Lista #4

Curso: Ciência da Computação

Disciplina: Inteligência Artificial

Profª. Cristiane Neri Nobre

Data de entrega: 23/03

Valor: 1,5 ponto

Aluno: Lucas Henrique Rocha Hauck
GitHub: https://github.com/o-hauck/IA

Questão 01

Utilizando o otimizador **BayesSearchCV** (from skopt import **BayesSearchCV**), ajuste os hiperparâmetros do Random Forest e Árvore de decisão para o problema do TITANIC.

Que modelo obteve o melhor desempenho? Quais os valores das métricas de avaliação?

Os atributos mais relevantes indicados pelo Random Forest e árvore de decisão são os mesmos? Discuta os resultados obtidos.

Eu fiz o ajuste dos hiperparâmetros usando o BayesSearchCV, que é uma forma mais eficiente de encontrar os melhores parâmetros para os modelos, ao invés de uma busca exaustiva. No Random Forest, otimizei o número de árvores (n_estimators) e a profundidade máxima (max_depth). Já na Árvore de Decisão, os parâmetros ajustados foram a profundidade máxima (max_depth) e o número mínimo de amostras para divisão (min_samples_split).

Os resultados mostraram que o Random Forest teve um desempenho melhor: alcançou 83,15% de acurácia, 88% de precisão, 64,71% de recall e F1-Score de 74,58%. A Árvore de Decisão, por outro lado, teve 78,09% de acurácia, 73,02% de precisão, 67,65% de recall e F1-Score de 70,23%. Apesar da Árvore de Decisão ter um recall um pouco maior, o Random Forest teve mais precisão e melhor F1-Score, o que é um bom indicativo de que ele conseguiu classificar melhor os sobreviventes.

Sobre os hiperparâmetros e ajustes:

- O uso do BayesSearchCV foi importante para otimizar os parâmetros de maneira mais eficiente, ajudando a evitar overfitting e a encontrar o equilíbrio certo.
- No caso do Random Forest, eu foquei no número de árvores e na profundidade máxima para controlar o overfitting e garantir uma boa generalização.

- Já na Árvore de Decisão, os ajustes na profundidade e no número mínimo de amostras para divisão ajudaram a evitar que a árvore ficasse muito complexa e sensível a pequenos ruídos nos dados.

Esses ajustes foram essenciais para que os modelos não ficassem "falsamente bons" no treinamento e conseguissem generalizar melhor nos dados de teste.

Os atributos mais relevantes indicados pelo **Random Forest** e pela **Árvore de Decisão** não são totalmente os mesmos, mas são similares variando principalmente nos pesos:

- **Random Forest**:
    - Sex: 0.3906
    - Fare: 0.2593
    - Age: 0.1877
    - Pclass: 0.1259
    - Embarked: 0.0364
- **Árvore de Decisão**:
    - Sex: 0.4253
    - Pclass: 0.1397
    - Age: 0.1681
    - Fare: 0.2463
    - Embarked: 0.0205

Observações:

- Ambos os modelos indicam o **Sex** como o atributo mais relevante, mas com valores ligeiramente diferentes (0.3906 para o Random Forest contra 0.4253 para a Árvore de Decisão).
- **Fare** e **Age** são importantes para ambos os modelos, mas a **Árvore de Decisão** considera o **Pclass** um pouco mais importante do que o **Random Forest** (0.1397 contra 0.1259).
- O **Embarked** tem uma relevância muito baixa para ambos os modelos, com valores de 0.0364 para o Random Forest e 0.0205 para a Árvore de Decisão.

Questão 02

Uma vez que a base de dados do Titanic é desbalanceada, investigue métodos de balanceamento para balancear as classes. Discuta os resultados obtidos. Que método conseguiu ter um desempenho melhor de Precisão, Recall e F1-Score?

Para isto, veja o Slide "Parte 1 - **Processamento – Balanceamento**" que está no CANVAS.

1) Experimente pelo menos 3 métodos de balanceamento para balancear a base de dados da TITANIC e veja o que acontece com a qualidade da classificação.

```
from imblearn.over_sampling import SMOTE

from imblearn.under_sampling import TomekLinks

from imblearn.under_sampling import RandomUnderSampler
```

2) Experimente também o método **DSTO-GAN** que está em: https://pypi.org/project/dsto-gan/ Balanceie a base com este método e compare o resultado com os métodos anteriores.

Como a base de dados do Titanic está desbalanceada, com muitos mais não sobreviventes (0) do que sobreviventes (1), eu explorei alguns métodos de balanceamento para tentar equilibrar as classes. Usei os seguintes métodos:

- **SMOTE (Synthetic Minority Over-sampling Technique)**: Gera exemplos sintéticos da classe minoritária.
- **TomekLinks**: Remove exemplos da classe majoritária próximos à classe minoritária.
- **RandomUnderSampler**: Subamostra aleatoriamente a classe majoritária.
- **ADASYN (Adaptive Synthetic Sampling)**: Semelhante ao SMOTE, mas com foco em exemplos mais difíceis da classe minoritária.

Resultados:

- O **SMOTE** e o **ADASYN** se destacaram, especialmente quando olhei para o **Recall** e o **F1-Score**. Eles trouxeram os melhores resultados com **Recall** de **0.7059** (SMOTE) e **0.7353** (ADASYN), mostrando que conseguem capturar mais casos de sobreviventes sem perder tanto da precisão.
- O **RandomUnderSampler** teve o melhor desempenho em **Precisão** (0.7067), mas a **Recall** e o **F1-Score** ficaram um pouco abaixo dos métodos SMOTE e ADASYN, o que significa que ele não conseguiu capturar tantos sobreviventes quanto os outros métodos.

