Trabalho Final da disciplina de Machine Learning: Classificadores e Validação de Modelos

Grupo: Harlan e Emmanuel

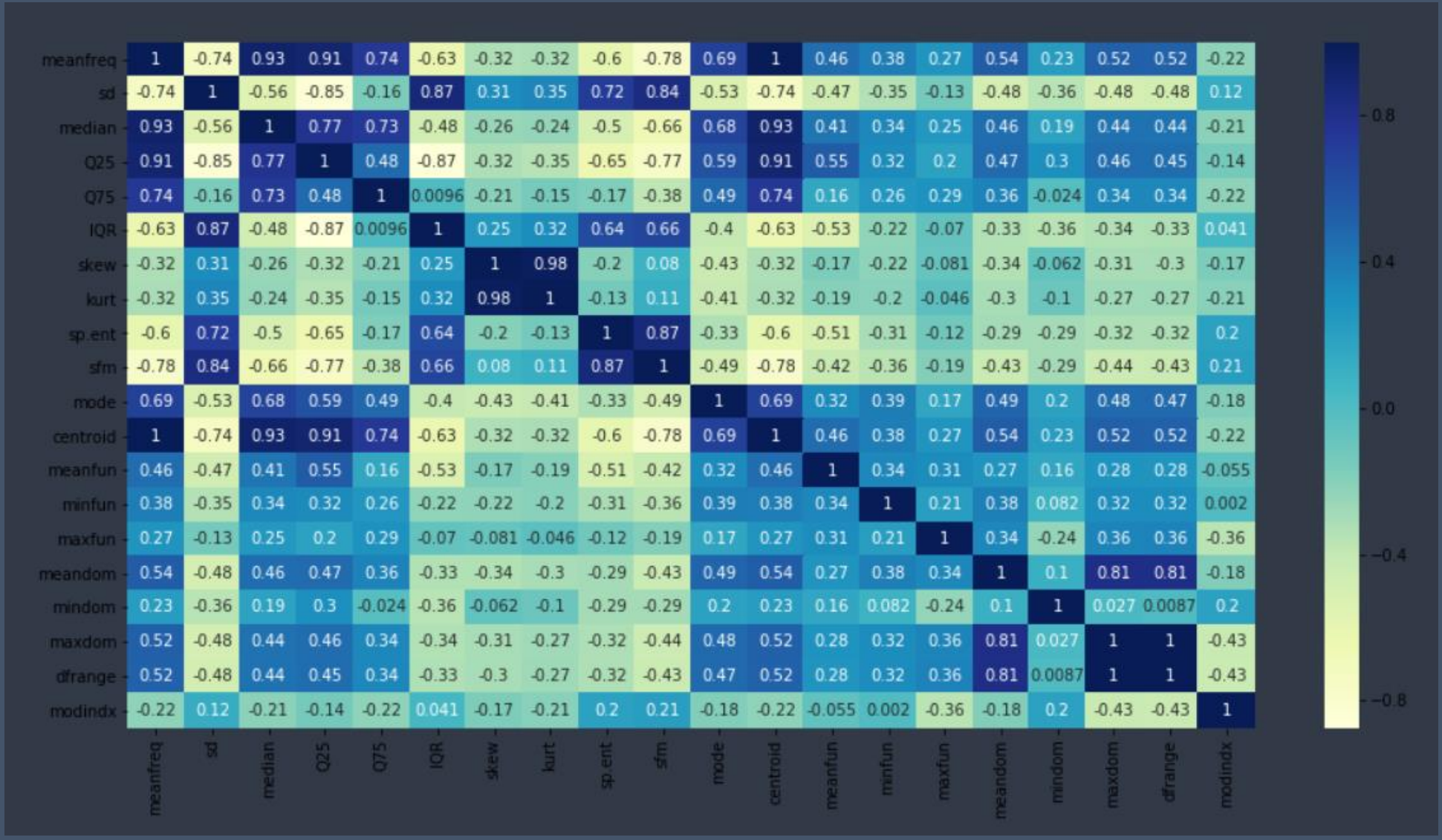
Github: https://github.com/hmaas00/projeto_final_voice_gender

Dataset Utilizado

- **Voice Gender:** https://www.kaggle.com/primaryobjects/voicegender

- O dataset possui a classificação de vozes de homens e mulheres baseada nas propriedades acústicas da fala humana. São 3168 pontos de dados pré-processados com a análise da frequência no intervalo de 0hz-280hz (intervalo da voz humana).

