MODELOS DE DATA MINING APLICADOS À CLASSIFICAÇÃO DE PULSARES - CÓDIGO

June 1, 2019

1 Classificação de Pulsares usando modelos de Machine Learning

Este trabalho tem por objetivo a Análise da Base de dados HTRU sobre estrelas e pulsares, bem como a modelagem através de modelos de ML e comparação entre diferentes modelos.

2 Importando Bibliotecas e a Base de Dados

```
In [1]: #Importando bibliotecas principais
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: # importando ferramentas de pré-processamento e métricas.
        from sklearn.preprocessing import StandardScaler
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
        #importando modelos
        from sklearn.neural_network import MLPClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.naive_bayes import GaussianNB
```

```
In [3]: #importando dataset e colocando os nomes da colunas
        columns = ['Mean of the integrated profile','Standard deviation of the integrated profil
        data = pd.read_csv('HTRU_2.csv', names=columns)
```

3 Pré-processamento

Na fase do pré-processamento é feita a análise exploratória do modelo de forma que sejam apagadas inconsistências e valores ausentes nos dados, e que os melhores atributos sejam selecionados para serem usados para predição.

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17898 entries, 0 to 17897
Data columns (total 9 columns):
Mean of the integrated profile                  17898 non-null float64
Standard deviation of the integrated profile    17898 non-null float64
Excess kurtosis of the integrated profile       17898 non-null float64
Skewness of the integrated profile              17898 non-null float64
Mean of the DM-SNR curve                        17898 non-null float64
Standard deviation of the DM-SNR curve          17898 non-null float64
Excess kurtosis of the DM-SNR curve             17898 non-null float64
Skewness of the DM-SNR curve                    17898 non-null float64
Label                                           17898 non-null int64
dtypes: float64(8), int64(1)
memory usage: 1.2 MB
```

In [5]: data.head()

```
Out[5]:    Mean of the integrated profile  \
        0                      140.562500
        1                      102.507812
        2                      103.015625
        3                      136.750000
        4                       88.726562

           Standard deviation of the integrated profile  \
        0                                     55.683782
        1                                     58.882430
        2                                     39.341649
        3                                     57.178449
        4                                     40.672225

           Excess kurtosis of the integrated profile  \
        0                                  -0.234571
        1                                   0.465318
        2                                   0.323328
        3                                  -0.068415
        4                                   0.600866

           Skewness of the integrated profile  Mean of the DM-SNR curve  \
        0                           -0.699648                  3.199833
        1                           -0.515088                  1.677258
        2                            1.051164                  3.121237
        3                           -0.636238                  3.642977
        4                            1.123492                  1.178930

           Standard deviation of the DM-SNR curve  \
        0                               19.110426
```

```
1                                 14.860146
2                                 21.744669
3                                 20.959280
4                                 11.468720

   Excess kurtosis of the DM-SNR curve  Skewness of the DM-SNR curve  Label
0                             7.975532                     74.242225      0
1                            10.576487                    127.393580      0
2                             7.735822                     63.171909      0
3                             6.896499                     53.593661      0
4                            14.269573                    252.567306      0
```

In [6]: data.describe()

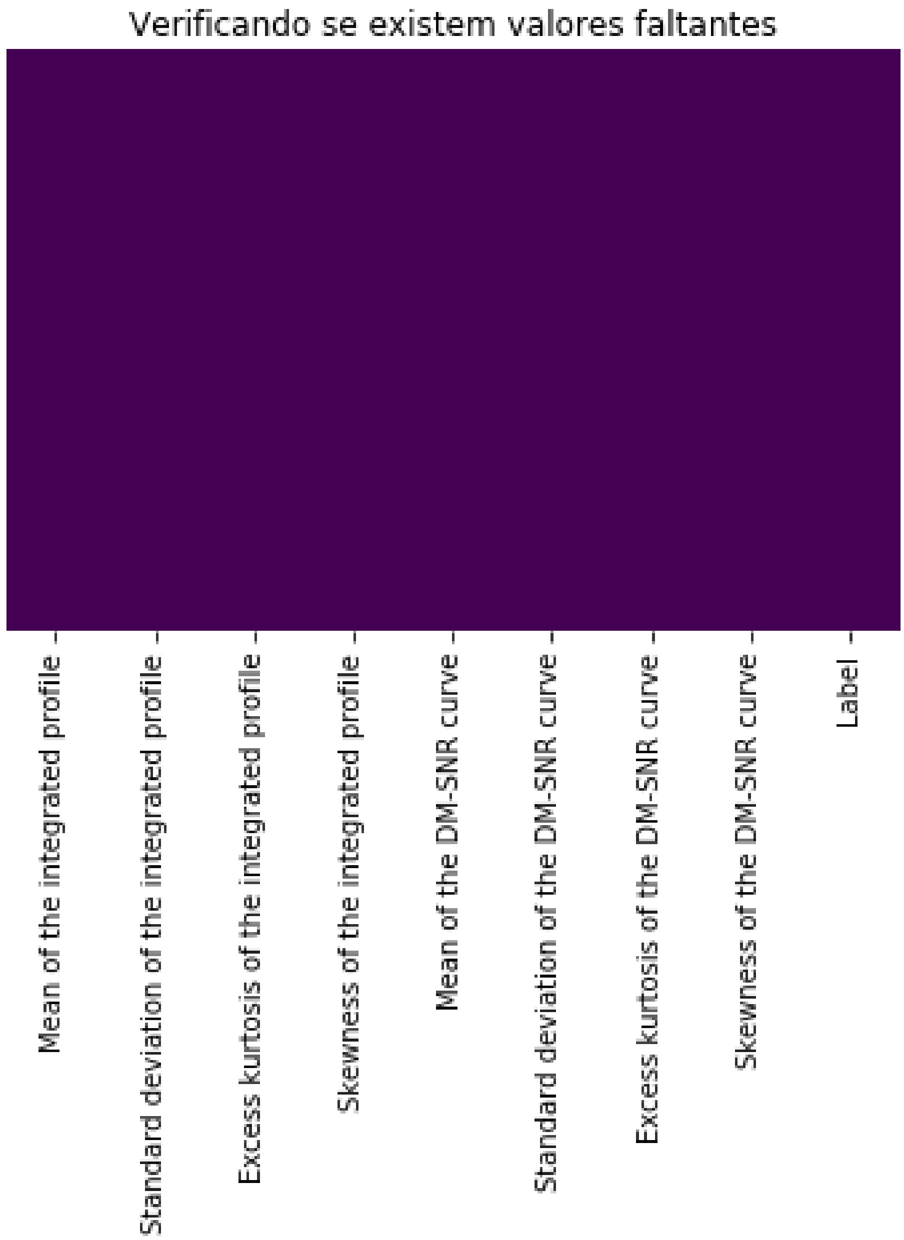
Out[6]:

```
       Mean of the integrated profile  \
count                    17898.000000
mean                       111.079968
std                         25.652935
min                          5.812500
25%                        100.929688
50%                        115.078125
75%                        127.085938
max                        192.617188

       Standard deviation of the integrated profile  \
count                                  17898.000000
mean                                      46.549532
std                                        6.843189
min                                       24.772042
25%                                       42.376018
50%                                       46.947479
75%                                       51.023202
max                                       98.778911

       Excess kurtosis of the integrated profile  \
count                               17898.000000
mean                                    0.477857
std                                     1.064040
min                                    -1.876011
25%                                     0.027098
50%                                     0.223240
75%                                     0.473325
max                                     8.069522

       Skewness of the integrated profile  Mean of the DM-SNR curve  \
count                        17898.000000              17898.000000
mean                             1.770279                 12.614400
std                              6.167913                 29.472897
```

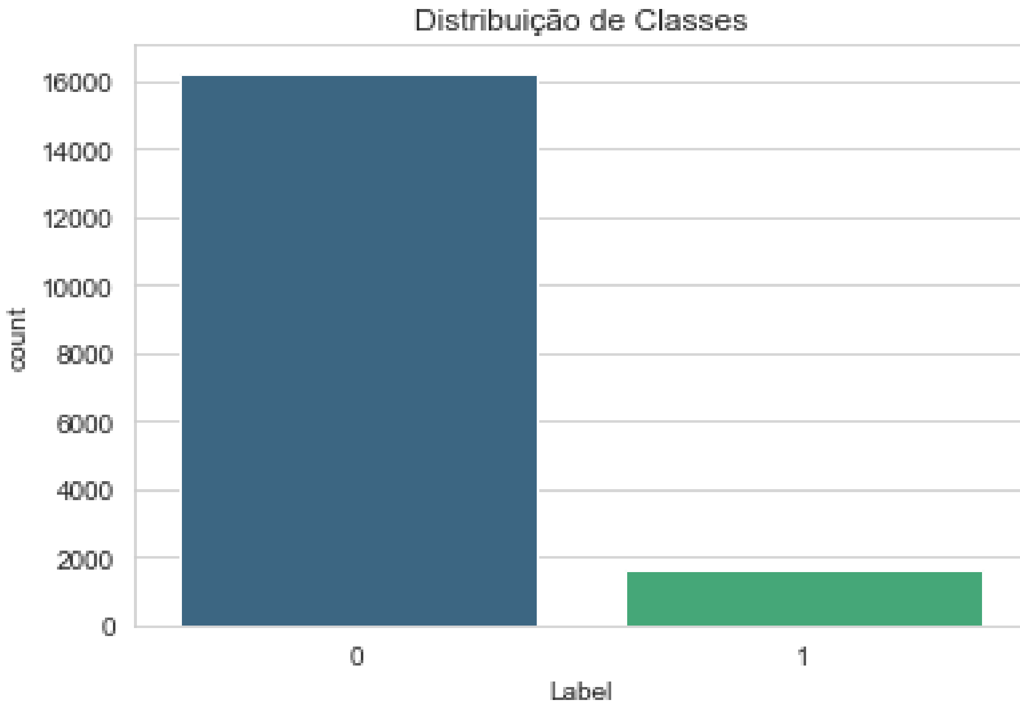
```
min                                  -1.791886             0.213211
25%                                  -0.188572             1.923077
50%                                   0.198710             2.801839
75%                                   0.927783             5.464256
max                                  68.101622           223.392140