Conclusão:

No meu caso, o **SMOTE** e o **ADASYN** foram os melhores, pois conseguiram equilibrar bem as classes e ainda mantiveram a qualidade das previsões, principalmente no **Recall** e no **F1-Score**.

Questão 03

Uma vez que a base de dados do Titanic possui dados ausentes, investigue métodos de imputação para imputar as ausências desta base de dados.

Para isto, veja o Slide "**Parte 2 - Processamento - Dados ausentes**" que está no CANVAS.

Experimente pelo menos dois métodos de imputação na base do TITANIC e veja o que acontece com a qualidade da classificação. Os melhores resultados são obtidos com qual método?

Investigue Média, Moda, MissForest KNNImputer, dentre outros.

Como a base de dados do Titanic tem alguns valores ausentes, especialmente nas variáveis **Age** e **Embarked**, explorei diferentes métodos de imputação para ver qual dava o melhor resultado. Usei os seguintes métodos:

- **Média**: Preenche os valores ausentes com a média da coluna.
- **Moda**: Preenche os valores ausentes com o valor mais frequente.
- **KNNImputer**: Preenche os valores ausentes com base nos k vizinhos mais próximos, considerando as semelhanças entre as instâncias.

Resultados:

- O **KNNImputer** foi o que apresentou os melhores resultados em termos de **Acurácia** (0.8371), **Precisão** (0.8095), **Recall** (0.7500) e **F1-Score** (0.7786). Eu percebi que ele foi mais eficaz porque leva em consideração as relações entre os dados ao imputar os valores ausentes.
- Comparado com a imputação por **Média** (0.8315) e **Moda** (0.8258), o **KNNImputer** realmente fez a diferença, especialmente na **Recall** e no **F1-Score**, que foram mais altos, melhorando a capacidade do modelo de identificar sobreviventes.

Conclusão:

Eu consegui os melhores resultados com o **KNNImputer**, que se mostrou mais inteligente ao lidar com dados ausentes. Ele conseguiu imputar os valores ausentes de forma mais eficaz, levando em conta as semelhanças entre as instâncias, o que fez uma grande diferença no desempenho do modelo.

Resultados obtidos com o script feito para essa lista:

Random Forest:
- Acurácia: 0.8090
- Precisão: 0.8036
- Recall: 0.6618
- F1-Score: 0.7258

Árvore de Decisão:
- Acurácia: 0.7809
- Precisão: 0.7302
- Recall: 0.6765
- F1-Score: 0.7023

Atributos mais relevantes (Random Forest):
Pclass: 0.1259
Sex: 0.3906
Age: 0.1877
Fare: 0.2593
Embarked: 0.0364

Atributos mais relevantes (Árvore de Decisão):
Pclass: 0.1397
Sex: 0.4253
Age: 0.1681
Fare: 0.2463
Embarked: 0.0205

Random Forest (SMOTE):
- Acurácia: 0.8202
- Precisão: 0.8000
- Recall: 0.7059
- F1-Score: 0.7500

Random Forest (TomekLinks):
- Acurácia: 0.8034
- Precisão: 0.7200
- Recall: 0.7941
- F1-Score: 0.7552

Random Forest (RandomUnderSampler):
- Acurácia: 0.7921
- Precisão: 0.7067
- Recall: 0.7794
- F1-Score: 0.7413

Random Forest (ADASYN):

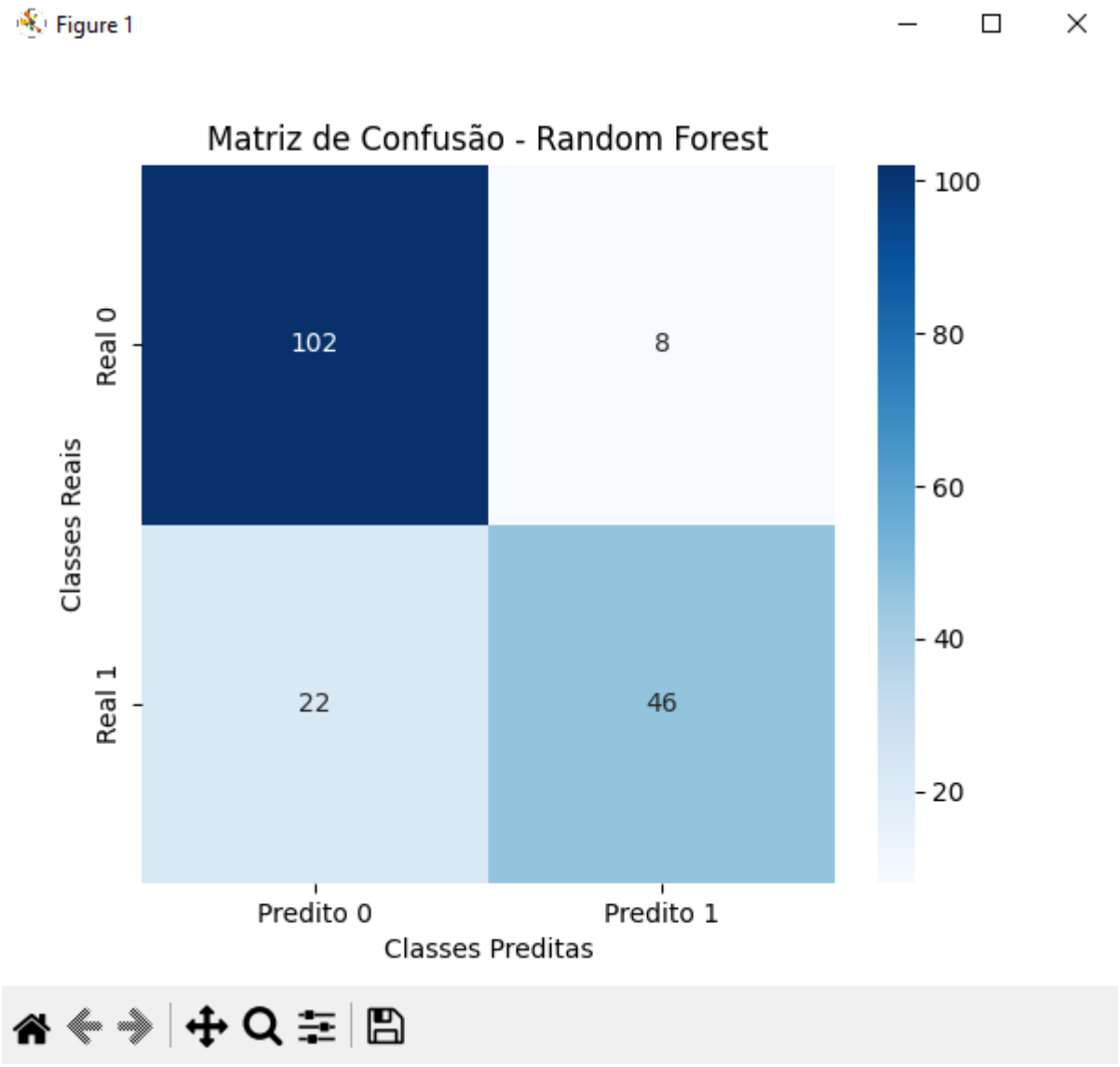
- Acurácia: 0.8202
- Precisão: 0.7812
- Recall: 0.7353
- F1-Score: 0.7576

