UNIVERSIDADE FEDERAL DA BAHIA

ISABELLA CALFA VIEIRA COSTA

ATIVIDADE INDIVIDUAL - APRENDIZADO DE MÁQUINA

Salvador
2021

ISABELLA CALFA VIEIRA COSTA

ATIVIDADE INDIVIDUAL - APRENDIZADO DE MÁQUINA

Salvador
2021

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

UCI	Universidade da California, em Irvine
VP	Verdadeiro Positivo
VN	Veradeiro Negativo
FP	Falso Positivo
FN	Falso Negativo
ACC	Acurácia
MCC	Coeficiente de Correlação de Matthews
R	Recall (Sensibilidade)
P	Precisão
F1	F1-Score

SUMÁRIO

1 INTRODUÇÃO

O câncer de mama é o tipo de câncer que mais atinge mulheres ao redor do mundo - em 2018 estimou-se crescimento no ano de 2,1 milhões de casos e de 627 mil óbitos em todo planeta - sendo reconhecidamente um relevante problema de saúde pública. O diagnóstico e tratamento dessa doença vem evoluindo ao longo dos anos: no Brasil, por exemplo, a estimativa de sobrevida em 5 anos passou de 76,9% entre 2005 e 2009 para 75,2% no período entre 2010 e 2014. (Instituto Nacional de Câncer José Alencar Gomes da Silva, 2019)

A estimativa ou taxa de sobrevida é uma medida utilizada na medicina para discutir o prognóstico dos pacientes. Ela não prevê o tempo que a pessoa viverá, mas indica a probabilidade de sucesso de tratamento após determinado tempo e é estabelecida a partir de resultados históricos de uma grande amostra da população que teve a doença. Dados de pacientes diagnosticados entre 2009 e 2015 indicam, por exemplo, que a taxa de sobrevida em 5 anos de pessoas com câncer de mama localizado (sem disseminação da doença) é de 99%, já do regional (tumor disseminado para estruturas próximas ou linfonodos) é de 86%. (Equipe Oncoguia, 2020)

O objetivo desse trabalho é definir qual é o melhor modelo de classificação para prever a sobrevivência ou não de um paciente de câncer de mama após 5 anos a partir dos dados de um estudo realizado entre 1958 e 1970 no Hospital Billings da Universidade de Chicago, que avaliou a sobrevivência após 5 anos de 306 pacientes que se submeteram a cirurgia de câncer de mama. (Tjen-Sien Lim, 1999)

2 MATERIAL E MÉTODOS

2.1 Linguagem Utilizada

O trabalho foi desenvolvido em linguagem Python com utilização de pacotes de Aprendizado de Máquina e Visualização de Gráficos. Os códigos estão disponíveis em A.

2.2 O conjunto de Dados: Haberman.data

O conjunto de dados utilizado para os modelos de predição foi o "Haberman's Survival Data Set"disponível no Repositório de Machine Learning da UCI (Tjen-Sien Lim, 1999). Nesse conjunto estão disponíveis dados de 306 pacientes submetidos à cirurgia de câncer de mama:

- **Idade:** Idade do paciente no momento da cirurgia. 306 valores inteiros e não nulos com variação entre 30 e 83 anos.
- **Ano:** Ano da cirurgia. 306 valores inteiros e não nulos com variação entre 1958 e 1969.
- **Nódulos:** Número de nódulos positivos detectados. 306 valores inteiros e não nulos com variação entre 0 e 52 nódulos.
- **Status de sobrevivência:** Sobrevivência ou não após 5 anos. 306 valores inteiros e não nulos com variação entre 30 e 83 anos. O valor 1 indica que sobreviveu 5 anos ou mais e o valor 2 significa que o paciente morreu antes de 5 anos.

A análise dos dados de idade e ano indica que não existem outliers para essas variáveis. A média de idade dos pacientes do estudo foi de 52,4 anos e mediana de 52 anos, com distribuição próxima do normal, como visto na Figura 1. Realizando o teste de Shapiro encontra-se um p-valor de 0,026047, ou seja, a distribuição não é normal para 5% de significância. Para o ano encontra-se que a distribuição também não é normal para 5% de significância e há menor amplitude entre os valores analisados, como esperado.

A variável nódulos tem uma grande quantidade de outliers e também não apresenta distribuição normal. Os outliers tem grande impacto sobre a média de nódulos dos pacientes, que ficou em 4,03, apesar de apenas 25% dos pacientes registrarem 4 ou mais nódulos.

No que diz respeito a sobrevivência por 5 anos ou mais, ou seja, a taxa de sobre-

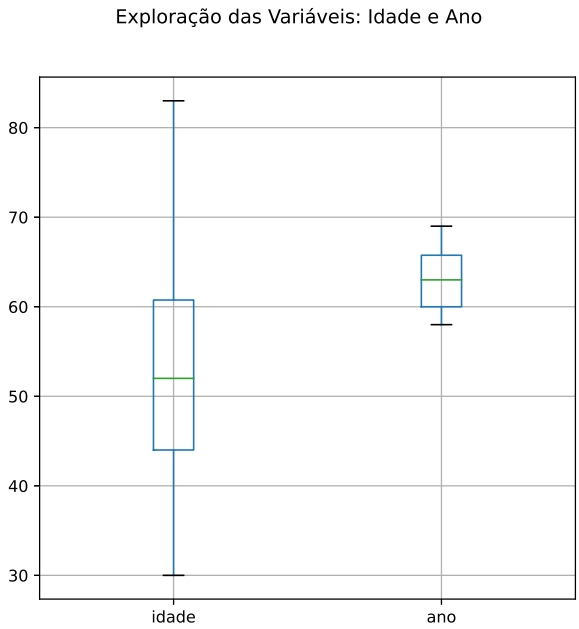

Figura 1 – Análise das Variáveis: Idade e Ano

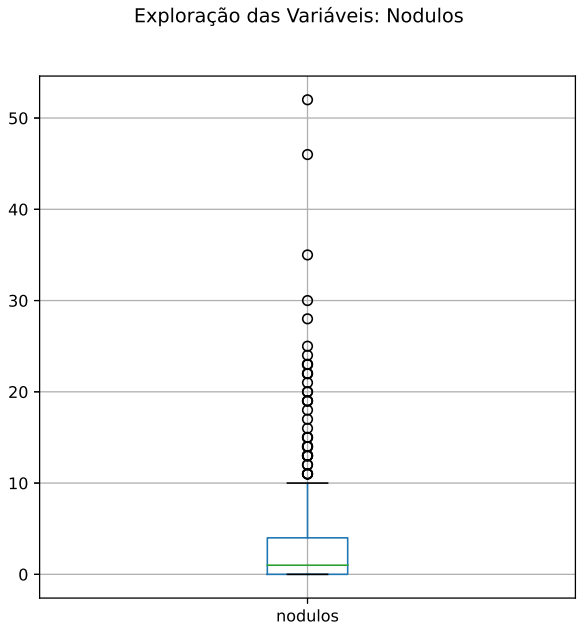

Figura 2 – Análise das Variáveis: Nódulos

vida, observa-se que em geral os pacientes do estudo sobreviveram por mais de 5 anos: a média da variável foi de 1,26, logo, mais próxima de 1 (sobrevida 5 anos ou mais).