       Standard deviation of the DM-SNR curve  \
count                            17898.000000
mean                                26.326515
std                                 19.470572
min                                  7.370432
25%                                 14.437332
50%                                 18.461316
75%                                 28.428104
max                                110.642211

       Excess kurtosis of the DM-SNR curve  Skewness of the DM-SNR curve  \
count                         17898.000000                  17898.000000
mean                              8.303556                    104.857709
std                               4.506092                    106.514540
min                              -3.139270                     -1.976976
25%                               5.781506                     34.960504
50%                               8.433515                     83.064556
75%                              10.702959                    139.309331
max                              34.539844                   1191.000837

              Label
count  17898.000000
mean       0.091574
std        0.288432
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
```

In [7]:
```python
sns.heatmap(data.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.title('Verificando se existem valores faltantes')
```

Out[7]:
```
Text(0.5,1,'Verificando se existem valores faltantes')
```

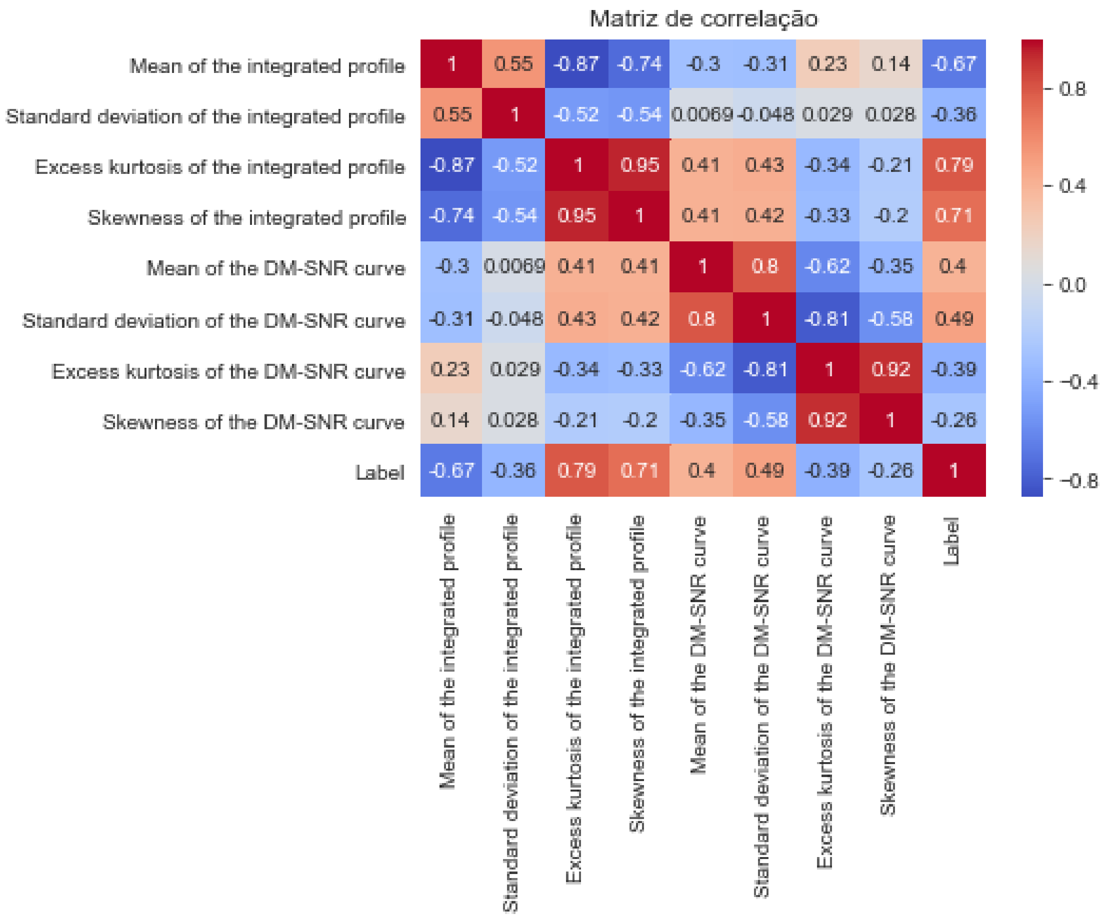