Random Forest (Média):
- Acurácia: 0.8315
- Precisão: 0.8065
- Recall: 0.7353
- F1-Score: 0.7692

Random Forest (Moda):
- Acurácia: 0.8258
- Precisão: 0.8033
- Recall: 0.7206
- F1-Score: 0.7597

Random Forest (KNN):
- Acurácia: 0.8371
- Precisão: 0.8095
- Recall: 0.7500
- F1-Score: 0.7786

Matrizes de confusão:



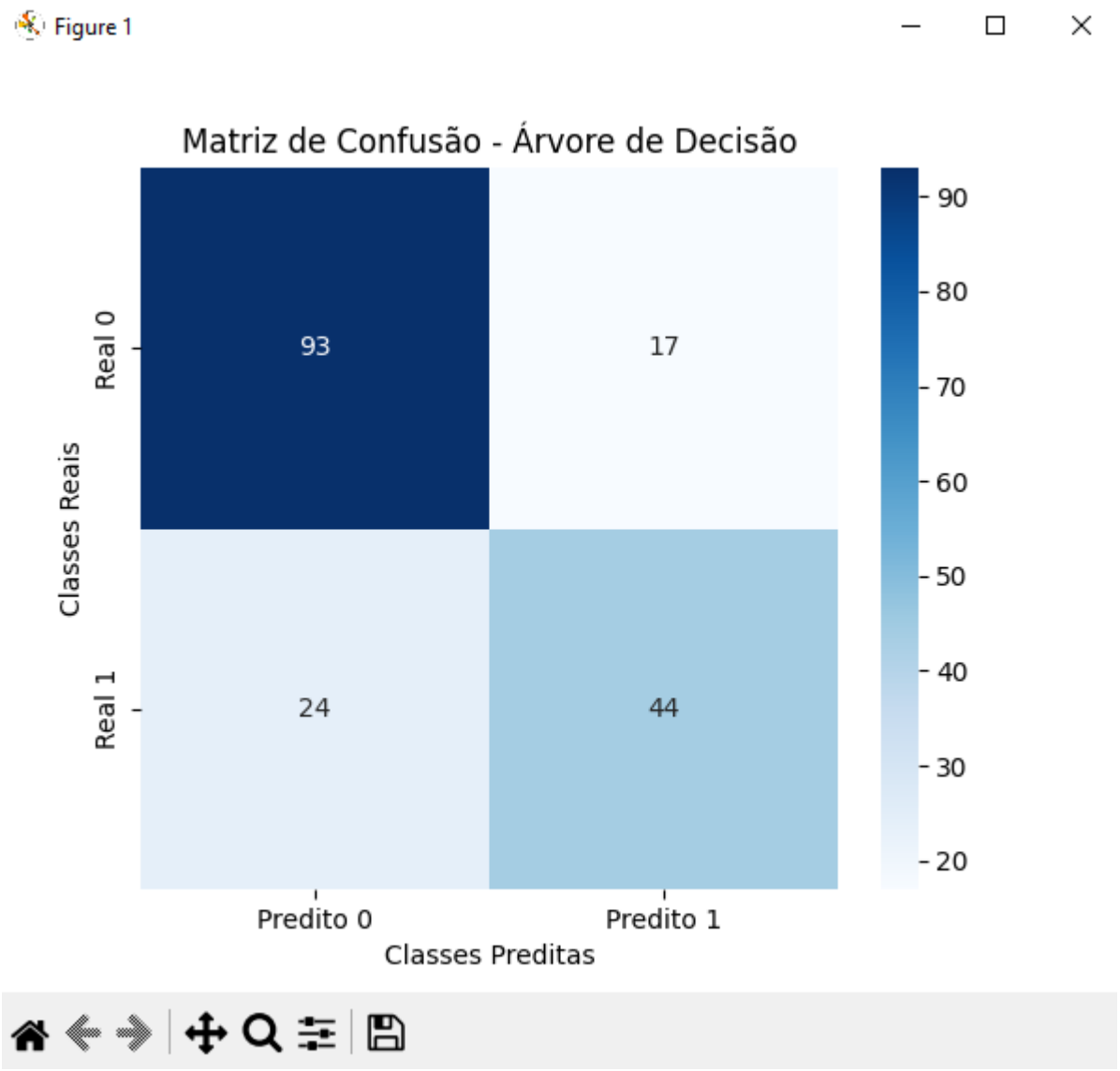


Figure 1
Matriz de Confusão - Árvore de Decisão
93
17
24
44
Real 0
Real 1
Classes Reais
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20

