

# Tratamento e Mineração de Dados

## Naive Bayes e Florestas Randômicas

Marcos Pereira

12 de maio de 2020

## 1 Abordagem Teórica

- Aprendizado Supervisionado (ML)
- Naive Bayes

## 2 Implementação do Código

- Bibliotecas
- Tratamento e Visualização dos Dados
- Hipótese\*
- Pré processamento
- Naive Bayes

## 3 Resultados

## 4 Discussão

## 5 Otimização

$$\mathbf{R} = \left[ \begin{array}{ccc|c} x_1 & x_2 & x_3 & y_1 \\ x'_1 & x'_2 & x'_3 & y_2 \\ x_1 & x'_2 & x_3 & y_3 \end{array} \right] \left. \vphantom{\begin{array}{ccc|c} x_1 & x_2 & x_3 & y_1 \\ x'_1 & x'_2 & x'_3 & y_2 \\ x_1 & x'_2 & x_3 & y_3 \end{array}} \right\} \begin{array}{l} \text{classe} \\ \text{instâncias} \end{array} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix}$$

Desse modo os vetores instâncias são os vetores  $\mathbf{x}_i$ , é importante notar que as relações equivalem aos nossos data sets.

- Criar modelos genéricos;
- Evitar a criação de modelos superajustados (overfitting);
- Hipótese apresenta resultados específicos.
- Modelo ajusta-se aos dados de treinamento mas não é eficaz para novos exemplos.[1]
- Treinar o modelo e testá-lo. Como?

Tratamento e Mineração de Dados

Marcos Pereira

Abordagem Teórica

- Considerações
- Aprendizado Supervisionado (ML)
- Naive Bayes

Implementação do Código

- Bibliotecas
- Tratamento e Visualização dos Dados
- Hipótese\*
- Pré processamento
- Naive Bayes

- Resultados
- Discussão
- Otimização



# Tratamento e Mineração de Dados

Marcos Pereira

## Abordagem Teórica

Considerações

Aprendizado Supervisionado (ML)