2.3 Modelos de Predição

2.3.1 Regressão Logística

A regressão logística objetiva produzir um modelo para prever valores que serão assumidos por uma variável categórica (normalmente binária) a partir de uma ou mais variáveis independentes contínuas contidas em um conjunto de observações. Esse modelo permite modelar a probabilidade de um evento ocorrer a depender dos valores das variáveis independentes de análise, estimar a probabilidade do evento selecionado, prever o efeito do conjunto de variáveis sobre a variável dependente e classificar as observações (a partir da probabilidade de estar em uma ou outra categoria). (GONZALEZ, 2018) Esse modelo tem alto grau de confiabilidade e requer pequeno número de suposições e pode ser utilizado para prever o risco de uma pessoa desenvolver uma doença com base nas características, risco de fraude bancária, análise de crédito, análise ambiental entre outros.

Esse modelo pode ser implementado no Python através do pacote Sklearn, função LogisticRegression(). A melhoria do resultado previsto pelo modelo pode ser feito através do próprio pacote através da atualização de parâmetros como:

- **Penalidade (penalty):** Parâmetro utilizado para reduzir a quantidade de variáveis de um modelo, buscando atingir um modelo mais simples, mas com bons resultados. Pode ser Lasso 1 (L1), que zera os coeficiente menos importantes (seleção de atributos) e Lasso 2, que reduz os coeficientes com altos valores para que tendam a zero causando a suavização do modelo. (Gustavo Santos, 2021)
- **Regularização (C):** Parâmetro usado para definir a força da regularização: C grande pode gerar overfit e pequeno pode gerar underfit. (Gabriel Stankevix, 2020)

2.3.2 Análise Discriminante

A análise discriminante é uma técnica que compara as diferenças entre os grupos e classifica o objeto de análise no grupo que tenha características mais próximas as dele, isto é, procura-se classificar a variável de interesse para pertencer no grupo que minimize o custo da classificação incorreta. As variáveis utilizadas, assim como na Regressão Logística, são categóricas, mas podem ter duas ou mais categorias. Utiliza-se esse tipo de

modelo para explorar as diferenças entre grupos de observações previamente definidos. (Admir Antonio Betarelli Junior,)

Esse modelo pode ser implementado no Python através do pacote Sklearn, função LinearDiscriminantAnalysis().

2.3.3 KNN

O método do KNN (K Vizinhos mais próximos) utiliza a distância entre os dados de análise para classificar a variável de interesse: quanto menor for a distância entre as variáveis, maior o grau de similaridade. (SILVA, 2017) O cálculo da distância entre as instâncias usualmente é realizado através da distância euclidiana, mas visando a melhoria dos resultados de predição do modelo, outras métricas como Manhattan podem ser utilizadas. O valor K dos vizinhos também pode ser otimizado ao longo do modelo. A implementação desse modelo em Python pode ser feita através do pacote Sklearn, função KNeighborsClassifier().

2.3.4 Árvore de Decisão

A Árvore de Decisão também utiliza do aprendizado supervisionado para classificar e prever os dados e utiliza de uma árvore com ramificações variáveis de acordo com a quantidade de atributos e seus valores. Essa técnica busca identificar os atributos que fornecem a maioria das informações, removendo os raros, para melhorar o modelo. (SILVA, 2017) A estratégia utilizada por esses algoritmos é de fragmentar um problema complexo em problemas menores, que serão divididos em problemas menores ainda, e assim sucessivamente. A árvore formada tem um nó de decisão que contém um teste de cada atributo, conjuntos de ramos distintos e folhas ligadas a classes, que são ligadas com as raízes, que de fato tem a regra de classificação. (MARTINS, 2021) A implementação desse modelo em Python pode ser feita através do pacote Sklearn, função DecisionTreeClassifier().

2.3.5 Naive Bayes

O algoritmo de Naive Bayes tem como premissa que cada variável influenciará de forma independente na classificação da variável de interesse. Durante o treinamento

do modelo, visto que é de aprendizado supervisionado, cada variável de análise recebe um peso em cada uma das classes, esses pesos serão somados e a classe com maior peso será a que classificará o novo objeto. (SILVA, 2017) A implementação desse modelo em Python pode ser feita através do pacote Sklearn, função GaussianNB().

2.3.6 Florestas Aleatórias

O modelo de Florestas Aleatórias utiliza-se da técnica Árvore de Decisão, já apresentada nesse documento. Essa técnica cria várias árvores de decisão a partir de subconjuntos de atributos aleatoriamente selecionados e em seguida efetua a classificação da variável de interesse de acordo com a árvore que possui a melhor lógica e vantagens para tomar a decisão. Cada subconjunto criado tem um peso e quanto mais precisa for a árvore, maior a contribuição para a definição da variável de interesse. (MARTINS, 2021) A implementação desse modelo em Python pode ser feita através do pacote Sklearn, função RandomForestClassifier().

2.3.7 KFold

A técnica de Validação Cruzada KFold consiste em subdividir o conjunto de interesse em subconjuntos de treinamento e de teste para avaliar os parâmetros ou o modelo que apresenta os melhores desempenhos.

2.4 Métricas de Avaliação

2.4.1 Matriz de Confusão

Uma matriz de confusão é uma tabela de comparação entre os valores reais do conjunto de teste e os valores preditos por um modelo. Através dela é possível identificar:

- **Verdadeiros Positivos (VP):** Valores que foram preditos como positivos e de fato o são no conjunto de teste.
- **Verdadeiros Negativos (VN):** Valores que foram preditos como negativos e de fato o são no conjunto de teste.

- **Falsos Positivos (FP):** Valores que foram preditos como positivos, porém são negativos no conjunto de teste.
- **Falsos Negativos (FN):** Valores que foram preditos como negativos, porém são positivos no conjunto de teste.

2.4.2 Acurácia (ACC)

Essa métrica de avaliação é calculada a partir da Matriz de Confusão e é a proporção de acertos do modelo, ou seja:

$$ACC = \frac{VP + VN}{VP + VN + FP + FN}$$

Essa é a medida mais utilizada na comparação entre os modelos de classificação.

2.4.3 Coeficiente de Correlação de Matthews (MCC)

Medida utilizada para interpretar a aleatoriedade e a classificação do modelo:

- **MCC próximo de 1:** Classificação perfeita.
- **MCC próximo de -1:** Classificação inversa.
- **MCC próximo de 0:** Classificação aleatória.

O cálculo do MCC também é realizado a partir da Matriz de Confusão:

$$MCC = \frac{VPxVN - FPxFN}{\sqrt{(VP + FP)(VP + FN)(VN + FP)(VN + FN)}}$$

2.4.4 Recall (R)

Essa medida indica a proporção de real positivo sobre o positivo.

$$R = \frac{VP}{VP + FN}$$

2.4.5 Precisão (P)

A precisão é a medida que indica a proporção de previstos positivos reais sobre os previstos postivos.

$$P = \frac{VP}{VP + FP}$$

2.4.6 F1-Score (F1)

É o trade-off entre o P e o R e é usado para avaliar o desempenho geral do classificador.

$$F1 = \frac{2xPxR}{P + R}$$

3 RESULTADOS

3.1 Preparativos

O conjunto de dados foi dividido em treinamento e teste na proporção 70% e 30% de forma aleatória. O conjunto de treinamento, então, ficou com 214 medições e o de testes com 92.