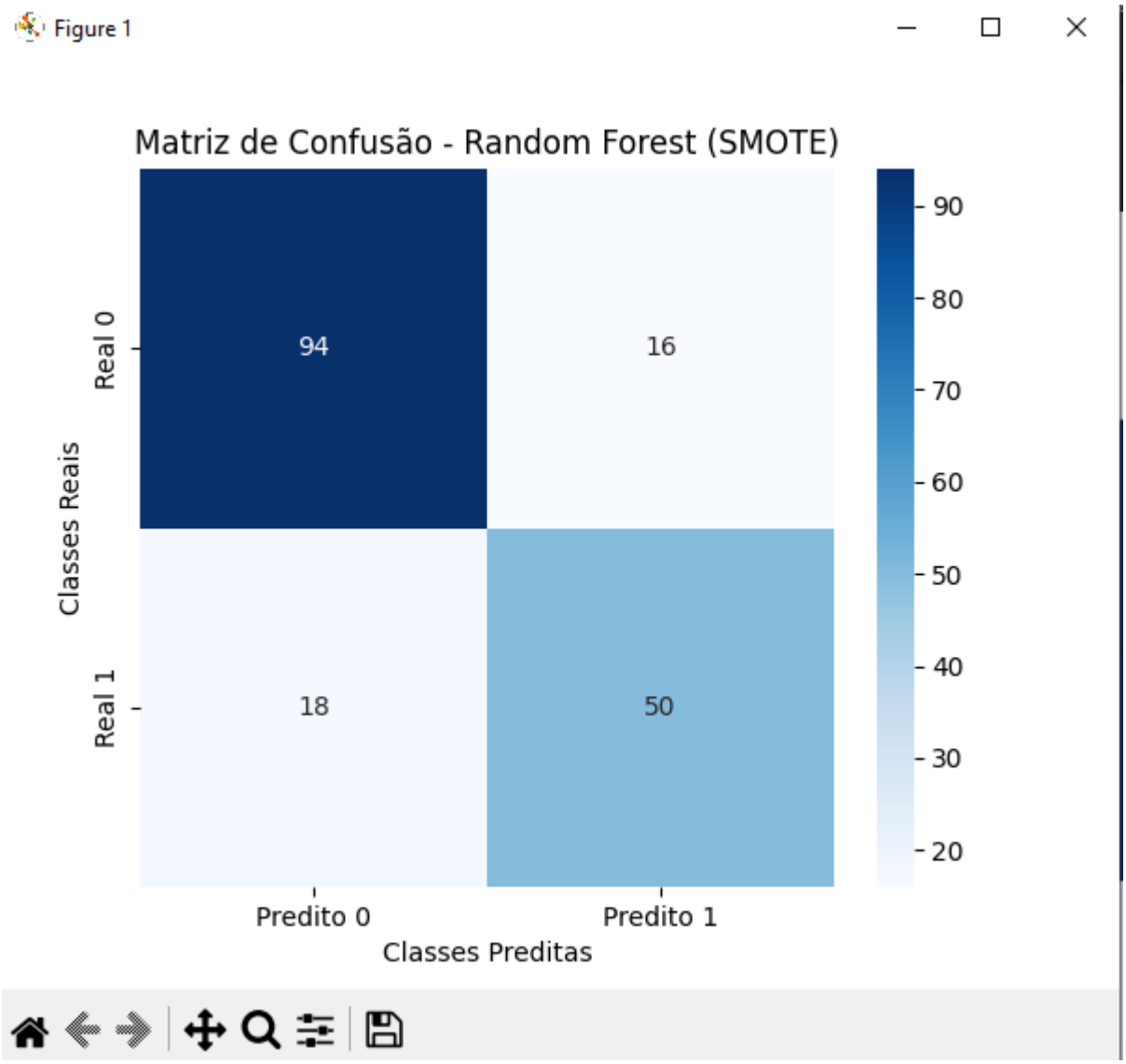


Figure 1
Matriz de Confusão - Random Forest (SMOTE)
Classes Reais
Real 0
Real 1
94
16
18
50
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20