```
In [8]: sns.set_style('whitegrid')
        sns.countplot(x='Label',data=data,palette='viridis')
        plt.title('Distribuição de Classes')

Out[8]: Text(0.5,1,'Distribuição de Classes')
```



In [9]:
```
sns.heatmap(data.corr(),annot=True, cmap='coolwarm')
plt.title('Matriz de correlação')
#Como a base tem 18 mil dados não é necessário excluir nenhum atributo
```

Out[9]:
```
Text(0.5,1,'Matriz de correlação')
```




```
In [10]: #instanciando o padronizador dos dados
         scaler = StandardScaler()
```

```
In [11]: scaler.fit(data.drop(data.columns[-1],axis=1))
```

```
Out[11]: StandardScaler(copy=True, with_mean=True, with_std=True)
```

```
In [12]: #nova variável de atributos padronizados
         scaled_features = scaler.transform(data.drop(data.columns[-1],axis=1))
```

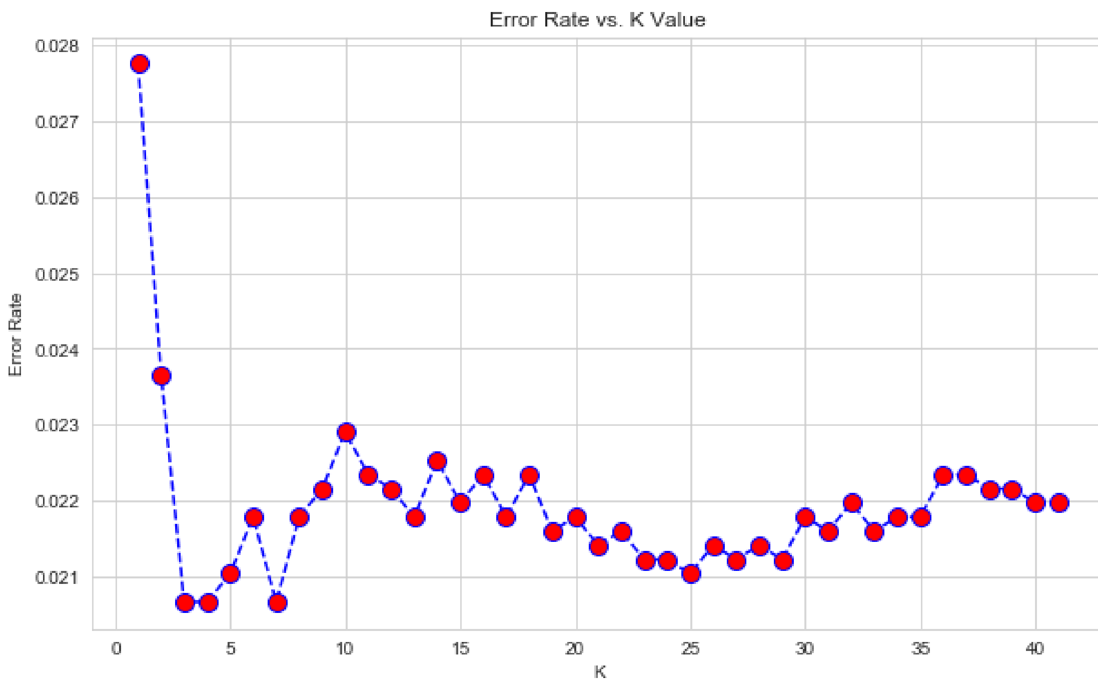
```
In [13]: #Divisão Treino e Teste
         X_train, X_test, y_train, y_test = train_test_split(scaled_features,data[data.columns[-
```

4 K Nearest Neighbour (KNN)

```
In [14]: #Escolhendo o melhor K
         error_rate = []
```

```
# Vai consumir algum tempo
for i in range(1,42):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

```
In [15]: plt.figure(figsize=(10,6))
         plt.plot(range(1,42),error_rate,color='blue', linestyle='dashed', marker='o',
                  markerfacecolor='red', markersize=10)
         plt.title('Error Rate vs. K Value')
         plt.xlabel('K')
         plt.ylabel('Error Rate')

Out[15]: Text(0,0.5,'Error Rate')
```