Correlação entre Variáveis



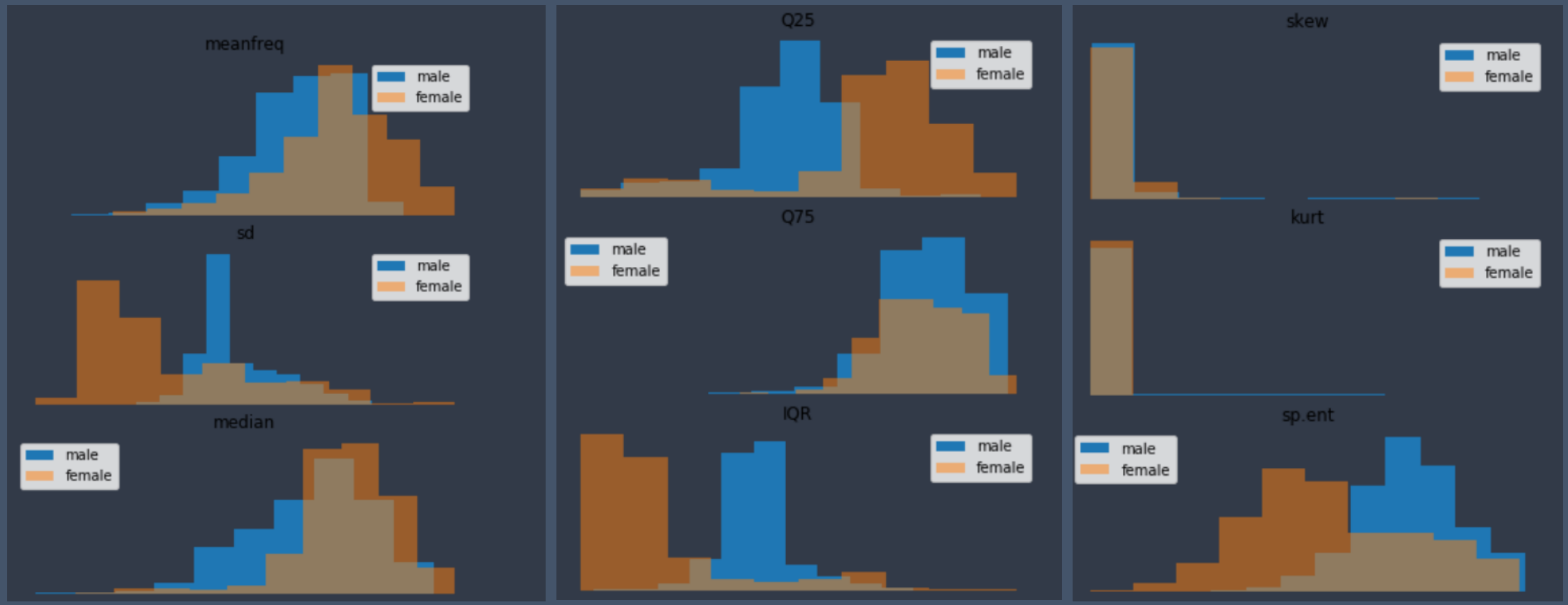
A alta correlação entre variáveis desperta a intuição de que a quantidade de variáveis é desnecessária

- Corr(meanfreq x median) = 0.93 (medidas de centro)
- Corr(sd x IQR) = 0.87 (medidas de variação)
- Corr(maxdom x dfrange) = 1 (maximum of dominant frequency e range of dominant frequency)
- Corr(meanfreq x centroid) = 1