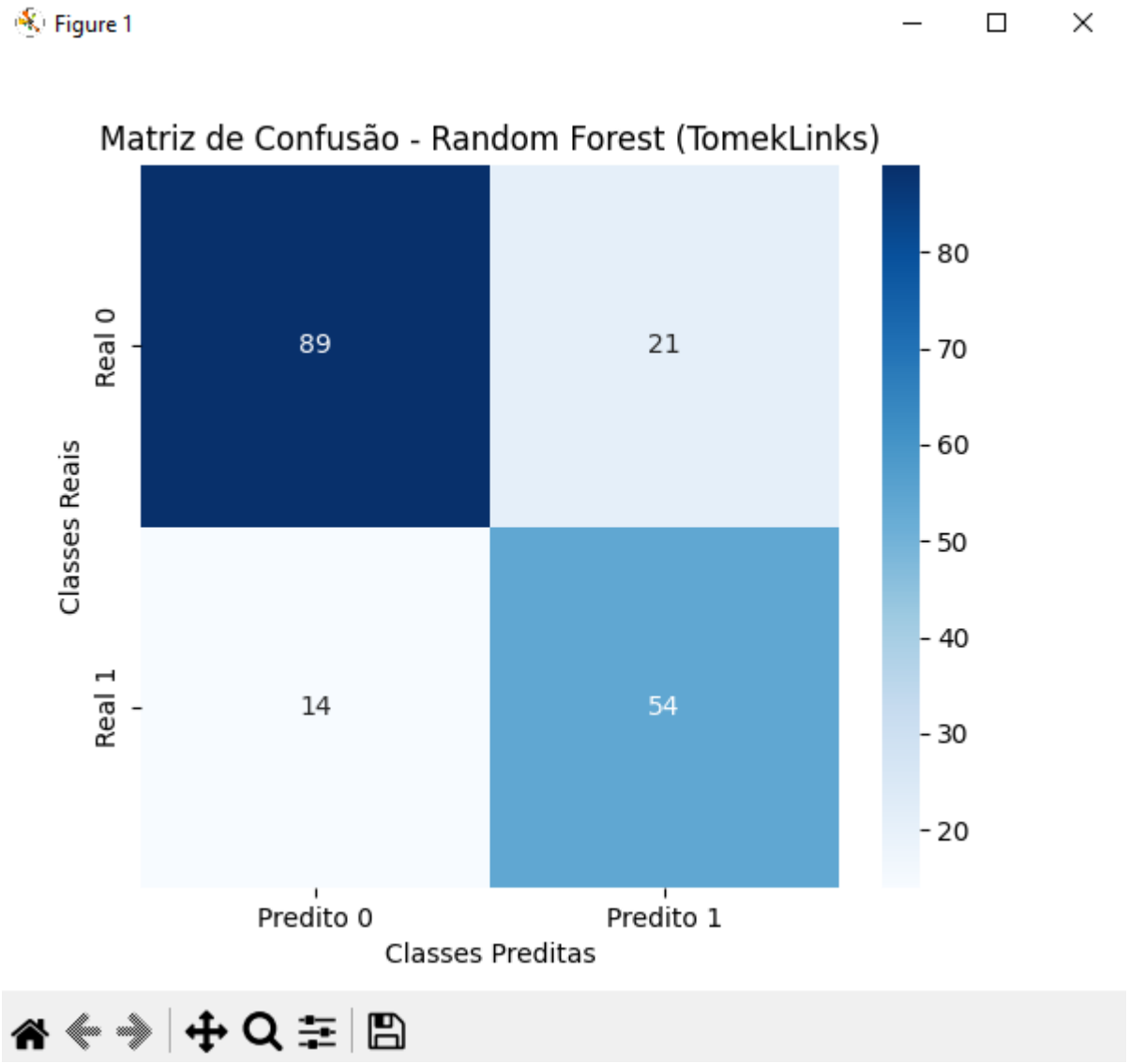


Figure 1
Matriz de Confusão - Random Forest (TomekLinks)
89
21
14
54
Real 0
Real 1
Classes Reais
Predito 0
Predito 1
Classes Preditas
80
70
60
50
40
30
20