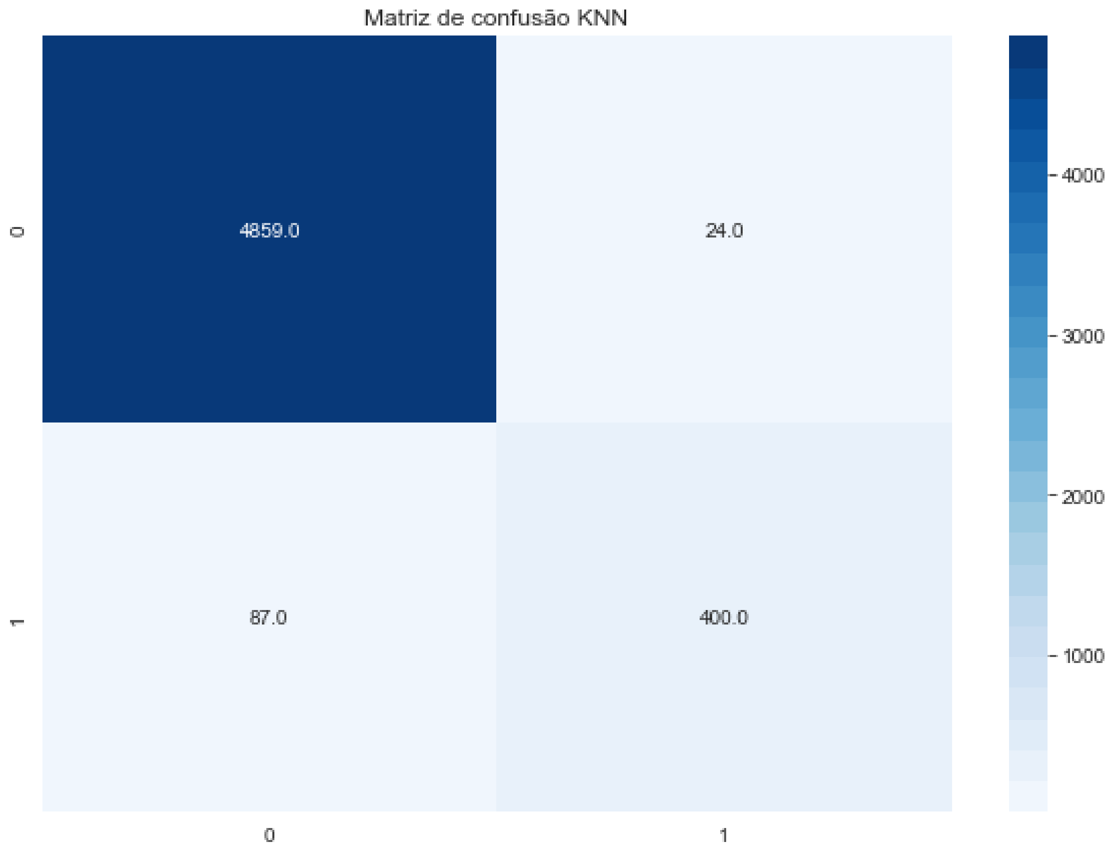



```
In [16]: knn = KNeighborsClassifier(n_neighbors=7)

         knn.fit(X_train,y_train)
         predictions = knn.predict(X_test)

         print('WITH K=7')
         print('\n')
         print(confusion_matrix(y_test,predictions))
```

```
print('\n')
print(classification_report(y_test,predictions))
print('\n')
knn_score = accuracy_score(y_test,predictions)
print('Acurácia:', knn_score)
```

```
WITH K=7


[[4859   24]
 [  87  400]]


             precision    recall  f1-score   support

          0       0.98      1.00      0.99      4883
          1       0.94      0.82      0.88       487

avg / total       0.98      0.98      0.98      5370



Acurácia: 0.9793296089385475
```

```
In [17]: plt.figure(figsize=(10,7))
         sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt='.1f', cmap=sns.colo
         plt.title('Matriz de confusão KNN')
         plt.show()
```



