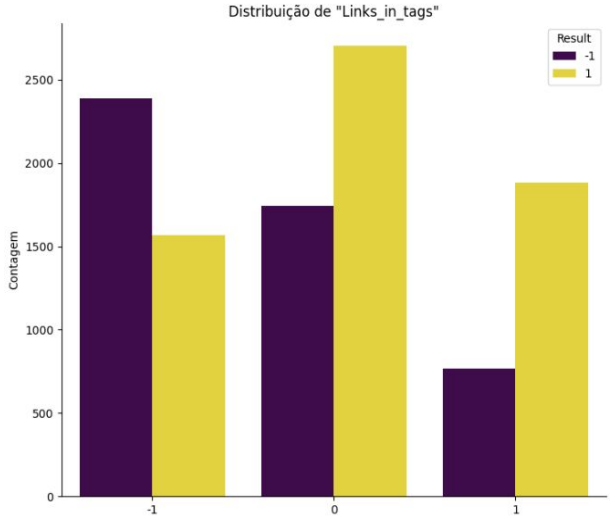


2. Pré-processamento e Visualização

having_Sub_Domain e Links_in_tags - Discriminação moderada:

- Padrões menos definidos, mas ainda contributivos para o modelo





3. Modelagem

Para estabelecer uma baseline sólida, se implementou uma abordagem sistemática utilizando as configurações padrão das bibliotecas. Os dados foram divididos em conjuntos de treino e teste, aplicando-se validação cruzada no conjunto de treino para garantir robustez na avaliação dos modelos. Os modelos selecionados foram:

Regressão Logística

- Modelo linear interpretável e eficiente
- Excelente baseline para problemas de classificação binária
- Fornece probabilidades e coeficientes interpretáveis
- Ideal para identificar features mais influentes na detecção de phishing

Random Forest

- Ensemble de árvores de decisão com robustez a overfitting
- Captura interações não-lineares entre features
- Fornece importância das variáveis automaticamente
- Eficaz em datasets com features categóricas como o nosso

LightGBM

- Gradient boosting otimizado para velocidade e performance
- Excelente para datasets com muitas features categóricas
- Reconhecido por alta performance em competições de ML
- Eficiente no tratamento de desbalanceamento de classes

3. Modelagem

Toda a etapa de modelagem pode ser visualizada nas imagens abaixo.

```python
# Separando as variáveis de treino e teste
X = df.drop('Result', axis=1)
y = df['Result']

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

print(f"Treino: {X_train.shape[0]} amostras")
print(f"Teste: {X_test.shape[0]} amostras\n")
```

```python
# Instanciando os modelos
models = {
    'Regressão Logística': LogisticRegression(random_state=42, max_iter=1000),
    'Random Forest': RandomForestClassifier(random_state=42),
    'LGBM': LGBMClassifier(random_state=42, verbose=-1)
}

cv_results = {}

scoring_metrics = ['accuracy', 'f1', 'recall', 'precision', 'roc_auc']

for name, model in models.items():
  print(f"\n{name}:")

  cv_results[name] = {}

  for metric in scoring_metrics:
      cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring=metric)
      cv_results[name][metric] = {
          'mean': cv_scores.mean(),
          'std': cv_scores.std(),
          'scores': cv_scores
      }

  acc_mean = cv_results[name]['accuracy']['mean']
  acc_std = cv_results[name]['accuracy']['std']
  f1_mean = cv_results[name]['f1']['mean']
  f1_std = cv_results[name]['f1']['std']
  recall_mean = cv_results[name]['recall']['mean']
  precision_mean = cv_results[name]['precision']['mean']
  auc_mean = cv_results[name]['roc_auc']['mean']

  print(f"Acurácia: {acc_mean:.4f} (+/- {acc_std * 2:.4f})")
  print(f"F1-Score: {f1_mean:.4f} (+/- {f1_std * 2:.4f})")
  print(f"Recall:   {recall_mean:.4f} (detectar phishing)")
  print(f"Precision:{precision_mean:.4f} (evitar falsos alarmes)")
  print(f"AUC-ROC:  {auc_mean:.4f}")
```

```python
final_results = {}

for name, model in models.items():
  print(f"\n{name}:")

  model.fit(X_train, y_train)

  y_pred = model.predict(X_test)
  y_pred_proba = model.predict_proba(X_test)[:, 1] if hasattr(model, 'predict_proba') else None

  accuracy = accuracy_score(y_test, y_pred)
  f1 = f1_score(y_test, y_pred)
  recall = recall_score(y_test, y_pred)
  precision = precision_score(y_test, y_pred)
  auc_score = roc_auc_score(y_test, y_pred_proba) if y_pred_proba is not None else None

  final_results[name] = {
      'model': model,
      'predictions': y_pred,
      'probabilities': y_pred_proba,
      'accuracy': accuracy,
      'f1': f1,
      'recall': recall,
      'precision': precision,
      'auc': auc_score
  }

  print(f"Accuracy:  {accuracy:.4f}")
  print(f" F1-Score: {f1:.4f} (métrica principal)")
  print(f"Recall:    {recall:.4f} ({recall*100:.1f}% dos phishing detectados)")
  print(f"Precision: {precision:.4f} ({precision*100:.1f}% dos alertas são reais)")
  print(f"AUC-ROC:   {auc_score:.4f}" if auc_score else "AUC-ROC: N/A")
```