Naive Bayes

## Implementação do Código

Bibliotecas

Tratamento e Visualização dos Dados

Hipótese\*

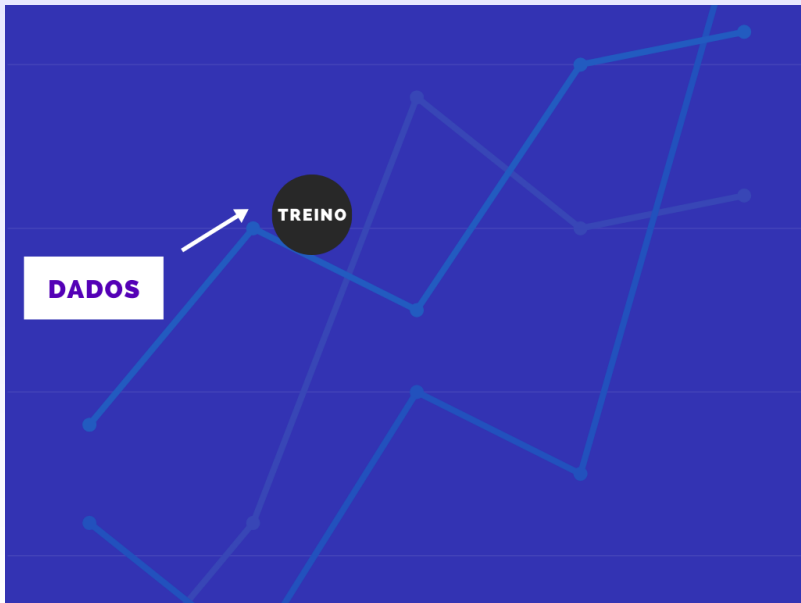
Pré processamento

Naive Bayes

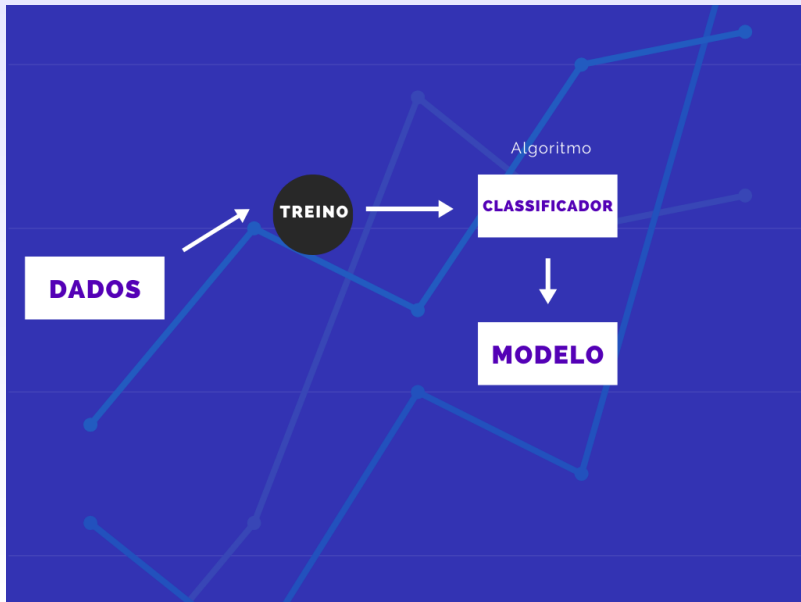
## Resultados

## Discussão

## Otimização
























- Conjunto de Algoritmos de aprendizado supervisionado;
- Paradigma estatístico:
  - Avaliação de hipóteses pela máxima verossimilhança (inferência Bayesiana).
- Teorema de Bayes<sup>1</sup>:

$$\begin{aligned} P(y|x_1, x_2, \dots, x_n) &= \prod_{i=1}^n \frac{P(y)P(x_i|y)}{P(x_i)} \\ &\equiv \frac{P(y) (P(x_1|y)P(x_2|y)\dots P(x_n|y))}{P(x_1)P(x_2)\dots P(x_n)} \end{aligned} \quad (1)$$

- A probabilidade A Priori  $P(\mathbf{x}) \in \mathbb{R}$  é constante, e utilizaremos a seguinte regra de classificação:

$$P(y|\mathbf{x}) \propto \prod_{i=1}^n P(y)P(x_i|y) \quad (2)$$

---

<sup>1</sup>O símbolo  $\prod$  representa um produtório. 



- Máxima Probabilidade A Posteriori para realizar predição de dados:

$$Y = \arg \max_y \prod_{i=1}^n P(y)P(x_i|y) \quad (3)$$

- $Y$  = Probabilidade de  $y$  posteriori condicionada a  $\mathbf{x}$
- Naive Bayes Gaussiano. Verossimilhança[2]:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi}\sigma_y} \exp \left[ - \left( \frac{x_i - \mu_y}{\sqrt{2}\sigma_y} \right)^2 \right]$$

- Predição de dados:

$$Y = \arg \max_y P(y) \exp \left[ - \frac{1}{2\sigma_y^2} \sum_{i=1}^n (x_i - \mu_y)^2 \right]$$

- Em alguns casos os atributos não são condicionalmente independentes;

- Ensemble Learning: Combinar previsões de vários métodos de classificação a fim de otimizar a previsão dos dados;
- Realizaremos a combinação de vários classificadores independentes, o resultado é a média das previsões obtidas;
- Floresta Randômica: Baseado em árvores de decisão, consiste em criarmos  $n$  árvores de decisão, treiná-las e obter a resposta de cada uma delas;
- Várias fontes de aleatoriedade reduzem a variância do "forest estimator", árvores de decisão sozinhas apresentam uma alta variância e consequentemente produzem modelos superajustados;

- Implementação do Método será realizado utilizando o Sklearn [2];
- Tratar dados → Pré processamento de dados → Treinamento → Avaliação → Predição → Otimização

## Importação das bibliotecas e métodos:

```
import numpy as np
import pandas as pd
```

```
from datetime import datetime
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

em seguida os algoritmos e métodos de Machine Learning do  
sklearn:

```
#ML
```

```
from sklearn.preprocessing import LabelEncoder,
OneHotEncoder
```

```
from sklearn.compose import make_column_transformer
```

```
#Divisão dados de treino e teste
```

```
from sklearn.model_selection import train_test_split
```

```
#Florestas Aleatórias
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
#Método de Naive Bayes
```