5 Gaussian Naive Bayes (GNB)

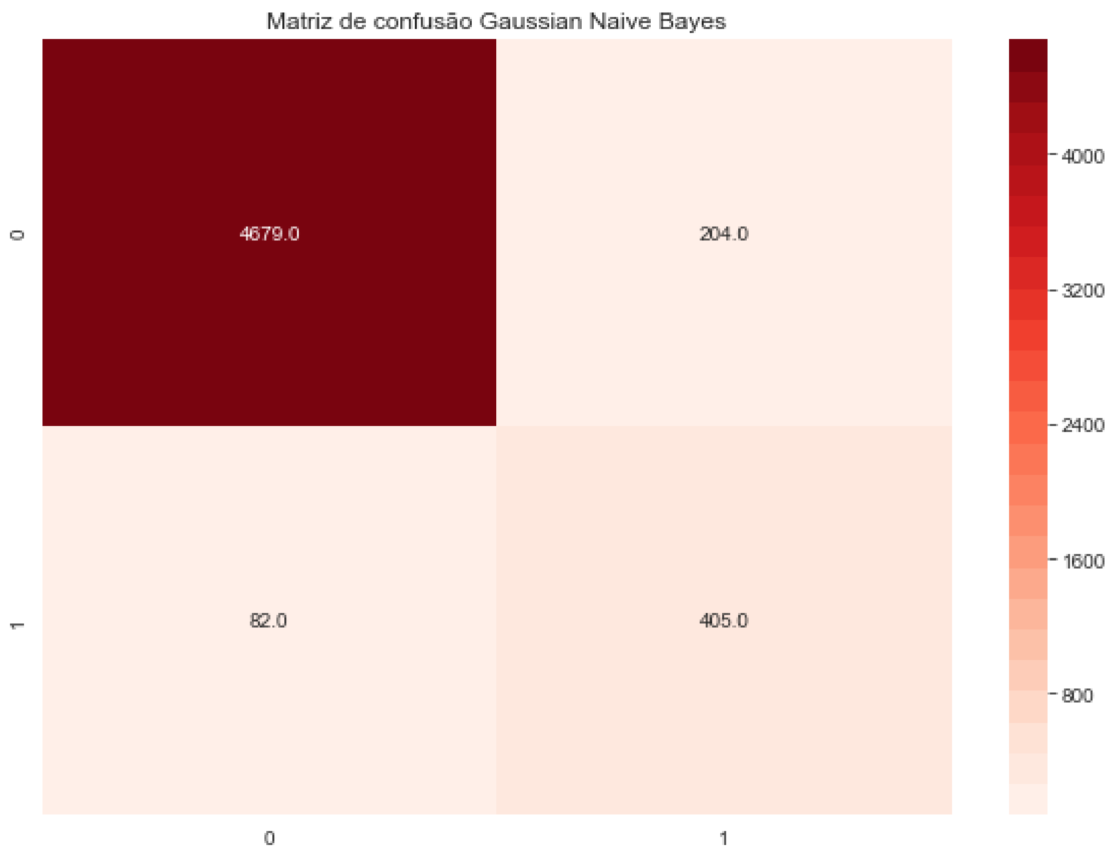
```
In [18]: gnb = GaussianNB()

In [19]: gnb.fit(X_train,y_train)
         predictions = gnb.predict(X_test)

In [20]: print(confusion_matrix(y_test,predictions))
         print('\n')
         print(classification_report(y_test,predictions))
         print('\n')
         gnb_score = accuracy_score(y_test,predictions)
         print('Acurácia:', gnb_score)
```

```
[[4679  204]
 [  82  405]]


             precision    recall  f1-score   support
```

```
           0       0.98      0.96      0.97      4883
           1       0.67      0.83      0.74       487

avg / total       0.95      0.95      0.95      5370
```

```
Acurácia: 0.9467411545623836
```

In [21]:
```
plt.figure(figsize=(10,7))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt='.1f', cmap=sns.colo
plt.title('Matriz de confusão Gaussian Naive Bayes')
plt.show()
```