3.2 Comparativo Inicial dos Modelos

No primeiro momento utilizou-se o conjunto de treinamento para avaliar qual dos modelos resultaria em melhor desempenho: Regressão Logística, Análise Discriminante, KNN, Árvore de Decisão, Naive Bayes ou Florestas Aleatórias. A métrica selecionada para análise foi a Acurácia. Utilizando-se o KFold com 30 divisões, identificou-se que o modelo que geraria melhor acurácia seria o Naive Bayes, com 79,03%, como visto na Tabela 1. O pior modelo seria obtido com a Árvore de Decisão (68,17% de acurácia).

Comparando-se o resultado de eficácia obtido pelo processo KFold em todos os modelos, Figura 3, observa-se que Naive Bayes apresenta realmente as melhores acurácias e que todos os modelos, exceto Florestas Aleatórias, apresentam alguma medição com outlier. Pelos valores obtidos, espera-se que a acurácia de Regressão Logística, KNN e Análise Discriminante varie entre 70% e 75% e o da Árvore de Decisão não ultrapasse 70%. O modelo gerado por Florestas Aleatórias apresenta resultado semelhante ao da Árvore de Decisão na mediana da acurácia, apesar da distribuição com assimetria diferente.

Modelo	Acurácia
Regressão Logística	77,46%
KNN	76,55%
Árvore de Decisão	68,17%
Naive Bayes	79,03%
Florestas Aleatórias	73,05%
Análise Discriminante	78,56%

Tabela 1 – Comparação inicial dos modelos: treinamento

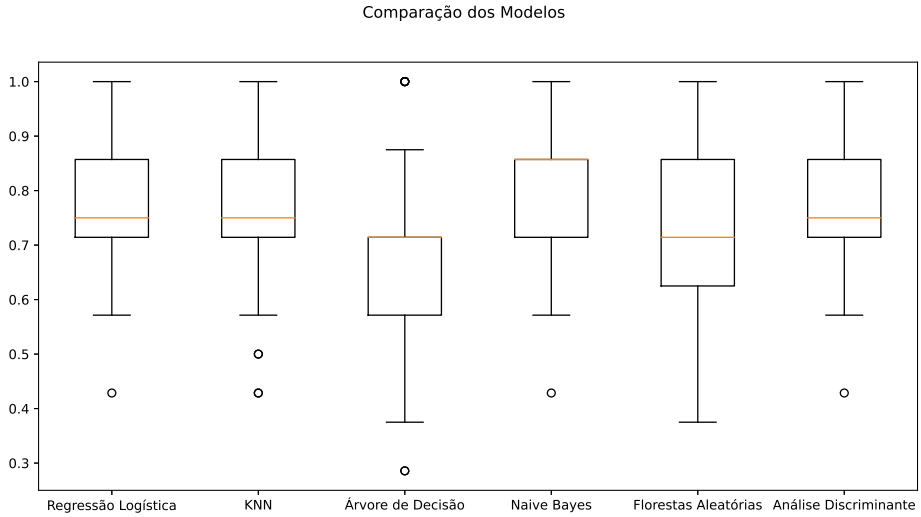

Figura 3 – Comparação inicial da acurácia dos modelos

3.3 Modelo Regressão Logística

O primeiro modelo testado no conjuntos de testes foi a Regressão Logística, obtendo-se acurácia de 63,04%. Buscando-se obter um melhor resultado, visto que no modelo de treinamento calculou-se acurácia de 77,46%, foi aplicada a técnica KFold para encontrar os melhores parâmetros de Regularização (penalty e C) e o melhor modelo de cálculo. Ao final, o modelo obteve acurácia de 64,13%.

A Matriz de Confusão do primeiro modelo e do melhor modelo obtido mostra que o modelo tem a tendência de prever resultados 1, ou seja, que o paciente viverá mais de 5 anos, conforme Figura 4.

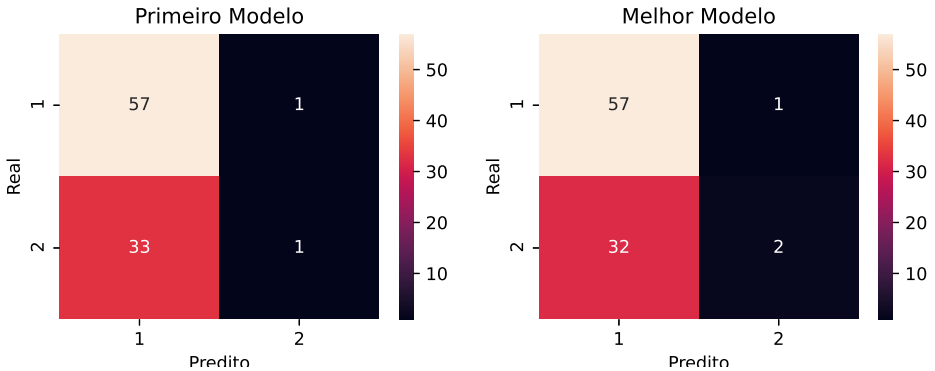

Figura 4 – Comparação da Matriz de Confusão dos modelos de Regressão Logística

O MCC do primeiro modelo estava em 4%, ou seja, era uma definição quase aleatória. Com o melhor modelo, o MCC passou para 11,30%, vide Tabela 3, indicando

menor aleatoriedade e mais próxima da classificação perfeita. As medidas de R, P e, consequentemente, F1 pouco sofreram alterações, como esperado pelas Matrizes de Confusão.

Métrica	Valor
ACC	64,13%
MCC	11,30%
R	98,27%
P	64,04%
F1	77,55%

Tabela 2 – Métricas Regressão Logística

3.4 Modelo Análise Discriminante

O método da Análise Discriminante foi aplicado em sequência gerando acurácia também de 64,13%. A matriz de confusão gerada, Figura 5, também foi bastante semelhante ao modelo anterior.

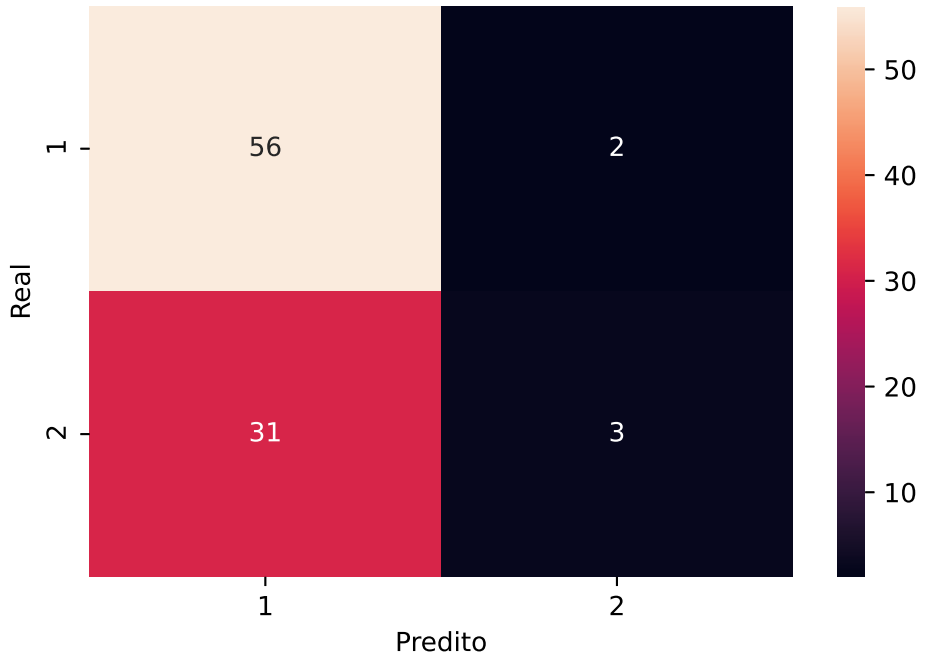

Figura 5 – Matriz de Confusão da Análise Discriminante

Poucos ganhos foram obtidos em relação ao modelo de Regressão Logística, porém o R apurado na Análise Discriminante foi menor. Apesar dos outros indicadores