```
from sklearn.naive_bayes import GaussianNB
```

```
#Taxa de acerto e de erro
```

```
from sklearn.metrics import accuracy_score
```

```
#Visualizaremos a matriz de confusão através da
```

```
#Yellow Brick
```

```
from yellowbrick.classifier import ConfusionMatrix
```

# Começamos o tratamento dos dados, importando o dataset usando pandas

```
df = pd.read_csv('Data_Base.csv', encoding =  
                  'ISO-8859-1')
```

```
df.head()
```

verificamos então que a tabela encontra-se sem informação com relação aos atributos ou até mesmo em relação à classe. Levantar suposição com base nos dados contidos no Data Frame.

# Tratamento e Mineração de Dados

Marcos Pereira

## Abordagem Teórica

Considerações

Aprendizado Supervisionado (ML)

Naive Bayes

## Implementação do Código

Bibliotecas

Tratamento e Visualização dos Dados

Hipótese\*

Pré processamento

Naive Bayes

## Resultados

## Discussão

## Otimização

```
In [5]: DF.head()
Out[5]:
```

0	182351688	2018-01-02T00:00	1	2018	08	354916128	...	B	Cana	Venda Bruta	431.41131292320006	140738.00750943169	326.2269748
0	182452212	2018-01-04T00:00	1	2018	09	354914539	...	A	Soja	Venda Bruta	143.803771	9523.271841	66.224076
1	183769629	2018-01-08T00:00	1	2018	09	354914637	...	C	Soja	Venda Bruta	17.256453	588.312615	34.092327
2	183757146	2018-01-09T00:00	1	2018	08	354915778	...	B	Batata	Venda Bruta	11.983648	1054.273963	87.976049
3	183771827	2018-01-10T00:00	1	2018	08	354916121	...	B	Cana	Venda Bruta	57.521508	23002.341012	399.891130
4	183743145	2018-01-10T00:00	1	2018	09	354915757	...	A	[Cultura Não Encontrada]	Venda Bruta	599.182379	65097.633939	108.644106

```
[5 rows x 19 columns]
```

Definir nome aos atributos para realizar a análise:

#

## Tratamento e Mineração de Dados

Marcos  
Pereira

### Abordagem Teórica

Considerações

Aprendizado  
Supervisionado  
(ML)

Naive Bayes

### Implementação do Código

Bibliotecas

Tratamento e  
Visualização  
dos Dados

Hipótese\*

Pré  
processamento

Naive Bayes

### Resultados

### Discussão

### Otimização

```
colunas = ['ID', 'Data/Hora Dia', 'Mês',  
           'Ano', 'D#', 'Código', 'Código 2',  
           'Código 3', 'Tipo de venda', 'UF',  
           'Código 4', 'Família', 'Produto 1',  
           'ABC', 'Produto 2', 'Venda Bruta col',  
           'Venda Bruta 1', 'Venda Bruta 2',  
           'Venda Bruta 3']
```

```
df = pd.read_csv('Data_Base.csv', names=colunas,  
encoding = 'ISO-8859-1')
```



- Transformar segundo atributo no formato datetime;
- Remover atributos que são invariantes;
- Analisar uma visualização dos dados a partir da matriz de correlação;
- Analisar o plot dos pares;

```
YmdHM = (lambda x: datetime.strptime(x,  
    '%Y-%m-%dT%H:%M'))
```

```
#Year month day hour minute  
#converter coluna em formato datetime
```

```
df['Data/Hora Dia'] =  
    df['Data/Hora Dia'].apply(YmdHM)
```

```
#Verificar quais colunas possuem valores  
#que não se repetem e remover
```

```
verificar = [len(df.iloc[:,i].value_counts().values)  
    for i in range(len(df.columns.values))]
```

```
unit, index = -1, []  
for i in verificar:  
    unit+=1  
    print('verificando {}'.format(colunas[unit]))
```

```
if i==1:
    index.append(unit)
    df.drop(colunas[unit], axis=1,
            inplace=True)
    print('Coluna {} removida'
          .format(colunas[unit]))
    #colunas.remove(colunas[unit])

for i in index:
    colunas.remove(colunas[i])