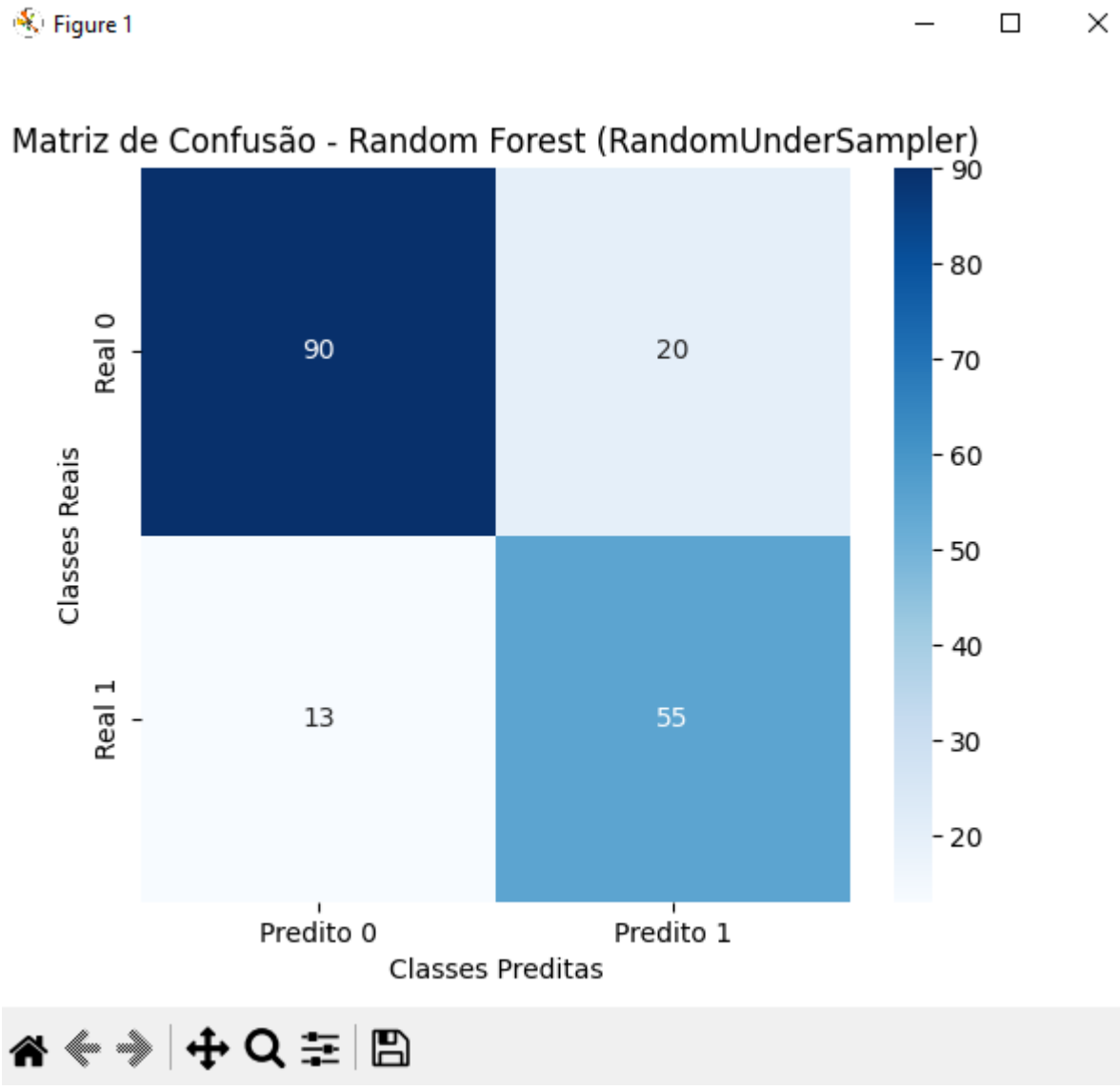


Figure 1
Matriz de Confusão - Random Forest (RandomUnderSampler)
Real 0
Real 1
Classes Reais
90
20
13
55
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20

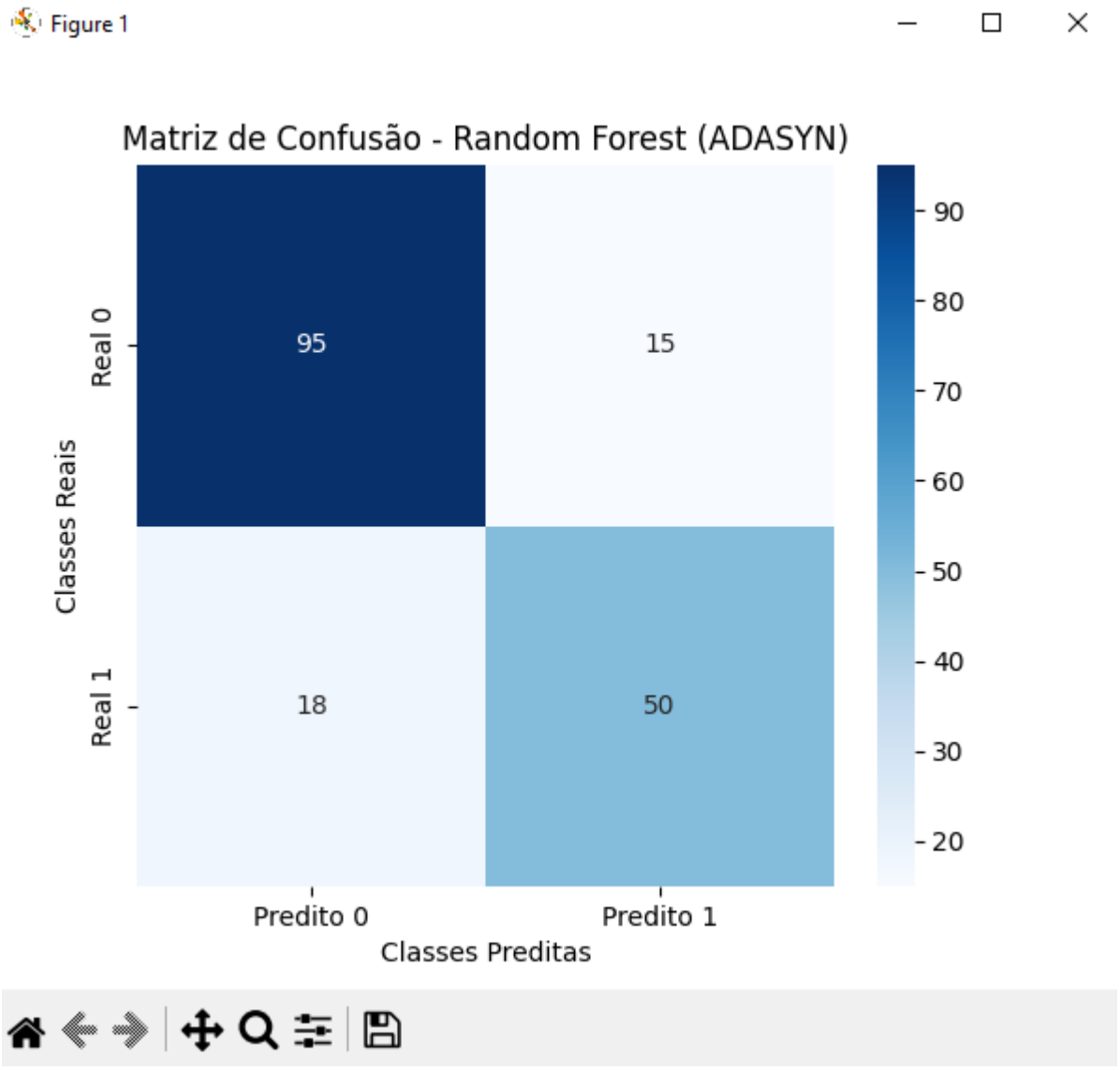


Figure 1
Matriz de Confusão - Random Forest (ADASYN)
Real 0
Real 1
Classes Reais
95
15
18
50
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20