serem levemente superiores (exceto a acurácia), a maior discrepância do Recall indica que seria melhor utilizar a Regressão Logística, para evitar um diagnóstico para o paciente de que ele teria mais chances de sobreviver por mais de 5 anos, sendo que não seria verdade, gerando falsos expectativas para ele e para a família.

Métrica	Valor
ACC	64,13%
MCC	11,44%
R	96,55%
P	64,34%
F1	77,24%

Tabela 3 – Métricas Regressão Logística

3.5 Modelo KNN

O terceiro modelo testado no conjuntos de testes foi o KNN que obteve acurácia de 65,22%, maior do que os dois modelos anteriores e dentro do esperado pela análise inicial. No primeiro modelo montado, utilizou-se K = 5 e distância euclidiana. Buscando-se obter um melhor resultado, visto que no modelo de treinamento calculou-se acurácia de 76,55%, foi aplicada a técnica KFold para encontrar os melhores parâmetros de K e da forma de cálculo das distâncias: buscou-se valores de K entre 1 e 21, considernado apenas os ímpares, e calculando-se com as distâncias Euclidiana, de Manhattan e Minkowski. Ao final, o melhor modelo obtido foi com K = 21 e a distância euclidiana, obtendo acurácia de 70,65%.

Novamente, a Matriz de Confusão do primeiro modelo e do melhor modelo obtido mostra que o modelo tem a tendência de prever resultados 1, porém passa a acertar mais quando o paciente viverá até 5 anos, conforme Figura 6. O melhor modelo, em especial, erra pouco ao dizer ao paciente que ele viveria menos de 5 anos e ele viveria mais, o que é um bom indicativo, pois o paciente poderia fazer planos de longo prazo com maiores certezas.

Todas as métricas de desempenho avaliadas para o KNN foram as maiores até então, como visto na Tabela 4, chegando ao MCC de 35,43%, ou seja, cada vez mais próximo do real e mais distante da aleatoriedade. Apesar da manutenção do Recall em relação ao modelo anterior, o F1 cresce com o aumento do P, que indica maior acerto dos positivos.

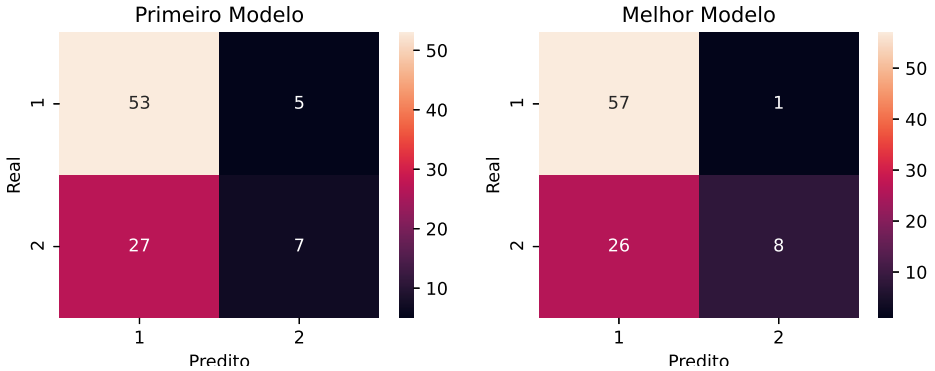

Figura 6 – Comparação da Matriz de Confusão dos modelos de KNN

Métrica	Valor
ACC	70,65%
MCC	35,43%
R	98,27%
P	68,67%
F1	80,85%

Tabela 4 – Métricas do KNN

3.6 Modelo Árvore de Decisão

A Árvore de Decisão foi aplicada em sequência no conjunto de dados para predição obtendo-se acurácia de 60,89%, a menor até então e bem aquém do esperado pelo conjunto de treinamento. Foram aplicados, então, no KFold alguns ajustes de parâmetros visando identificar o melhor modelo, chegando-se até acurácia de 63,04%, a menor apurada entre os modelos. A árvore que gerou o melhor modelo encontra-se na Figura 7.

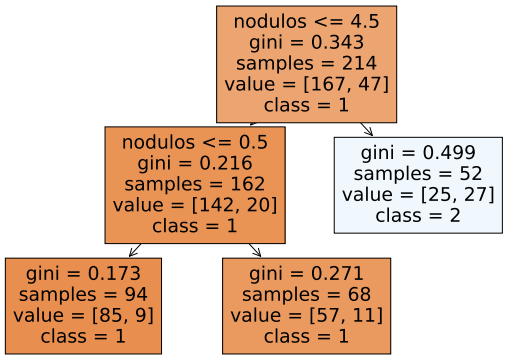

Figura 7 – Árvore de Decisão do Melhor Modelo

A Matriz de Confusão dos modelos, Figura 8, mostra que esse modelo indica

mais predições 2 (sobrevida inferior a 5 anos) do que os demais: tanto para quando é real, quanto para quando não é.

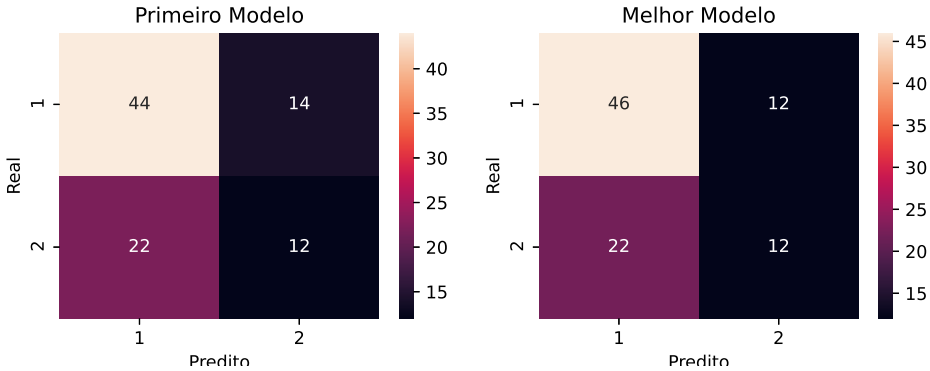

Figura 8 – Comparação da Matriz de Confusão dos modelos de KNN

Ao extrair as métricas de avaliação da matriz, percebe-se que a acurácia do modelo é inferior ao KNN, assim como as demais métricas Tabela 5. Destaca-se negativamente o baixo Recall em comparação com os outros modelos apresentados até aqui, praticamente descartando o modelo, pois aumenta as chances de sinalização equivocada de tempo de vida para o paciente.