#remover 'Mês'
colunas.remove('Mês')

df.drop('Mês', inplace=True, axis=1)
```

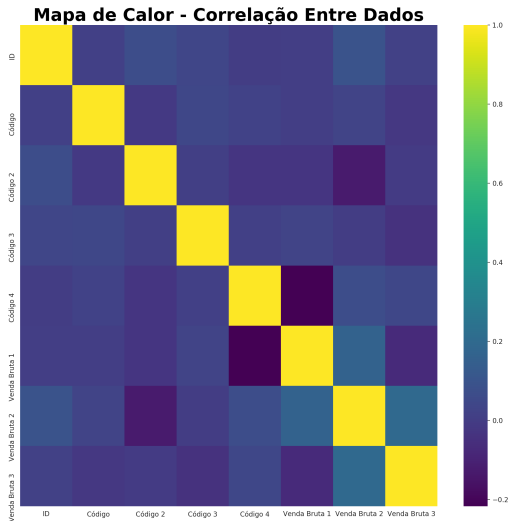


Figura: Atributos não estão fortemente correlacionados;

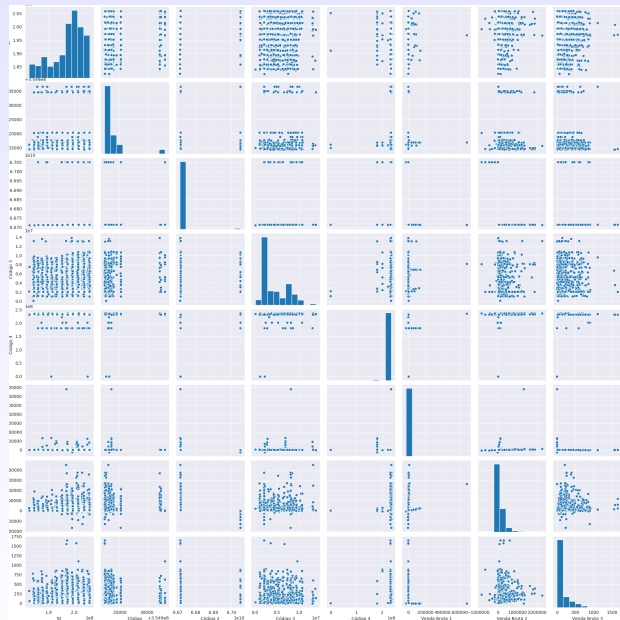


Figura: Consequência da correlação fraca

- Remover dados irrelevantes para o aprendizado, como o ID;
- Preparação para realizar a Codificação de Categorias (Categorical Encoding):
  - Alguns algoritmos do Python não conseguem ler dados categóricos;
  - Associar itens categóricos a dados numéricos;
  - Disponível apenas na v2 do programa.
  - Ambas versões estão no GitHub.
- Box plot para verificar outliers;
- Os mesmos passos são tomados no segundo versionamento do programa;

```
df3 = df.drop(['ID', 'D#', 'Código',  
              'Código 2', 'Código 3', 'Código 4'], axis=1)
```

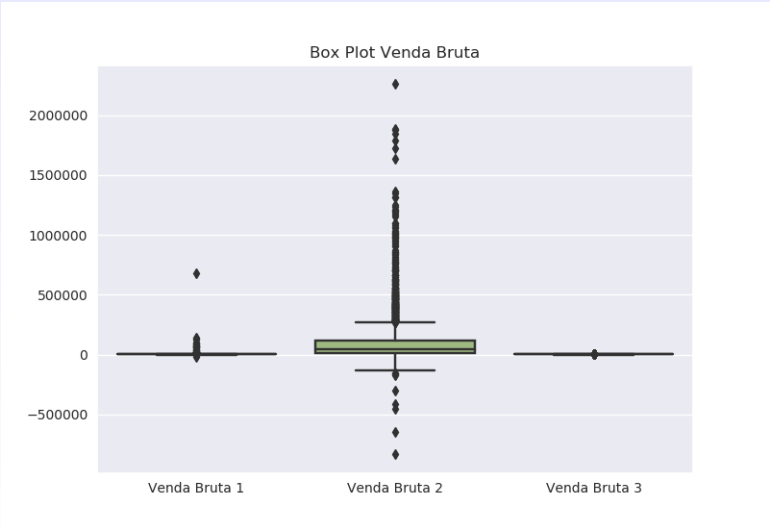
*#Remover outliers em atributos categóricos  
#(os que só aparecem uma vez ou são desconhecidos)*