6 Redes Neurais Artificiais (RNA)

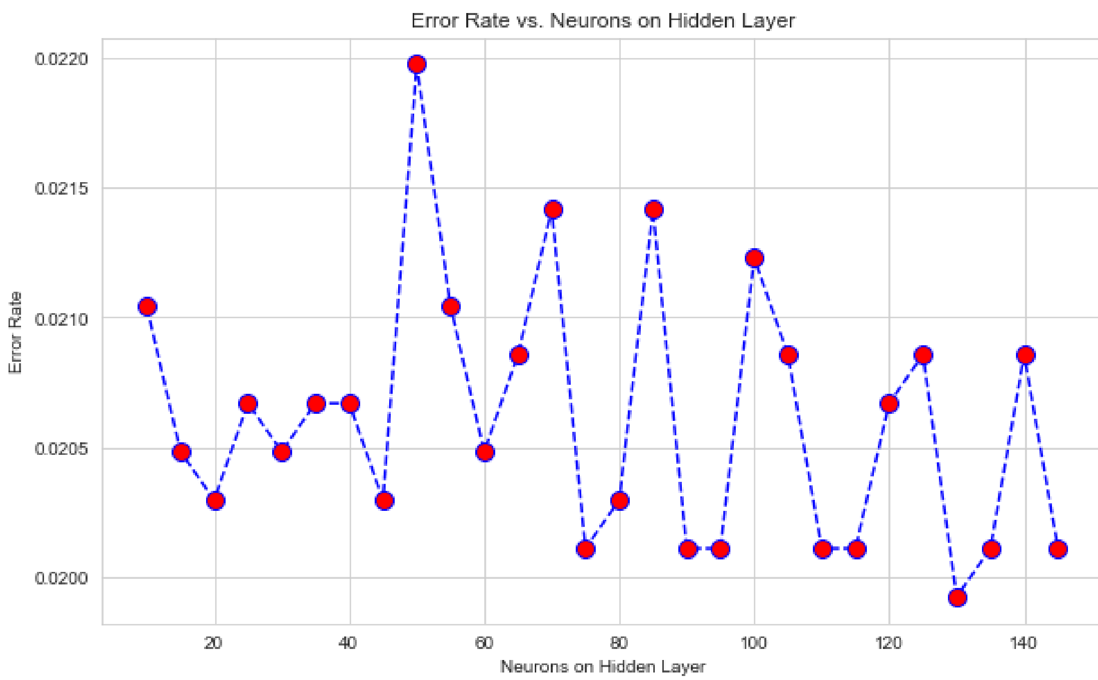
Rede com uma camada oculta

In [24]:
```
error_rate = []
```

```
# Vai demorar alguns minutos
for i in range(10,150,5):

    mlp = MLPClassifier(hidden_layer_sizes=(i, ), activation='logistic')
    mlp.fit(X_train,y_train)
    pred_i = mlp.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [25]:
```
plt.figure(figsize=(10,6))
plt.plot(range(10,150,5),error_rate,color='blue', linestyle='dashed', marker='o',
         markerfacecolor='red', markersize=10)
plt.title('Error Rate vs. Neurons on Hidden Layer')
plt.xlabel('Neurons on Hidden Layer')
plt.ylabel('Error Rate')
```

Out[25]:
```
Text(0,0.5,'Error Rate')
```



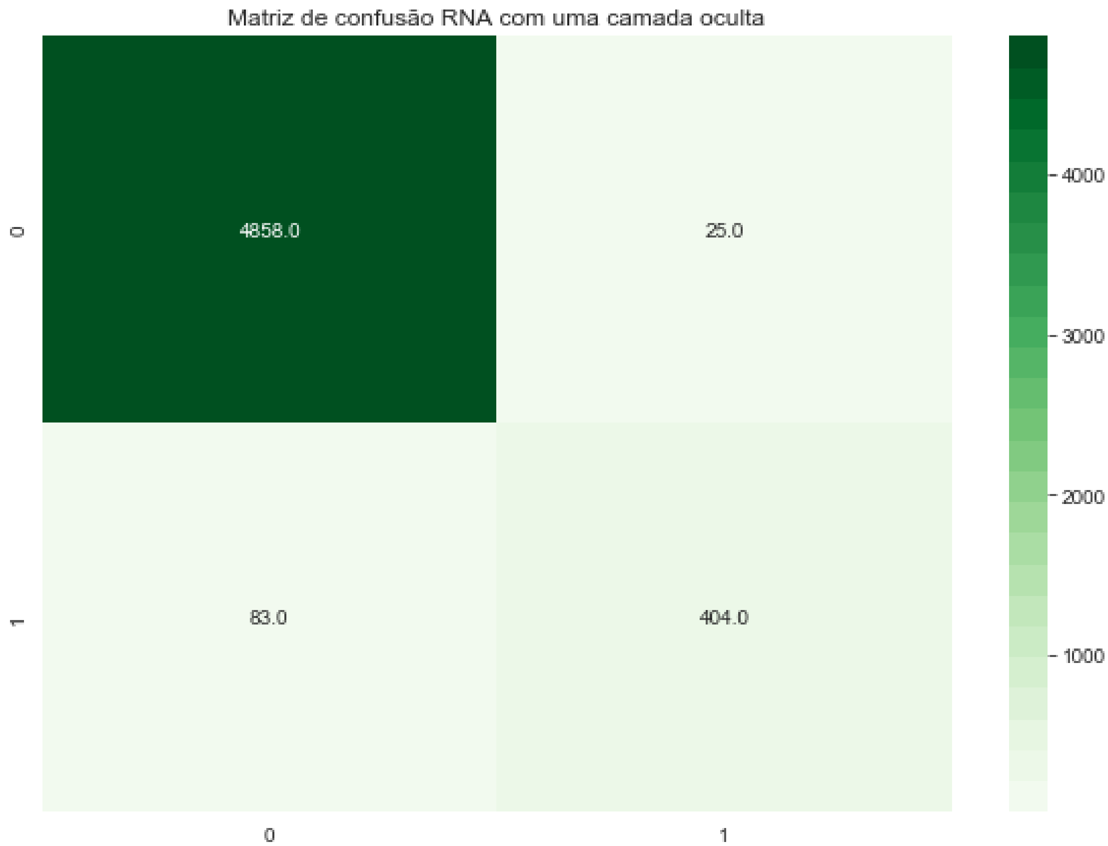
In [26]:
```
mlp = MLPClassifier(hidden_layer_sizes=(75, ), activation='logistic')
mlp.fit(X_train,y_train)
predictions = mlp.predict(X_test)
```

In [27]:
```
print('WITH 75 Neurons on Hidden Layer')
print('\n')
print(confusion_matrix(y_test,predictions))
print('\n')
print(classification_report(y_test,predictions))
```

```
        print('\n')
        mlp_score = accuracy_score(y_test,predictions)
        print('Acurácia:', mlp_score)
```