Métrica	Valor
ACC	63,04%
MCC	16,05%
R	78,31%
P	67,64%
F1	73,01%

Tabela 5 – Métricas da Árvore de Decisão

3.7 Modelo Naive Bayes

O modelo de Naive Bayes foi o que gerou melhor eficácia pela análise do KFold no conjunto de treinamento, porém tal resultado não se comprovou no conjunto de testes nem no primeiro modelo calculado, nem no melhor modelo encontrado pelas análises de parâmetros. Tal situação pode ter sido gerada pela divisão dos dados dos conjuntos de treinamento e de teste. No melhor modelo ajustado a acurácia não ultrapassou 65,22%, quase 14 pontos percentuais abaixo do esperado.

A Matriz de Confusão do primeiro modelo e do melhor modelo retoma a tendência apresentada de predizer preferencialmente valores 1, vide Figura 4.

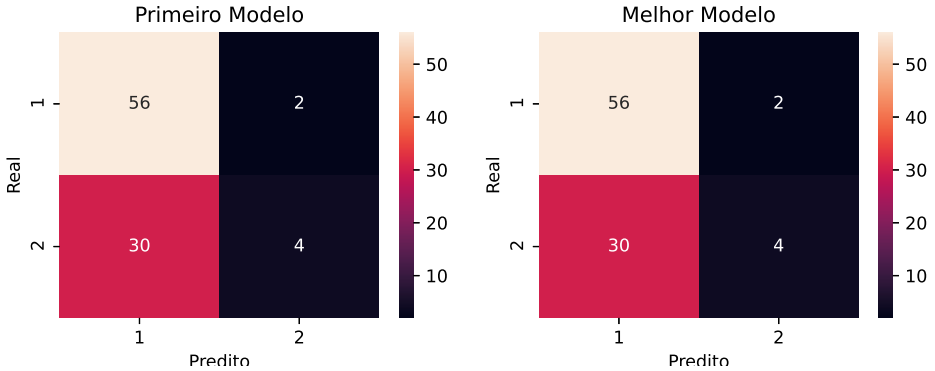

Figura 9 – Comparação da Matriz de Confusão dos modelos de Naive Bayes

Não só a acurácia, mas nenhuma das outras métricas de avaliação superou o apurado no KNN: o MCC indicou mais aleatoriedade e R e P indicaram maiores erros na predição de verdadeiros positivos, como mostrado na Tabela 7

Métrica	Valor
ACC	65,22%
MCC	16,26%
R	96,55%
P	65,12%
F1	77,77%

Tabela 6 – Métricas de Naive Bayes

3.8 Modelo Florestas Aleatórias

O modelo de Florestas Aleatórias foi o que mais demorou para gerar as predições (mais 5x mais do que os outros) e a acurácia também não superou o KNN. Esses fatores fizeram com que o modelo fosse descartado para esse conjunto de dados. Esse modelo apresentou bastante erro na predição dos positivos verdadeiros na comparação com os demais.

Métrica	Valor
ACC	66,30%
MCC	21,56%
R	87,93%
P	68,00%
F1	76,69%

Tabela 7 – Métricas de Florestas Aleatórias

3.9 Comparativo Final dos Modelos

Elaborados todos os modelos, busca-se avaliar o melhor modelo para o conjunto de dados em questão e, para isso, foram comparadas as medidas de avaliações de todos eles. Para cada medida foi identificado o modelo que gerou o melhor resultado, conforme Tabela 8.

Métrica	Modelo
ACC	KNN
MCC	KNN
R	KNN / Regressão Logística
P	KNN
F1	KNN

Tabela 8 – Comparativo Final das métricas dos modelos

O KNN foi o modelo que apresentou melhor acurácia, menor aleatoriedade (MCC) e melhores predições de verdadeiros positivos, portanto é o modelo mais adequado para esse conjunto de dados. É preciso, nesse problema, entender que por se tratar de um assunto sensível e crítico - sobrevivência do paciente 5 anos ou mais após a cirurgia de câncer -, erros são menos toleráveis, pois podem levar ao paciente a tomada de decisões ou atitudes críticas pensando ter mais ou menos tempo de vida.

Avaliando-se a curva ROC dos modelos previstos, Figura 10, comprova-se que o KNN gera o melhor resultado entre os métodos previstos para o conjunto de dados, pois gera a maior área abaixo da curva.

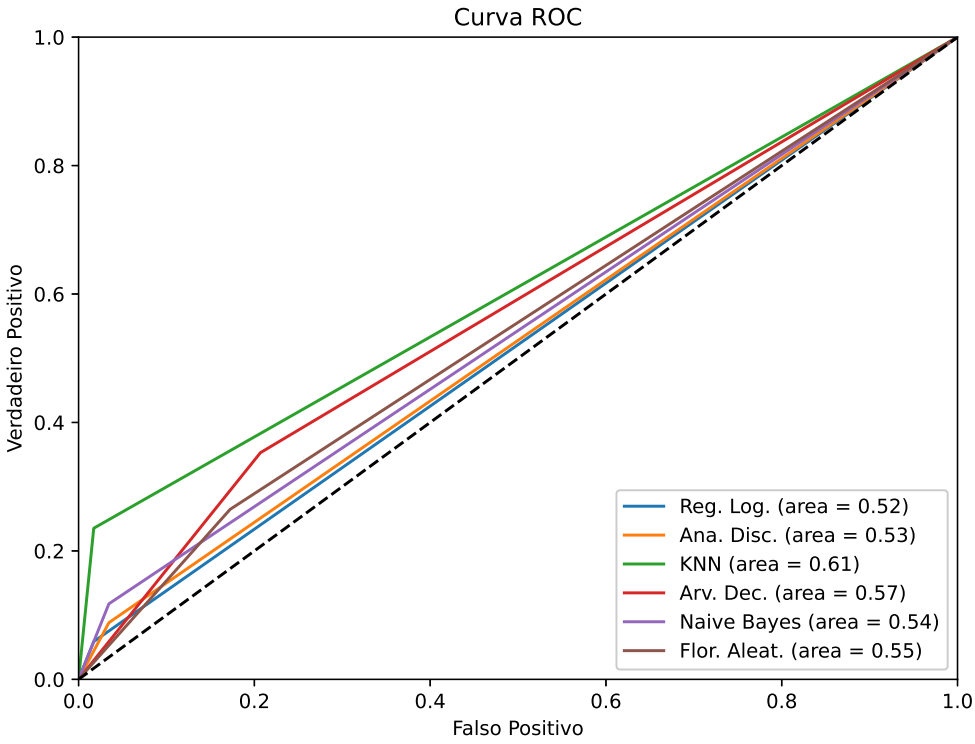

Figura 10 – Curva ROC dos Modelos

4 CONCLUSÃO

Os 6 modelos propostos para predição de resultados com base no conjunto de dados geraram respostas positivas: fizeram predições não aleatórias e com acurácia acima de 50%. A análise das métricas de desempenho, sobretudo a acurácia, mostrou que o KNN gera o melhor modelo quando avaliado no conjunto de testes, o que difere da análise inicial do conjunto de treinamentos, que indicava que o melhor modelo seria Naive Bayes.