```
df3=df3[df3['Produto 1'].apply(lambda x:  
                               x!= df3['Produto 1'].max())]
```

```
df3 = df3[df3['Tipo de venda'].apply(lambda x:  
                                     x!='[Gr.Cliente Não Encontrado]')]
```

```
df3 = df3[df3['Produto 2'].apply(lambda x:  
                                False if x=='[Cultura Não Encontrada]'  
                                else (False if x=='[Cultura Não Especificada]'  
                                else True))]
```

```
df3 = df3[df3['Produto 1'].apply(lambda x:  
                                 x!='[Gr Segmento Não Encontrado]')]
```





#  
#  
#

```
df3 = df3[df3['Venda Bruta 1'] !=  
          df3['Venda Bruta 1'].max()]
```

```
df3 = df3[df3['Venda Bruta 2'] !=  
          df3['Venda Bruta 2'].max()]
```

```
df3 = df3[df3['Venda Bruta 2'] !=  
          df3['Venda Bruta 2'].min()]
```

```
df3 = df3[df3['Venda Bruta 2'] !=  
          df3['Venda Bruta 2'].min()]
```

*#recriar coluna de mês e outra do dia, o ano não  
#todas as instâncias ocorrem no mesmo ano.*

```
df3['Mês'] = df3['Data/Hora Dia'].apply(lambda x:  
    x.month)
```

```
df3['Dia'] = df3['Data/Hora Dia'].apply(lambda x:  
    x.day)
```

```
#remover atributo do tipo datetime  
#algoritmo não lê.
```

```
#remover data hora dia  
df3.drop('Data/Hora Dia',axis=1,inplace=True)
```

- Problema na coluna ABC, alguma das entradas não é um str, devemos remover essa entrada para realizar o label encoder;
- Classe d3info criada para verificar a variedade dos atributos e possivelmente aplicar one hot encoder.

#

Tratamento  
e Mineração  
de Dados

Marcos  
Pereira

Abordagem  
Teórica

Considerações

Aprendizado  
Supervisionado  
(ML)

Naive Bayes

Implementação  
do Código

Bibliotecas

Tratamento e  
Visualização  
dos Dados

Hipótese\*

Pré  
processamento

Naive Bayes

Resultados

Discussão

Otimização

```
df3 = df3[df3['ABC']].apply(lambda x:  
    False if type(x) != np.str else True)]
```

```
df3info.tipo_venda  
df3info.Produto_1  
df3info.Produto_2
```

*#tipo\_venda possui 4 categorias, os outros mais de 4.*

```
In [19]: df.head()
```

```
Out[19]:
```

	ID	Data/Hora	Dia	D#	Código	Código 2	Código 3	...	Produto 1 ABC	Produto 2	Venda Bruta 1	Venda Bruta 2	Venda Bruta 3
0	182351688	2018-01-02	08	354916128	66711752514	2120550	...	Inseticida B	Cana	431.411313	140738.007509	326.226975	
1	182452212	2018-01-04	09	354914539	66711761658	7537440	...	Inseticida A	Soja	143.803771	9523.271841	66.224076	
2	183769629	2018-01-08	09	354914637	66711774546	2062314	...	Outros TS C	Soja	17.256453	588.312615	34.092327	
3	183757146	2018-01-09	08	354915778	66711782061	3021450	...	Fungicida B	Batata	11.983648	1054.273963	87.976049	
4	183771027	2018-01-10	08	354916121	66711791952	2220502	...	Inseticida B	Cana	57.521508	23002.341012	399.891130	