4. Avaliação dos Resultados

Após a etapa de modelagem, se iniciou a comparação sistemática dos resultados obtidos pelos três algoritmos. A seleção do F1-score se justifica pela natureza crítica do problema e pelo desbalanceamento do dataset. Como média harmônica entre precisão e recall, oferece avaliação equilibrada considerando tanto a detecção de sites maliciosos quanto a minimização de falsos alarmes, essencial em sistemas de segurança onde falsos negativos têm consequências graves. Os resultados demonstram que o Random Forest alcançou o melhor desempenho (F1-Score de 0.9770 e recall de 98.5%), superando LGBM e Regressão Logística. Esta superioridade se explica pela natureza categórica dos dados de phishing. O Random Forest é particularmente eficaz com features binárias (presença de IP, símbolos especiais, redirecionamentos) devido ao seu mecanismo de bootstrap sampling e robustez ao overfitting. O LGBM tende a brilhar em datasets maiores com features numéricas, não sendo o caso aqui. O recall excepcional de 98.5% do Random Forest é crucial para aplicações de segurança, onde perder um site malicioso tem consequências mais graves que gerar falsos alarmes.

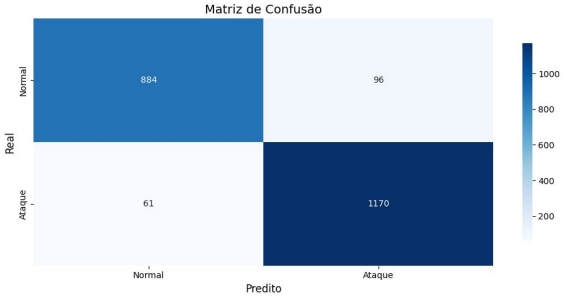
```
Regressão Logística:
Accuracy:  0.9290
 F1-Score: 0.9371 (métrica principal)
Recall:    0.9504 (95.0% dos phishing detectados)
Precision: 0.9242 (92.4% dos alertas são reais)
AUC-ROC:   0.9808

Random Forest:
Accuracy:  0.9742
 F1-Score: 0.9770 (métrica principal)
Recall:    0.9846 (98.5% dos phishing detectados)
Precision: 0.9696 (97.0% dos alertas são reais)
AUC-ROC:   0.9977

LGBM:
Accuracy:  0.9679
 F1-Score: 0.9713 (métrica principal)
Recall:    0.9764 (97.6% dos phishing detectados)
Precision: 0.9662 (96.6% dos alertas são reais)
AUC-ROC:   0.9967
```

	Modelo	F1-Score	Recall	Precision	AUC-ROC
1	Random Forest	0.9770	0.9846	0.9696	0.9977
2	LGBM	0.9713	0.9764	0.9662	0.9967
0	Regressão Logística	0.9371	0.9504	0.9242	0.9808