Nesse problema - predição de sobrevivência ou não de pacientes após determinado tempo de execução de procedimento - é fundamental que os erros sejam baixos, ou seja, a acurácia seja a maior possível, por se tratar de tema sensível (vida ou morte de alguém) e que pode impactar consideravelmente nas decisões da pessoa no curto e médio prazo. Esse motivo levou a acurácia a ser a princpal métrica de avaliação e o KNN o melhor modelo calculado, conforme também comprovado em Curva ROC.

REFERÊNCIAS

Admir Antonio Betarelli Junior. **Análise Multivariada**. [S.l.]. Disponível em <https://www2.ufjf.br/lates//files/2016/12/Conte%c3%bado-7-%e2%80%93-A$_A n\%c3\%a1 lise - discriminante - AD.pdf$>.$Visitado em Dezembro$, 2021.

Equipe Oncoguia. **Taxa de Sobrevida para Câncer de Mama**. [S.l.], 2020. Disponível em <http://www.oncoguia.org.br/conteudo/taxa-de-sobrevida-para-cancer-de-mama/6563/264/>. Visitado em Dezembro, 2021.

Gabriel Stankevix. **Aprendizado Supervisionado**. [S.l.], 2020. Disponível em <https://medium.com/@gabriel.stankevix/aprendizado-supervisionado-903f31ed586>. Visitado em Novembro, 2021.

GONZALEZ, L. de A. **Regressão Logística e suas Aplicações**. 46 p. Dissertation (Monografia) — Universidade Federal do Maranhão, São Luís, Maranhão, 2018.

Gustavo Santos. **O que é Regularização**. [S.l.], 2021. Disponível em <https://medium.com/data-hackers/o-que-%C3%A9-regulariza%C3%A7%C3%A3o-caa1967b8b13>. Visitado em Novembro, 2021.

Instituto Nacional de Câncer José Alencar Gomes da Silva. **A situação do câncer de mama no Brasil: síntese de dados dos sistemas de informação**. [S.l.], 2019. Disponível em <https://www.inca.gov.br/sites/ufu.sti.inca.local/files/media/document/a_situacao_ca_mama_brasil_2019.p Visitado em Dezembro, 2021.

MARTINS, L. G. **Aplicação de um modelo de aprendizagem de máquina para predição de Churn: um estudo de caso**. 34 p. Dissertation (Monografia) — Universidade Federal do Ceará, Fortaleza, Ceará, 2021.

SILVA, C. G. R. da. **Aplicação da Qualidade da informação na área da saúde: aplicação de algoritmos de aprendizado de máquina**. 94 p. Dissertation (Monografia) — Universidade de Caxias do Sul, Caxias do Sul, Rio Grande do Sul, 2017.

Tjen-Sien Lim. **Haberman's Survival Data Set**. [S.l.], 1999. Disponível em <https://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival/>. Visitado em Novembro, 2021.