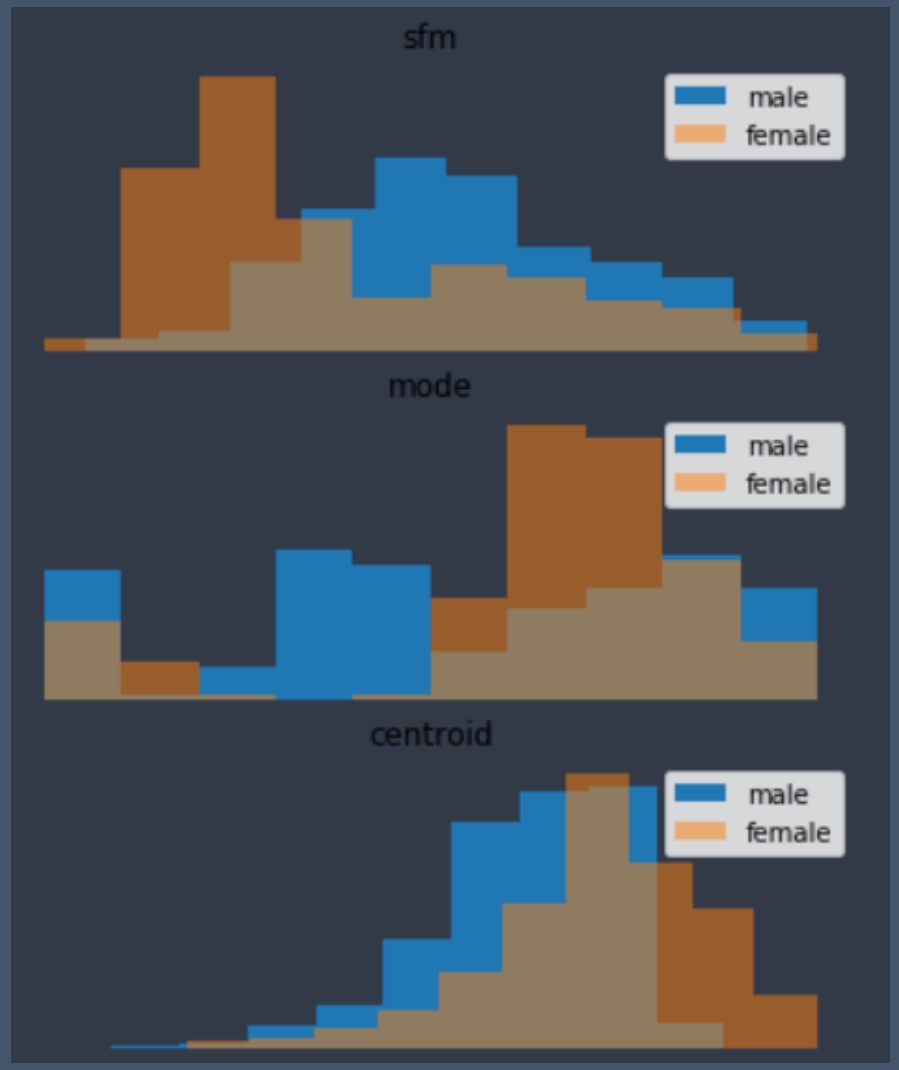
Objetivo do Trabalho

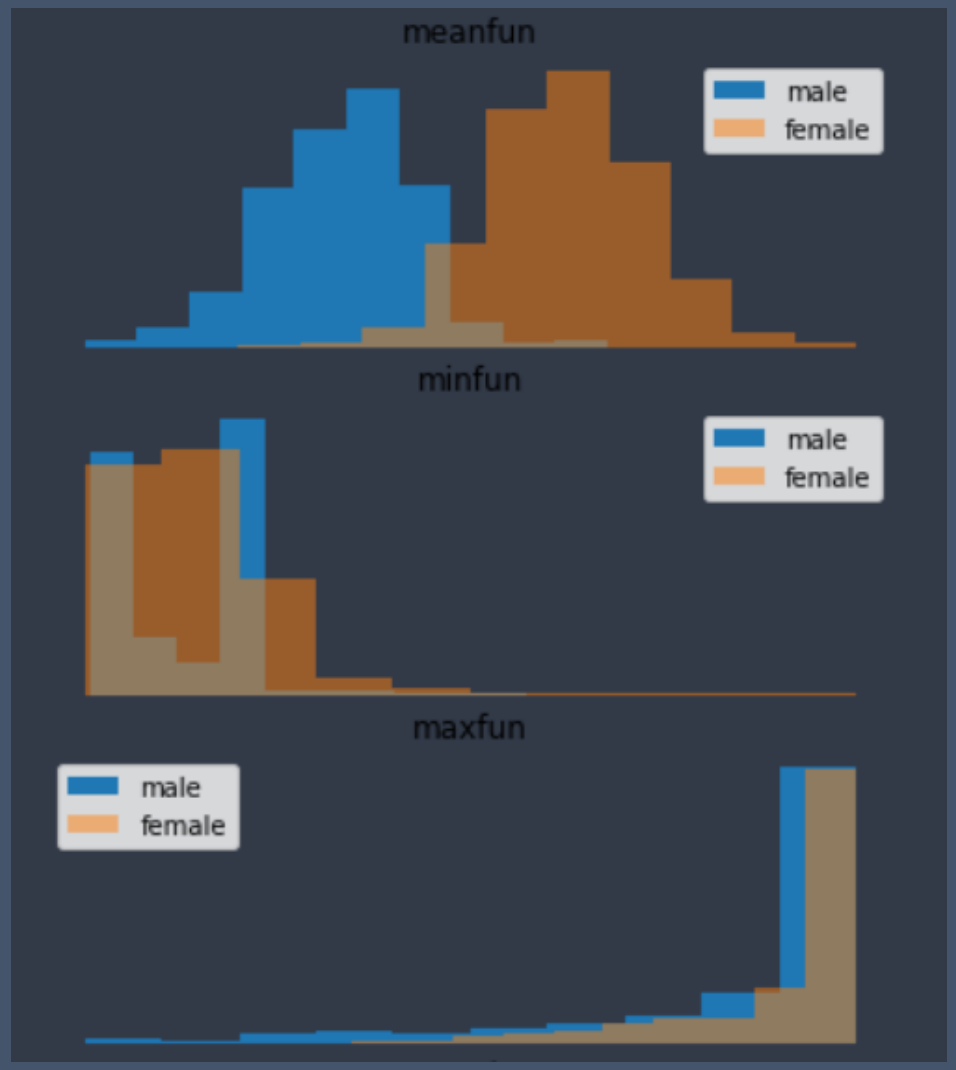
Mostrar que, se usarmos uma rede neural para resolver essa classificação, não será necessário usar todas as features, em outras palavras, existem modelos com desempenho equivalente que demandam menos processamento.