4. Avaliação dos Resultados

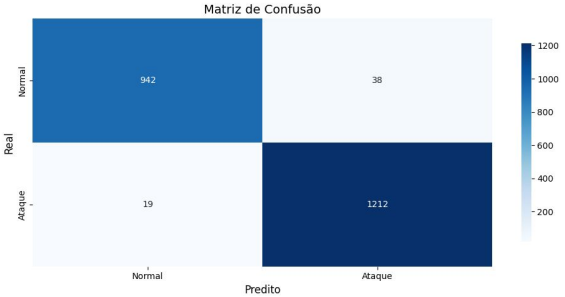
Regressão Logística:

- True Negatives (Legítimos corretos): 884
- False Positives (Falsos alarmes): 96
- False Negatives (Phishing perdido): 61
- True Positives (Phishing detectado): 1170



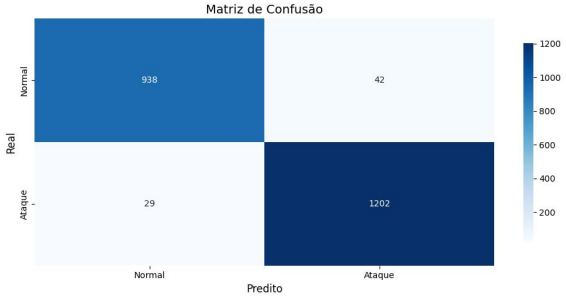
Random Forest:

- True Negatives (Legítimos corretos): 942
- False Positives (Falsos alarmes): 38
- False Negatives (Phishing perdido): 19
- True Positives (Phishing detectado): 1212



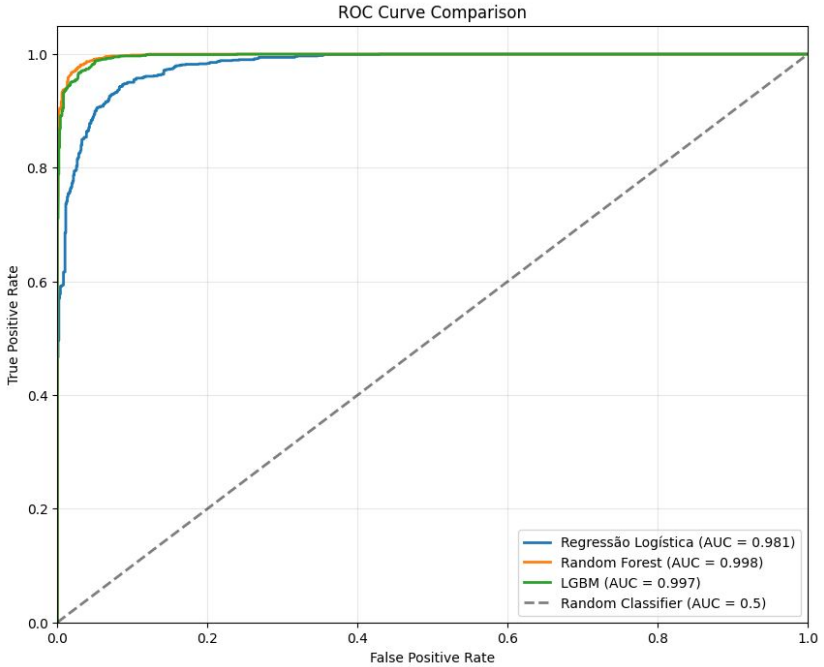
LGBM:

- True Negatives (Legítimos corretos): 938
- False Positives (Falsos alarmes): 42
- False Negatives (Phishing perdido): 29
- True Positives (Phishing detectado): 1202



4. Avaliação dos Resultados

A curva ROC confirma a excelente performance dos três modelos, com valores de AUC excelentes. O Random Forest mantém a liderança (AUC = 0.998), seguido pelo LGBM (AUC = 0.997) e Regressão Logística (AUC = 0.981).