APÊNDICE A — CÓDIGO UTILIZADO

```
## Bibliotecas:
import pandas as pd
import numpy as np
import seaborn as sn
import matplotlib.pyplot as plt
import graphviz
from scipy import stats
from sklearn import metrics
from sklearn import tree
from sklearn.metrics import confusion_matrix, classification_report,
   accuracy_score
from sklearn.metrics import roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split

## Funcoes:

#Metricas de Avaliacao:
def metricas(teste,predito):
    nome={}
    acc=metrics.accuracy_score(teste,predito)
    mcc=metrics.matthews_corrcoef(teste,predito)
    r=metrics.recall_score(teste,predito)
    p=metrics.precision_score(teste,predito)
    f1=metrics.f1_score(teste,predito)
    nome = dict(zip(['acc','mcc','r','p','f1'], [acc,mcc,r,p,f1]))
    return nome

def mc(teste,predito):
```

```
    confusion_matrix = pd.crosstab(teste, predito, rownames=['Real'],
   colnames=['Predito'])
    return confusion_matrix

def mapa_calor_mc(original,melhor):
    fig, (ax1, ax2) = plt.subplots(1,2)
    sn.heatmap(original, ax=ax1,annot=True)
    ax1.title.set_text('Primeiro Modelo')
    sn.heatmap(melhor, ax=ax2,annot=True)
    ax2.title.set_text('Melhor Modelo')
    fig.set_size_inches(9,3)

def shapiro_test(coluna):
    n=df.columns.get_loc(coluna)
    sp = stats.shapiro(df.iloc[:,n])
    est = sp.statistic
    pvalor = sp.pvalue
    if pvalor > 0.05:
        tipo_distribuicao='Normal'
    else:
        tipo_distribuicao='N o Normal'
    shap = dict(zip(['estatistica','pvalor','tipo_distribuicao'], [est,
   pvalor,tipo_distribuicao]))
    print(f'O p-valor da vari vel {coluna}    {pvalor:,.6f}, logo a
   distribui  o    {tipo_distribuicao}.')
    return shap

def create_comp(original,melhor):
    data_items=original.items()
    data_list=list(data_items)
    df1=pd.DataFrame(data_list,columns=['metrica','valor'])
    data_items=melhor.items()
    data_list=list(data_items)
    df2=pd.DataFrame(data_list,columns=['metrica','valor'])
    df_comp = df1.merge(df2, how='left', on='metrica',suffixes=['
   _primeiro_modelo','_melhor_modelo'])
    return df_comp

## Variaveis Globais:

#Variaveis dos Modelos:
```

```
n_folds=30 #N mero de folds
cv = RepeatedStratifiedKFold(n_splits=n_folds, n_repeats=3,
    random_state=1) #Gera  o do KFold
scoring='accuracy' #M trica de avalia  o principal

#Variaveis de ambiente (salvar figuras):
PROJECT_ROOT_DIR = "."
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "graficos")
os.makedirs(IMAGES_PATH, exist_ok=True)

## Importacao do df:
df = pd.read_csv('haberman.data', header=None, names=['idade','ano','
    nodulos','status'])
#Ver dataframe:
print('Dados carregados:')
df.head()

## Exploracao do df:

#Informacoes:
print('Tipologia:')
info=df.info()
print(info)
print()

#Vis o Geral:
resume = df.describe()
print('Dados Gerais:')
print(resume)
print()

#Comparacao das Variaveis
fig = plt.figure()
fig.suptitle('Explora  o das Vari veis: Idade e Ano')
fig.set_size_inches(6,6)
ax = fig.add_subplot(111)
df.boxplot(column=['idade','ano'])
plt.savefig(os.path.join(IMAGES_PATH, 'Exploracao_DF_Idade_Ano.eps'))
plt.show()
fig = plt.figure()
fig.suptitle('Explora  o das Vari veis: Nodulos')
```

```
fig.set_size_inches(6,6)
ax = fig.add_subplot(111)
df.boxplot(column=['nodulos'])
plt.savefig(os.path.join(IMAGES_PATH, 'Exploracao_DF_Nodulos.eps'))
plt.show()
fig = plt.figure()
fig.suptitle('Explora  o das Vari veis: Status')
fig.set_size_inches(6,6)
ax = fig.add_subplot(111)
df.boxplot(column=['status'])
plt.savefig(os.path.join(IMAGES_PATH, 'Exploracao_DF_Status.eps'))
plt.show()

## Teste de Shapiro:
print('Teste de Shapiro:')
shap_idade=shapiro_test('idade')
shap_nodulo=shapiro_test('nodulos')
shap_ano=shapiro_test('ano')
shap_status=shapiro_test('status')

## Divis o em treinamento e teste:

#Tamanho do dataframe:
print(f'Tamanho do df: {len(df):,.0f}')

#Vari veis de an lise:
x=df[['idade','ano','nodulos']]

#Vari vel resposta:
y=df['status']

#Divis o:
np.random.seed(7)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,
    random_state=0)
print(f'Tamanho do treinamento: {len(x_train):,.0f}')
print(f'Tamanho do teste: {len(x_test):,.0f}')

## K Folds:
# Cria  o dos modelos:
models = []
```

```
models.append(('Regress o Log stica', LogisticRegression(solver='
    newton-cg')))
models.append(('KNN', KNeighborsClassifier()))
models.append((' rvore  de Decis o', DecisionTreeClassifier()))
models.append(('Naive Bayes', GaussianNB()))
models.append(('Florestas Aleat rias', RandomForestClassifier()))
models.append(('An lise Discriminante',LinearDiscriminantAnalysis()))

# Avalia  o dos modelos com os conjuntos de treino:
results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, x_train.values, y_train.values,
     cv=cv, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

# Compara  o dos modelos
fig = plt.figure()
fig.suptitle('Compara  o dos Modelos')
fig.set_size_inches(12,6)
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.savefig(os.path.join(IMAGES_PATH, '
    Comparacao_Modelos_KFold_Treinamento.eps'))
plt.show()

## Regress o Log stica:

#Defini  o do modelo:
logistic_regression=LogisticRegression()
logistic_regression.fit(x_train,y_train)
y_pred_rl=logistic_regression.predict(x_test)

#Matriz de Confus o:
rl_mc = mc(y_test,y_pred_rl)

#Medidas de avalia  o:
```

```
rl_metricas = metricas(y_test,y_pred_rl)

#Tunning do modelo:
np.random.seed(7)
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]

param_grid = dict(solver=solvers,penalty=penalty,C=c_values)
grid_search = GridSearchCV(estimator=logistic_regression, param_grid=
    param_grid, n_jobs=-1, cv=cv, scoring=scoring,error_score=0)
grid_result = grid_search.fit(x_train, y_train)

best_solver=grid_result.best_params_['solver']
best_penalty=grid_result.best_params_['penalty']
best_C=grid_result.best_params_['C']

#Refazendo o modelo:
logistic_regression=LogisticRegression(solver=best_solver,penalty=
    best_penalty,C=best_C)
logistic_regression.fit(x_train.values,y_train.values)
y_pred_new_rl=logistic_regression.predict(x_test.values)

#Matriz de Confus o:
rl_new_mc = mc(y_test,y_pred_new_rl)

#Medidas de avalia  o:
rl_new_metricas = metricas(y_test,y_pred_new_rl)

#Mostrar resultados:
print('Matriz de Confus o:')
mapa_calor_mc(rl_mc,rl_new_mc)
plt.savefig(os.path.join(IMAGES_PATH, 'RL_Matriz_Confusao.eps'))
plt.show()
print()
print('M tricas de Valida  o:')
df_rl=create_comp(rl_metricas,rl_new_metricas)
print(df_rl)

## An lise Discriminante:
```

```
#Defini  o do modelo:
discriminant_analysis=LinearDiscriminantAnalysis()
discriminant_analysis.fit(x_train,y_train)
y_pred_da=discriminant_analysis.predict(x_test)

#Matriz de Confus o:
ad_mc = mc(y_test,y_pred_da)

#Medidas de avalia  o:
ad_metricas = metricas(y_test,y_pred_da)

#Mostrar resultados:
print('Matriz de Confus o:')
sn.heatmap(ad_mc,annot=True)
plt.savefig(os.path.join(IMAGES_PATH, 'AD_Matriz_Confusao.eps'))
plt.show()
print()
print('M tricas de Valida  o:')
data_items=ad_metricas.items()
data_list=list(data_items)
df_ad=pd.DataFrame(data_list,columns=['metrica','valor_melhor_modelo'])
print(df_ad)

## KNN:

#Defini  o do modelo:
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train.values,y_train.values)
y_pred_knn=knn.predict(x_test.values)

#Matriz de Confus o:
knn_mc = mc(y_test,y_pred_knn)

#Medidas de avalia  o:
knn_metricas = metricas(y_test,y_pred_knn)

#Definir melhor k:
np.random.seed(7)

k = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]
distancias = ['euclidean', 'manhattan', 'minkowski']