```
[5 rows x 16 columns]
```

```
In [20]: df3.head()
```

```
Out[20]:
```

	Tipo de venda	UF	Familia	Produto 1 ABC	Produto 2	Venda Bruta 1	Venda Bruta 2	Venda Bruta 3	Mês	Dia	Lucro
0	Cooperativa	SP	Familia 94	Inseticida B	Cana	431.411313	140738.007509	326.226975	1	2	141495.645797
1	Venda Direta	MT	Familia 78	Inseticida A	Soja	143.803771	9523.271841	66.224076	1	4	9733.299688
2	Venda Direta	MA	Familia 3	Outros TS C	Soja	17.256453	588.312615	34.092327	1	8	639.661395
4	Cooperativa	SP	Familia 94	Inseticida B	Cana	57.521508	23002.341012	399.891130	1	10	23459.753651
6	Revenda	RJ	Familia 123	Non Crop A	Non Crop	14.380377	1157.759276	80.509660	1	11	1252.649313

Figura: Data Frame tratado x Data frame

- Esse modelo consistirá num diagnóstico de oportunidade de vendas, ele visa otimizar as transações realizadas.
- Dados indicam que a venda bruta total é bem alta;
- Quais condições provocam uma venda bem sucedida (acima da média);
- Suposição:

$$L = \sum_{i=1}^3 V l_i$$

- A classe será o lucro que consiste na soma dos três últimos atributos, que por hipótese, são o lucro líquido produzido pela compra.
- Separar dados previsores dos dados classificadores;
- Realizar o label encoder;

#

```
df3['Lucro'] = df3['Venda Bruta 1']+df3['Venda Bruta  
+df3['Venda Bruta 3']
```

```
#classe LabelEncoder do sklearn
```

```
labelencoder = LabelEncoder()
```

```
#dados previsores:
```

```
previsores2 = df3.drop(['Venda Bruta 1',  
                        'Venda Bruta 2', 'Venda Bruta 3',  
                        'Lucro'], axis=1).values
```

```
#Label Encoder aos dados categóricos
```

```
for n in [1,2,4,3,5]:  
    previsores2[:,n] =  
        labelencoder.fit_transform(previsores2[:,n])
```

```
#One Hot Encoder em tipo de vendas
```

```
onehotencoder = make_column_transformer(  
    (OneHotEncoder(categories='auto',  
        sparse=False),[0]), remainder='passthrough')
```

```
X = onehotencoder.fit_transform(previsores2)
```

```
#Classe classificada como 0 caso  $x < \text{mean}$  e 1  $x \geq \text{mean}$ 
```

```
classe2 = df3['Lucro'].apply(lambda x:  
    0 if x < df3['Lucro'].mean() else 1)
```



- Dividir os dados em dados de treino e teste;
- Modelo treinado com 70% dos dados e testado com 30%;
- Treinar modelo de Naive Bayes Gaussiano e de Floresta Randômica;
- Verificar os dados testados e precisão do modelo;

#

Tratamento  
e Mineração  
de DadosMarcos  
PereiraAbordagem  
Teórica

Considerações

Aprendizado  
Supervisionado  
(ML)

Naive Bayes

Implementação  
do Código

Bibliotecas

Tratamento e  
Visualização  
dos Dados

Hipótese\*

Pré  
processamento

Naive Bayes

Resultados

Discussão

Otimização

```
X_treino2, X_teste2, Y_treino2, Y_teste2 =  
    train_test_split(X, classe2, test_size=.3, ran
```

```
#=====
```

```
#    NAIVE BAYES GAUSSIANO
```

```
#=====
```

```
#Carregar classe a partir do sklearn
```

```
nb2 = GaussianNB()
```

```
#Fitar o modelo
```

```
nb2.fit(X_treino2,Y_treino2)
```

```
#Testar o modelo
```

```
prev2 = nb2.predict(X_teste2)
```

```
#Taxas de acerto e erro
```

```
taxa_acerto2 = (accuracy_score(Y_teste2, prev2))*100  
taxa_erro2 = (1 - taxa_acerto2/100)*100
```