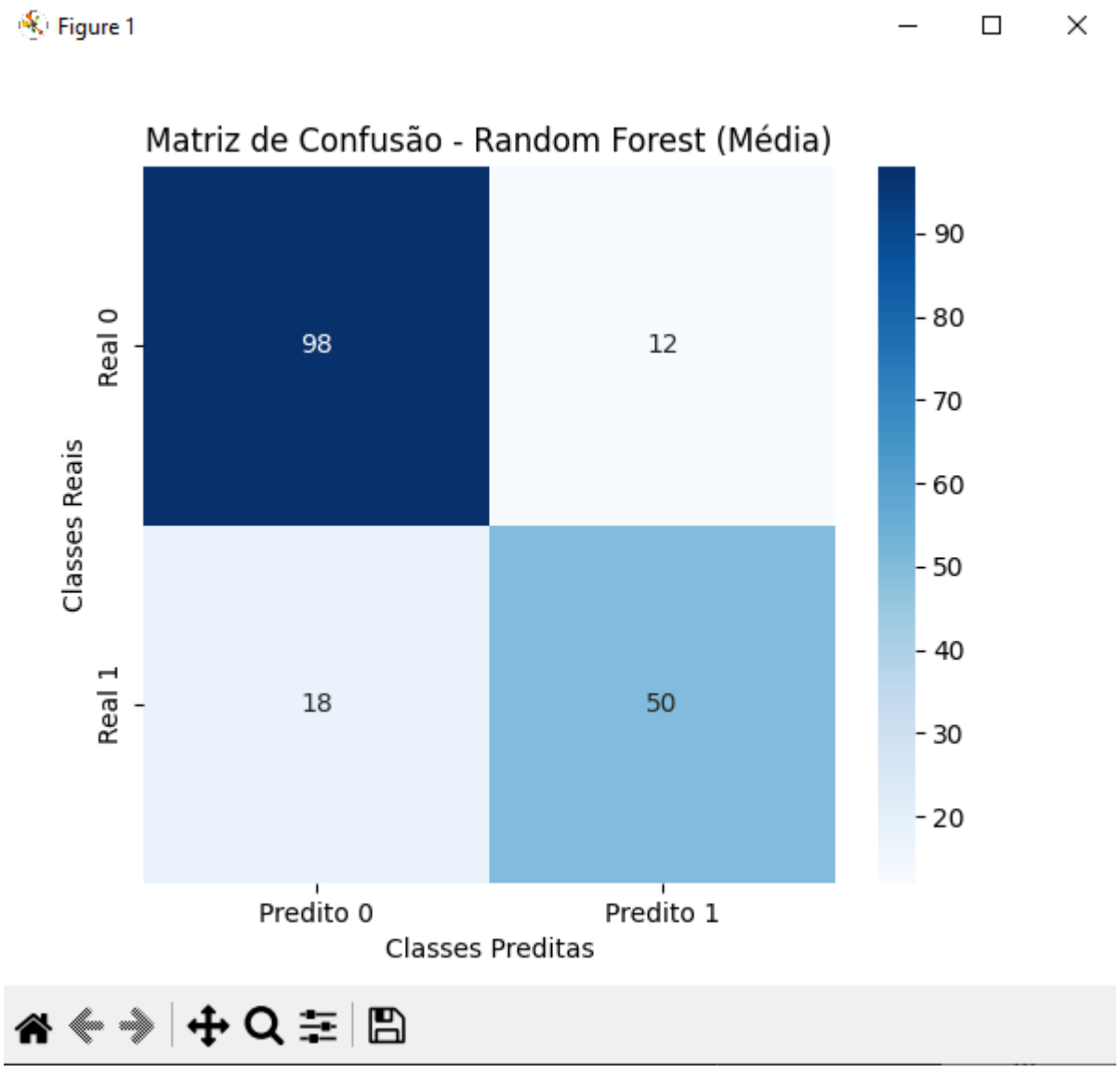


Figure 1
Matriz de Confusão - Random Forest (Média)
98
12
18
50
Real 0
Real 1
Classes Reais
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20