```

```
param_grid = dict(n_neighbors=k, metric=distancias)
grid = GridSearchCV(estimator=knn, param_grid=param_grid, scoring=
    scoring, cv=cv)
grid_result = grid.fit(x_train.values, y_train.values)
best_k=grid_result.best_params_['n_neighbors']
best_acc=grid_result.best_score_
best_dist=grid_result.best_params_['metric']

#Refazendo o modelo:
knn=KNeighborsClassifier(n_neighbors=best_k,metric=best_dist)
knn.fit(x_train.values,y_train.values)
y_pred_knn=knn.predict(x_test.values)

#Matriz de Confus o:
knn_new_mc = mc(y_test,y_pred_knn)

#Medidas de avalia  o:
knn_new_metricas = metricas(y_test,y_pred_knn)

#Mostrar resultados:
print('Matriz de Confus o:')
mapa_calor_mc(knn_mc,knn_new_mc)
plt.savefig(os.path.join(IMAGES_PATH, 'KNN_Matriz_Confusao.eps'))
plt.show()
print()
print('M tricas de Valida  o:')
df_knn=create_comp(knn_metricas,knn_new_metricas)
print(df_knn)

##  rvore  de Decis o:

#Defini  o do modelo:
arv_dec=DecisionTreeClassifier()
arv_dec.fit(x_train.values,y_train.values)
y_pred_arv_dec=arv_dec.predict(x_test.values)

#Matriz de Confus o:
arv_dec_mc = mc(y_test,y_pred_arv_dec)

#Medidas de avalia  o:
```

```
arv_dec_metricas = metricas(y_test,y_pred_arv_dec)

#Tunning o modelo:
np.random.seed(7)

max_depth = [2, 3, 5, 10, 20]
min_samples_leaf = [5, 10, 20, 50, 100]
criterion = ["gini", "entropy"]

param_grid = dict(max_depth=max_depth, min_samples_leaf=
    min_samples_leaf,criterion=criterion)

grid_search = GridSearchCV(estimator=arv_dec, param_grid=param_grid, cv
    =n_folds, n_jobs=-1, verbose=1, scoring = "accuracy")
grid_result=grid_search.fit(x_train, y_train)
prof=grid_result.best_params_['max_depth']
minimo=grid_result.best_params_['min_samples_leaf']
crit=grid_result.best_params_['criterion']

#Refazendo modelo:
arv_dec=DecisionTreeClassifier(max_depth=prof, min_samples_leaf=minimo,
    criterion=crit)
arv_dec.fit(x_train.values,y_train.values)
y_pred_arv_dec_new=arv_dec.predict(x_test.values)

#Mostrar a  rvore :
print(' rvore  de Decis o do Melhor Modelo:')
fig = plt.figure(figsize=(15,10))
_ = tree.plot_tree(arv_dec, feature_names=x.columns,filled=True,
    class_names=['1','2'])
plt.savefig(os.path.join(IMAGES_PATH, 'Arvore_decisao.eps'))
plt.show()

#Matriz de Confus o:
arv_dec_new_mc = mc(y_test,y_pred_arv_dec_new)

#Medidas de avalia  o:
arv_dec_new_metricas = metricas(y_test,y_pred_arv_dec_new)

#Mostrar resultados:
print('Matriz de Confus o:')
```

```
mapa_calor_mc(arv_dec_mc,arv_dec_new_mc)
plt.savefig(os.path.join(IMAGES_PATH, 'Arvore_Decisao_Matriz_Confusao.
    eps'))
plt.show()
print()
print('M tricas de Valida  o:')
df_arv_dec=create_comp(arv_dec_metricas,arv_dec_new_metricas)
print(df_arv_dec)

## Naive Bayes:

#D e f i n i  o do modelo:
NB=GaussianNB()
NB.fit(x_train.values,y_train.values)
y_pred_NB=NB.predict(x_test.values)

#Matriz de Confus o:
NB_mc = mc(y_test,y_pred_NB)

#Medidas de avalia  o:
NB_metricas = metricas(y_test,y_pred_NB)

#Tunning o modelo:
np.random.seed(7)

var_smoothing = np.logspace(0,-9, num=100)
param_grid = dict(var_smoothing=var_smoothing)

grid_search = GridSearchCV(estimator=NB, param_grid=param_grid, cv=cv,
    n_jobs=-1, verbose=1, scoring=scoring)
grid_result=grid_search.fit(x_train, y_train)
var=grid_result.best_params_['var_smoothing']

#Refazendo modelo:
NB=GaussianNB(var_smoothing=var)
NB.fit(x_train.values,y_train.values)
y_pred_new_NB=NB.predict(x_test.values)

#Matriz de Confus o:
NB_new_mc = mc(y_test,y_pred_new_NB)
```

```
#Medidas de avalia  o:
NB_new_metricas = metricas(y_test,y_pred_new_NB)

#Mostrar resultados:
print('Matriz de Confus o:')
mapa_calor_mc(NB_mc,NB_new_mc)
plt.savefig(os.path.join(IMAGES_PATH, 'NB_Matriz_Confusao.eps'))
plt.show()
print()
print('M tricas de Valida  o:')
df_NB=create_comp(NB_metricas,NB_new_metricas)
print(df_NB)

## Floresta Aleat ria:

#Defini  o do modelo:
rf=RandomForestClassifier()
rf.fit(x_train.values,y_train.values)
y_pred_rf=rf.predict(x_test.values)

#Matriz de Confus o:
rf_mc = mc(y_test,y_pred_rf)

#Medidas de avalia  o:
rf_metricas = metricas(y_test,y_pred_rf)

#Tunning o modelo:
np.random.seed(7)

n_estimators = [10,20,50,100,200,500]
max_features = ['sqrt', 'log2']
param_grid = dict(n_estimators=n_estimators,max_features=max_features)

grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=cv,
   n_jobs=-1, verbose=1, scoring=scoring)
grid_result=grid_search.fit(x_train, y_train)
n_estimators=grid_result.best_params_['n_estimators']
max_features=grid_result.best_params_['max_features']

#Refazendo modelo:
rf=RandomForestClassifier(n_estimators=n_estimators,max_features=
```

```
    max_features)
rf.fit(x_train.values,y_train.values)
y_pred_new_rf=rf.predict(x_test.values)

#Matriz de Confus o:
rf_new_mc = mc(y_test,y_pred_new_rf)

#Medidas de avalia  o:
rf_new_metricas = metricas(y_test,y_pred_new_rf)

#Mostrar resultados:
print('Matriz de Confus o:')
mapa_calor_mc(rf_mc,rf_new_mc)
plt.savefig(os.path.join(IMAGES_PATH, '
    Floresta_Aleatoria_Matriz_Confusao.png'))
plt.show()
print()
print('M tricas de Valida  o:')
df_rf=create_comp(rf_metricas,rf_new_metricas)
print(df_rf)

## Compara  o de resultados

#Cria  o de coluna de m todo:
df_rl['metodo']='regressao logistica'
df_ad['metodo']='analise discriminante'
df_knn['metodo']='knn'
df_NB['metodo']='naive bayes'
df_arv_dec['metodo']='arvore decisao'
df_rf['metodo']='florestas aleatorias'

#DataFrame de an lise:
df_analise=[]
frames=[df_rl[['metrica','valor_melhor_modelo','metodo']],df_ad[['
    metrica','valor_melhor_modelo','metodo']],df_knn[['metrica','
    valor_melhor_modelo','metodo']],df_NB[['metrica','
    valor_melhor_modelo','metodo']],df_arv_dec[['metrica','
    valor_melhor_modelo','metodo']],df_rf[['metrica','
    valor_melhor_modelo','metodo']]]
df_analise=pd.concat(frames)
```

```
#Ver m todo com melhores m tricas:
idx = df_analise.groupby(['metrica'])['valor_melhor_modelo'].transform(
    max) == df_analise['valor_melhor_modelo']
df_result=df_analise[idx]
df_result.sort_values(by='metrica')

## Curva ROC:

fpr_rl, tpr_rl, threshold = metrics.roc_curve(y_test, y_pred_new_rl,
    pos_label=2)
roc_rl = metrics.auc(fpr_rl,tpr_rl)
fpr_da, tpr_da, threshold = metrics.roc_curve(y_test, y_pred_da,
    pos_label=2)
roc_da = metrics.auc(fpr_da,tpr_da)
fpr_knn, tpr_knn, threshold = metrics.roc_curve(y_test, y_pred_knn,
    pos_label=2)
roc_knn = metrics.auc(fpr_knn,tpr_knn)
fpr_arv_dec, tpr_arv_dec, threshold = metrics.roc_curve(y_test,
    y_pred_arv_dec_new,pos_label=2)
roc_arv_dec = metrics.auc(fpr_arv_dec,tpr_arv_dec)
fpr_nb, tpr_nb, threshold = metrics.roc_curve(y_test, y_pred_new_NB,
    pos_label=2)
roc_nb = metrics.auc(fpr_nb,tpr_nb)
fpr_rf, tpr_rf, threshold = metrics.roc_curve(y_test, y_pred_new_rf,
    pos_label=2)
roc_rf = metrics.auc(fpr_rf,tpr_rf)
fig = plt.figure()
fig.set_size_inches(8,6)
plt.plot(fpr_rl, tpr_rl, label='Reg. Log. (area = %0.2f)' % roc_rl)
plt.plot(fpr_da, tpr_da, label='Ana. Disc. (area = %0.2f)' % roc_da)
plt.plot(fpr_knn, tpr_knn, label='KNN (area = %0.2f)' % roc_knn)
plt.plot(fpr_arv_dec, tpr_arv_dec, label='Arv. Dec. (area = %0.2f)' %
    roc_arv_dec)
plt.plot(fpr_nb, tpr_nb, label='Naive Bayes (area = %0.2f)' % roc_nb)
plt.plot(fpr_rf, tpr_rf, label='Flor. Aleat. (area = %0.2f)' % roc_rf)
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('Falso Positivo')
plt.ylabel('Verdadeiro Positivo')
plt.title('Curva ROC')
```

```
plt.legend(loc="lower right")
plt.savefig(os.path.join(IMAGES_PATH, 'Curva_ROC.eps'))
plt.show()
```