```
WITH 75 Neurons on Hidden Layer


[[4858   25]
 [  83  404]]


             precision    recall  f1-score   support

          0       0.98      0.99      0.99      4883
          1       0.94      0.83      0.88       487

avg / total       0.98      0.98      0.98      5370



Acurácia: 0.9798882681564246
```

```
In [28]: plt.figure(figsize=(10,7))
         sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt='.1f', cmap=sns.colo
         plt.title('Matriz de confusão RNA com uma camada oculta')
         plt.show()
```

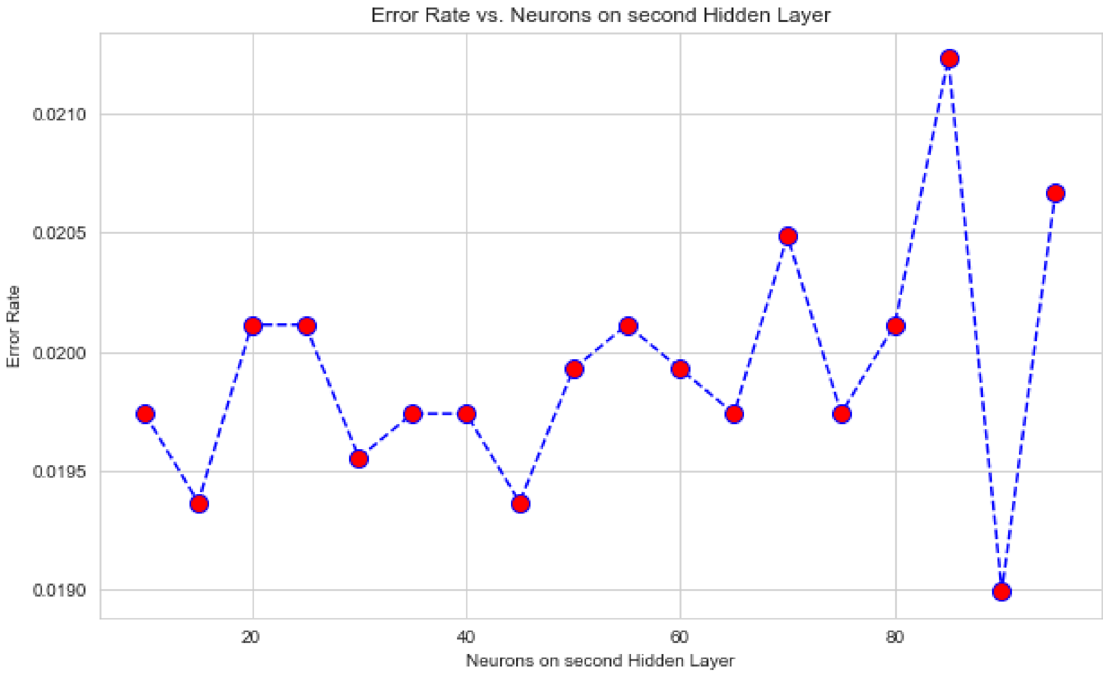


Verificando o ganho de desempenho com a adição de uma nova camada oculta

```
In [31]: # Vai demorar alguns minutos
         error_rate = []
         for i in range(10,100,5):

             mlp = MLPClassifier(hidden_layer_sizes=(75,i, ), activation='logistic')
             mlp.fit(X_train,y_train)
             pred_i = mlp.predict(X_test)
             error_rate.append(np.mean(pred_i != y_test))
```

```
In [32]: plt.figure(figsize=(10,6))
         plt.plot(range(10,100,5),error_rate,color='blue', linestyle='dashed', marker='o',
                  markerfacecolor='red', markersize=10)
         plt.title('Error Rate vs. Neurons on second Hidden Layer')
         plt.xlabel('Neurons on second Hidden Layer')
         plt.ylabel('Error Rate')
```

```
Out[32]: Text(0,0.5,'Error Rate')
```




```
In [33]: mlp = MLPClassifier(hidden_layer_sizes=(75, 45, ), activation='logistic')
         mlp.fit(X_train,y_train)
         predictions = mlp.predict(X_test)

In [34]: print('WITH 75 Neurons on Second Hidden Layer')
         print('\n')
         print(confusion_matrix(y_test,predictions))
         print('\n')
         print(classification_report(y_test,predictions))
         print('\n')
         second_mlp_score = accuracy_score(y_test,predictions)
         print('Acurácia:', second_mlp_score)
```

```
WITH 75 Neurons on Second Hidden Layer


[[4860   23]
 [  84  403]]


             precision    recall  f1-score   support

          0       0.98      1.00      0.99      4883
          1       0.95      0.83      0.88       487
```

```
avg / total       0.98      0.98      0.98      5370
```

```
Acurácia: 0.9800744878957169
```

In [35]:
```python
plt.figure(figsize=(10,7))
sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt='.1f', cmap=sns.colo
plt.title('Matriz de confusão RNA com duas camadas ocultas')
plt.show()
```