Distribuição das 20 Variáveis por Label

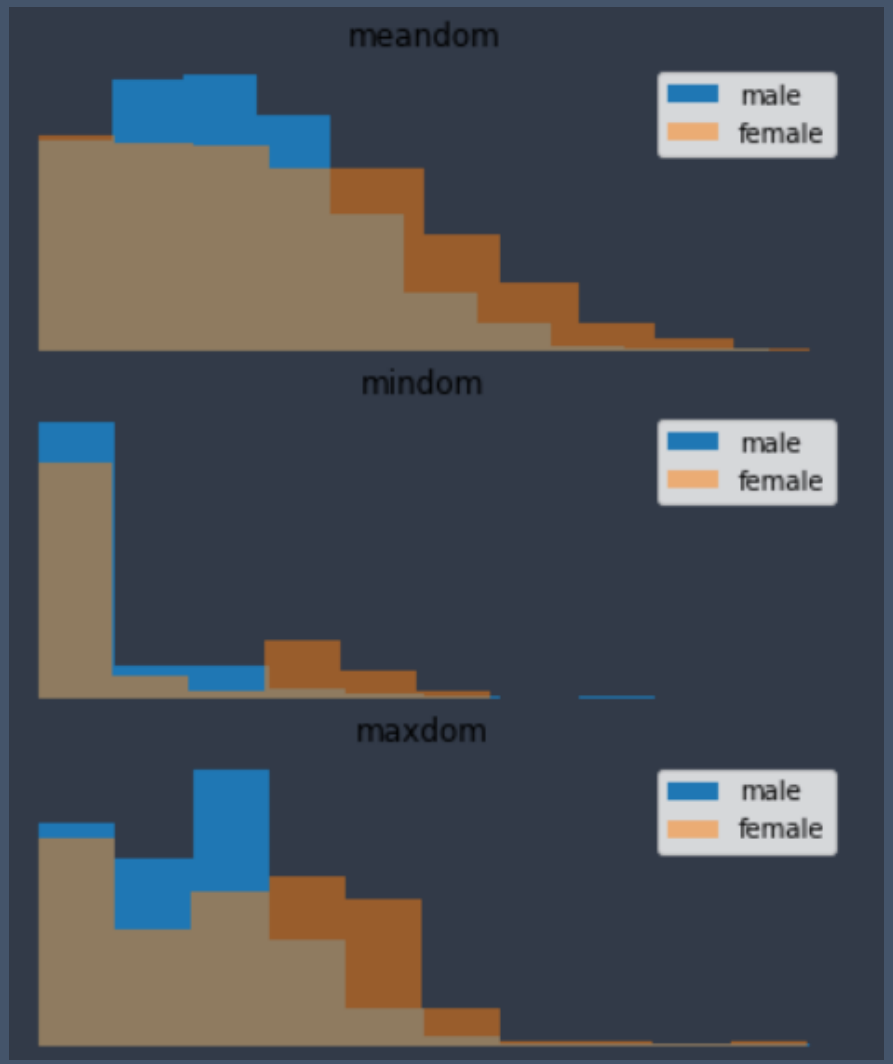


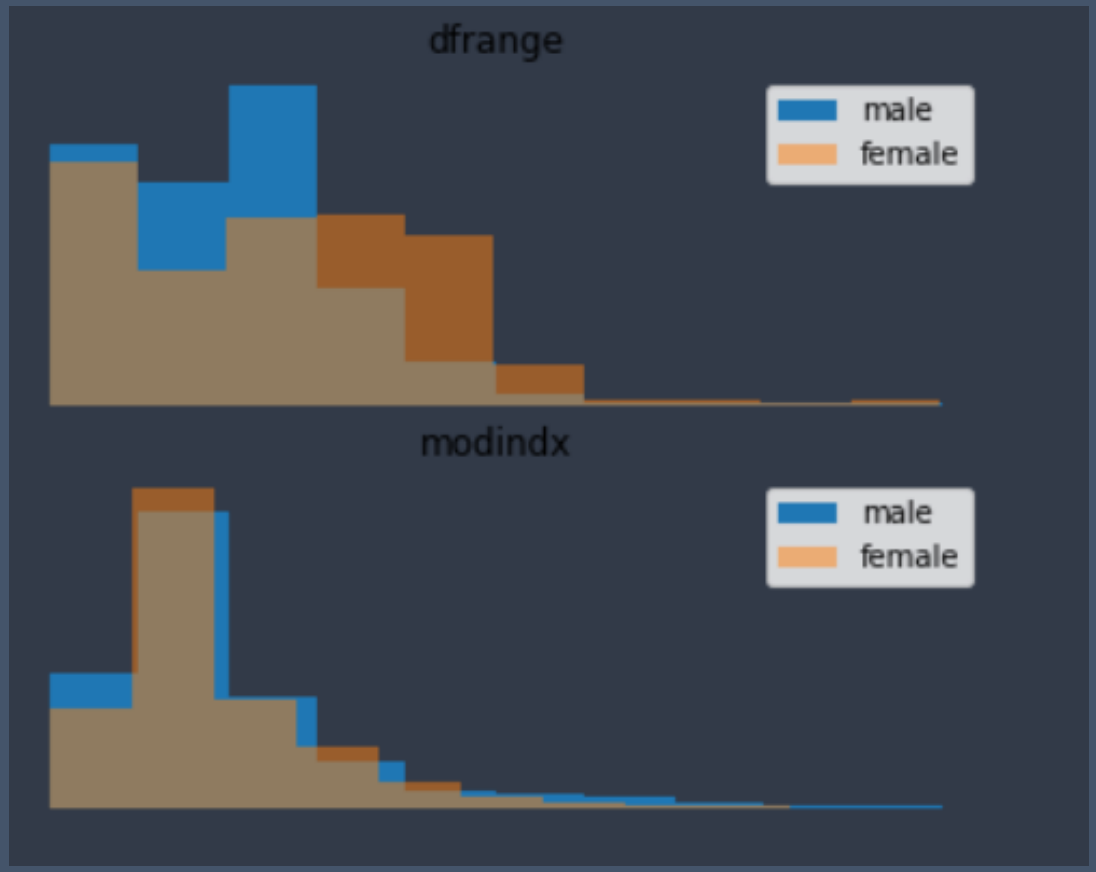
Distribuição das 20 Variáveis por Label





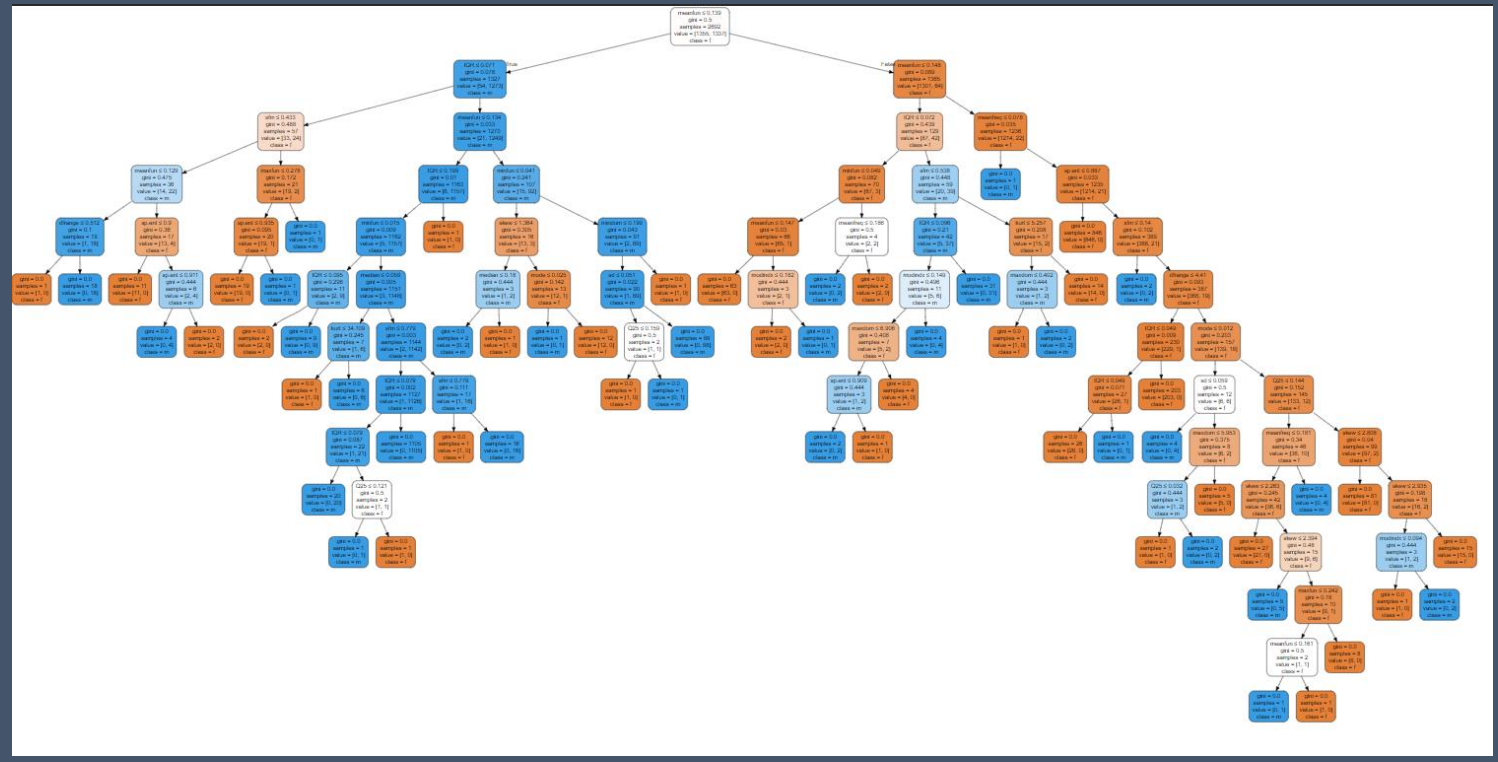