5. Expansão da Modelagem

Após estabelecer a baseline com excelentes resultados, se iniciou uma fase de expansão e refinamento visando explorar o potencial máximo dos algoritmos através de engenharia de features e otimização de hiperparâmetros.

Modelo 2

- Criação de novas features baseadas em combinações e transformações das variáveis originais
- Aplicação de técnicas de balanceamento para tratar o desbalanceamento das classes

Modelo 3

- Replica todos os processos do Modelo 2 (novas features + balanceamento)
- Adiciona otimização automática de hiperparâmetros usando biblioteca Optuna
- Busca sistemática pelos melhores parâmetros através de algoritmos de otimização bayesiana

5. Expansão da Modelagem

Como o Modelo 3 representa apenas uma extensão incremental do Modelo 2, não se apresentou os processos separadamente, uma vez que o terceiro modelo reproduz todas as etapas do segundo, acrescentando exclusivamente a camada de hiperparametrização.

```python
# Criação de features
df['url_risk'] = (
    (df['having_IP_Address'] == 1).astype(int) +
    (df['URL_Length'] == 1).astype(int) +
    (df['Shortining_Service'] == 1).astype(int) +
    (df['having_At_Symbol'] == 1).astype(int) +
    (df['double_slash_redirecting'] == 1).astype(int)
)

df['domain_trust'] = (
    (df['age_of_domain'] == 1).astype(int) +
    (df['DNSRecord'] == 1).astype(int) +
    (df['Domain_registeration_length'] == 1).astype(int)
)

df['reputation'] = (
    (df['web_traffic'] == 1).astype(int) +
    (df['Page_Rank'] == 1).astype(int) +
    (df['Google_Index'] == 1).astype(int) +
    (df['Links_pointing_to_page'] == 1).astype(int)
)

```

```python
# Balanceamento dos dados
balancer = SMOTETomek(random_state=42)
X_balanced, y_balanced = balancer.fit_resample(X_train, y_train)

print(f"Treino: {X_balanced.shape[0]} amostras")
```

```python
############### SÓ RODE SE TIVER PELO MENOS 38min PARA ESPERAR TODO O PROCESSO DE OTIMIZAÇÃO ###############
study_lr = optimize_model(objective_lr, "Regressão Logística")
study_rf = optimize_model(objective_rf, "Random Forest")
study_lgbm = optimize_model(objective_lgbm, "LightGBM")
```

5. Expansão da Modelagem

Como o Modelo 3 representa apenas uma extensão incremental do Modelo 2, não se apresentou os processos separadamente, uma vez que o terceiro modelo reproduz todas as etapas do segundo, acrescentando exclusivamente a camada de hiperparametrização.

```python
# Criação de features
df['url_risk'] = (
    (df['having_IP_Address'] == 1).astype(int) +
    (df['URL_Length'] == 1).astype(int) +
    (df['Shortining_Service'] == 1).astype(int) +
    (df['having_At_Symbol'] == 1).astype(int) +
    (df['double_slash_redirecting'] == 1).astype(int)
)

df['domain_trust'] = (
    (df['age_of_domain'] == 1).astype(int) +
    (df['DNSRecord'] == 1).astype(int) +
    (df['Domain_registeration_length'] == 1).astype(int)
)

df['reputation'] = (
    (df['web_traffic'] == 1).astype(int) +
    (df['Page_Rank'] == 1).astype(int) +
    (df['Google_Index'] == 1).astype(int) +
    (df['Links_pointing_to_page'] == 1).astype(int)
)