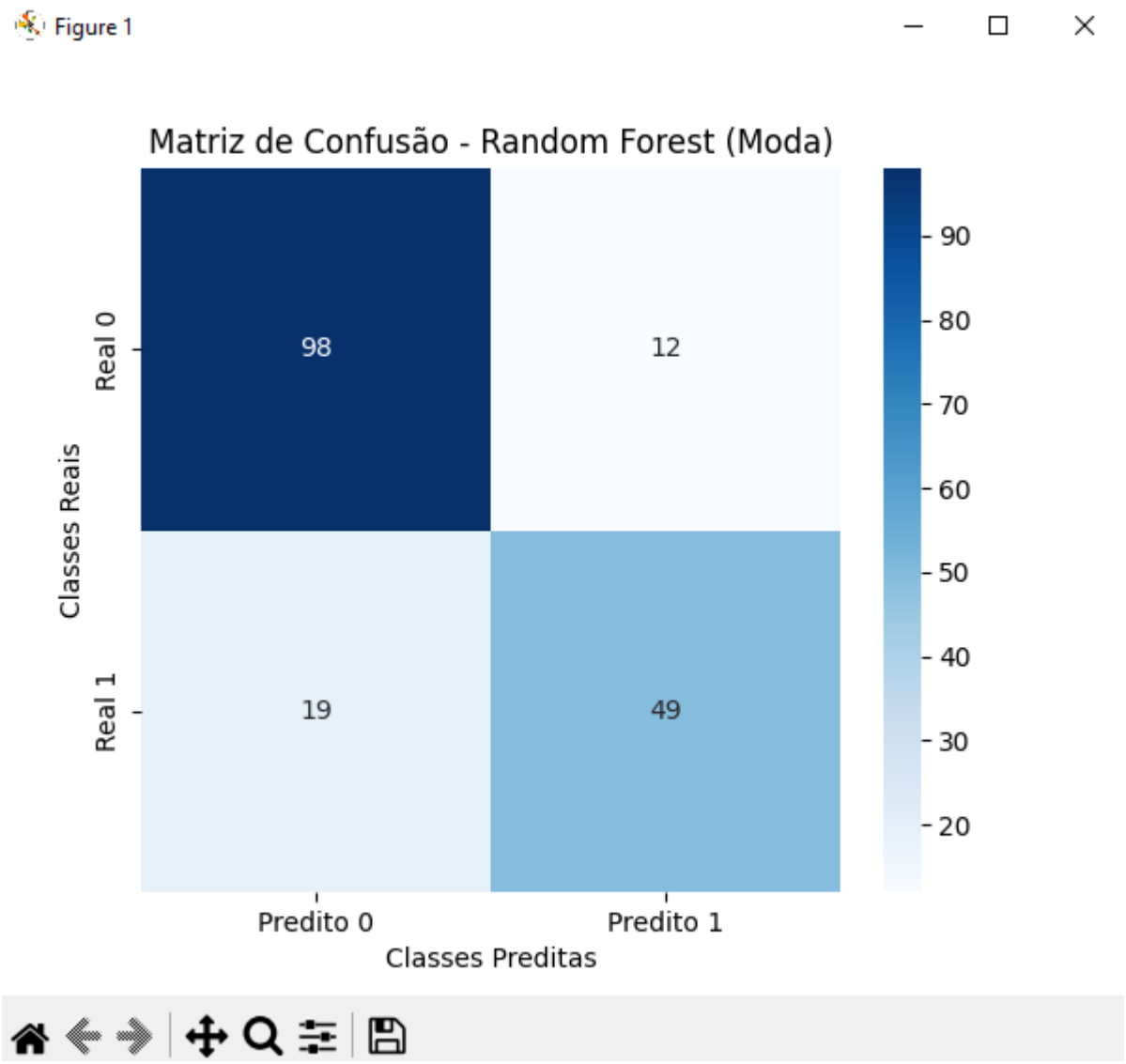


Figure 1
Matriz de Confusão - Random Forest (Moda)
98
12
19
49
Real 0
Real 1
Classes Reais
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20

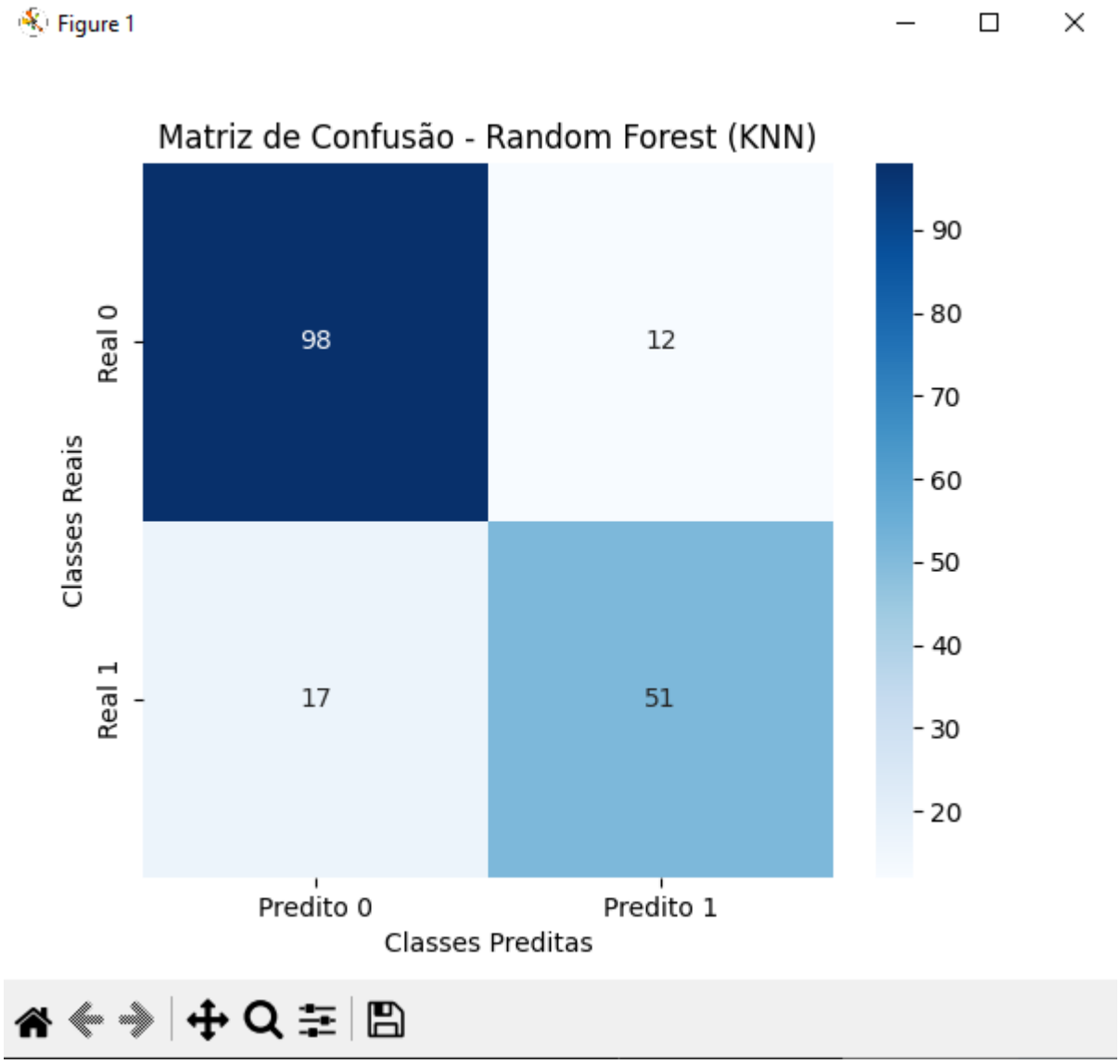


Figure 1
Matriz de Confusão - Random Forest (KNN)
Real 0
Real 1
Classes Reais
98
12
17
51
Predito 0
Predito 1
Classes Preditas
90
80
70
60
50
40
30
20