```
#=====
```

```
# FLORESTA RANDOMICA
```

```
#=====
```

```
#Carregar classe a partir do sklearn.ensemble
```

```
floresta2 = RandomForestClassifier(n_estimators =  
    200)
```

```
#n_estimators é o número de árvores de decisão
```

```
#ajustar o modelo
```

```
floresta2.fit(X_treino2,Y_treino2)
```

```
#testar modelo
```

```
previsoes2 = floresta2.predict(X_teste2)
```

- Verificar Taxas de acerto e de erro;
- Criar matriz de confusão do modelo;
  - Métrica voltada para modelos de classificação;
  - Quantidade de FP, FN, VP, VN;
  - Acurácia e sensibilidade;

		Classe Predita	
		0	1
Classe Original	0	<b>TN</b>	<b>FP</b>
	1	<b>FN</b>	<b>TP</b>

Figura: Tabela de Confusão

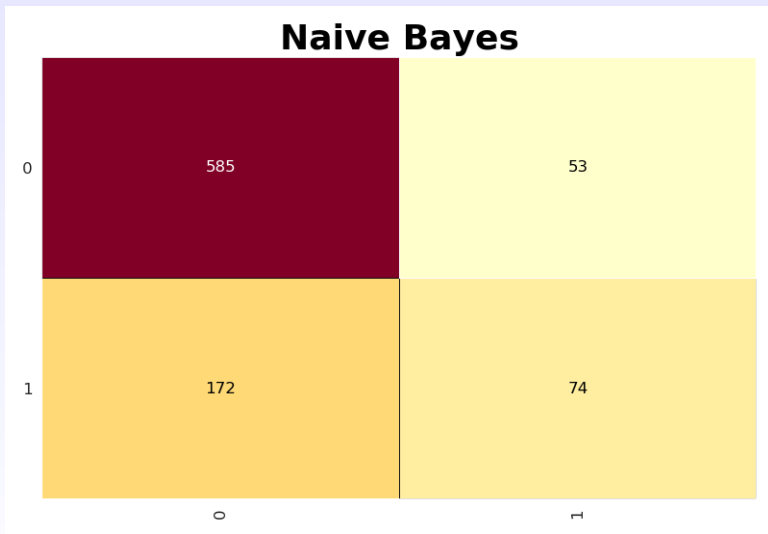


Figura: Naive Bayes

## Florestas Aleatórias 2

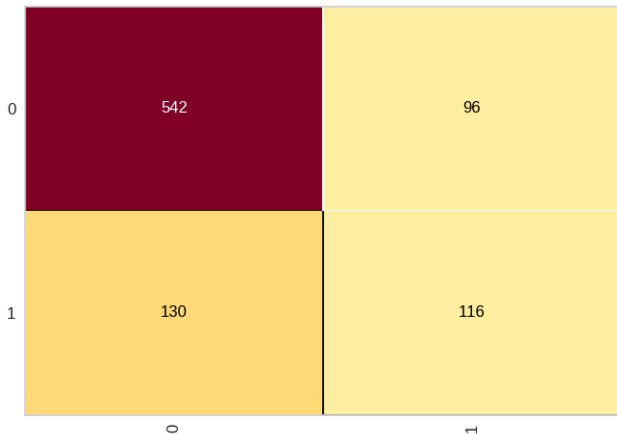


Figura: Naive Bayes

O modelo foi fitado com sucesso e apresentou as seguintes taxas:

Modelo	Taxa de acerto	Taxa de erro
Naive Bayes	74,55%	25,45%
Ensemble	74,77%	25,23%

As taxas da tabela acima implicam que o modelo foi bem treinado, mas levando em conta as hipóteses que foram tomadas. Obtendo acesso à mais informações sobre a lista, pode ser necessário recorrer a outros modelos de aprendizado.



O modelo pode ser otimizado levando em conta os valores reais dos atributos que não foram informados e tomando considerações reais sobre os valores do lucro bruto que não pode ser contabilizada devido à falta de informação

Projeto disponível no github



[https://github.com/acidbutter96/  
diagnostico-oportunidades/](https://github.com/acidbutter96/diagnostico-oportunidades/)



Lucas Ferreira, Murilo Gazzola, Deborah Ferrari, Marina Zupelari, Paula Paiva, and Jose Rodrigues Jr. Métodos de classificação aplicados à detecção automática de estilos de aprendizagem em um ambiente real de ensino. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 28, page 1517, 2017.



F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.