```

```python
study_lr = optimize_model(objective_lr, "Regressão Logística")
study_rf = optimize_model(objective_rf, "Random Forest")
study_lgbm = optimize_model(objective_lgbm, "LightGBM")
```

```
Regressão Logística
Melhor F1-Score: 0.9339
Melhores parâmetros:
  C: 2.0076447814978837
  solver: saga
  max_iter: 453
  class_weight: None
  penalty_saga: l1
```

```
LGBM
Melhor F1-Score: 0.9739
Melhores parâmetros:
  num_leaves: 73
  learning_rate: 0.15714843121189861
  n_estimators: 254
  max_depth: 7
  min_child_samples: 12
  subsample: 0.892711088697747
  colsample_bytree: 0.6121301967092603
  reg_alpha: 4.435974433960475e-05
  reg_lambda: 1.0645287668011355e-08
  class_weight: None
```

```
Random Forest
Melhor F1-Score: 0.9721
Melhores parâmetros:
  n_estimators: 217
  max_depth: 19
  min_samples_split: 2
  min_samples_leaf: 1
  max_features: log2
  bootstrap: True
  class_weight: balanced_subsample
```

```python
# Balanceamento dos dados
balancer = SMOTETomek(random_state=42)
X_balanced, y_balanced = balancer.fit_resample(X_train, y_train)

print(f"Treino: {X_balanced.shape[0]} amostras")
```

```
Treino: 9818 amostras
```

5. Expansão da Modelagem

A análise comparativa dos F1-Scores revela insights importantes sobre o impacto das técnicas aplicadas.

- Random Forest: Mantém liderança consistente em todas as versões (0.9770 → 0.9757), demonstrando estabilidade e robustez. A ligeira variação entre modelos indica que o algoritmo já estava bem otimizado na configuração baseline.
- Regressão Logística: Apresenta melhora modesta com feature engineering e balanceamento (0.9371 → 0.9391), sugerindo que o modelo linear se beneficia das novas características criadas, embora de forma limitada.
- LGBM: Mostra o maior ganho com hiperparametrização (0.9713 → 0.9761), confirmando que este algoritmo requer maior ajuste fino para alcançar seu potencial máximo, sendo mais sensível à configuração de parâmetros.

As diferenças sutis entre os modelos indicam que a baseline já capturava eficientemente os padrões dos dados. O Random Forest permanece como escolha ideal, combinando alta performance com estabilidade, enquanto a hiperparametrização se mostrou mais impactante para o LGBM do que para os demais algoritmos. A tabela abaixo apresenta essas comparações.

Teste	Métrica	Modelo_v1	Modelo_v2	Modelo_v3
Regressão Logística	F1-Score	0,9371	0,9391	0,9390
Random Forest	F1-Score	0,9770	0,9757	0,9757
LGBM	F1-Score	0,9713	0,9713	0,9761

6. Conclusão

A abordagem Random Forest baseline se mostrou mais efetiva, evidenciando que nem sempre complexidade adicional resulta em melhor performance. As principais lições incluem:

- Efetividade dos modelos
  - Random Forest demonstrou superioridade consistente devido à natureza categórica dos dados de phishing
  - Baseline robusta: Os modelos iniciais já capturavam eficientemente os padrões, limitando ganhos de técnicas avançadas
  - Hiperparametrização seletiva: LGBM se beneficiou mais da otimização que Random Forest, indicando diferentes sensibilidades algoritmo-específicas
- Limitações
  - Técnicas de phishing evoluem constantemente, exigindo atualizações frequentes
  - Features categóricas podem não capturar nuances sutis de ataques sofisticados
- Trabalho Futuros
  - Explorar deep learning para capturar padrões mais complexos
  - Validação externa com datasets de diferentes fontes e períodos
  - Investigar interpretabilidade através de técnicas como SHAP para explicar decisões do modelo

Referências

LIGHTGBM. LightGBM 3.3.5 Documentation. Disponível em: https://lightgbm.readthedocs.io/en/stable/. Acessado em: 12-09-2025.

MOHAMMAD, Rami M.; THABTAH, Fadi; MCCLUSKEY, Lee. Phishing Websites. UCI Machine Learning Repository, 2015. Disponível em: https://archive-beta.ics.uci.edu/dataset/327/phishing+websites. Acessado em: 12-09-2025.

SCIKIT-LEARN. sklearn.ensemble.RandomForestClassifier. Scikit-Learn 0.20.3 Documentation, 2025. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Acessado em: 12-09-2025.

SCIKIT-LEARN. sklearn.linear_model.LogisticRegression. Scikit-Learn Documentation. Disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Acessado em: 12-09-2025.