Distribuição das 20 Variáveis por Label





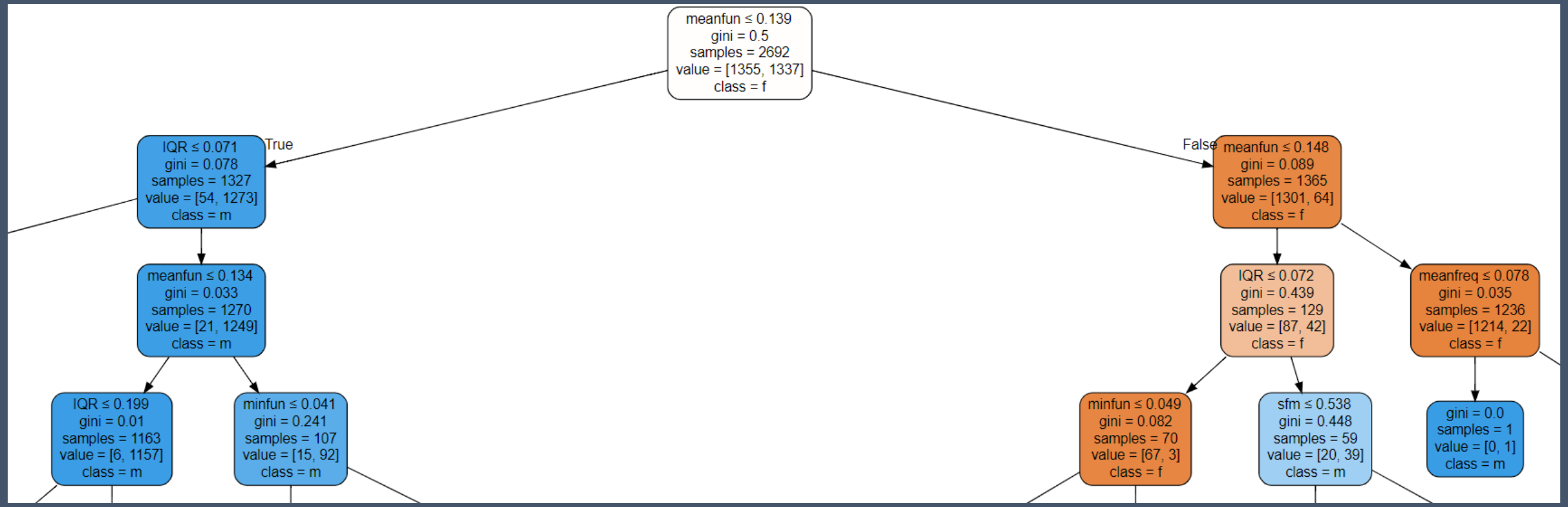
Como uma árvore de decisão dividiria o dataset?

Árvore Livre



Como uma árvore de decisão dividiria o dataset?

Árvore Livre



Como uma árvore de decisão dividiria o dataset?

Árvore Livre

In [37]:
```
pred = tree_free.predict(x_test)
```

In [38]:
```
print(confusion_matrix(y_test, pred))
```

```
[[221   8]
 [ 10 237]]
```

In [39]:
```
print(accuracy_score(y_test, pred))
```

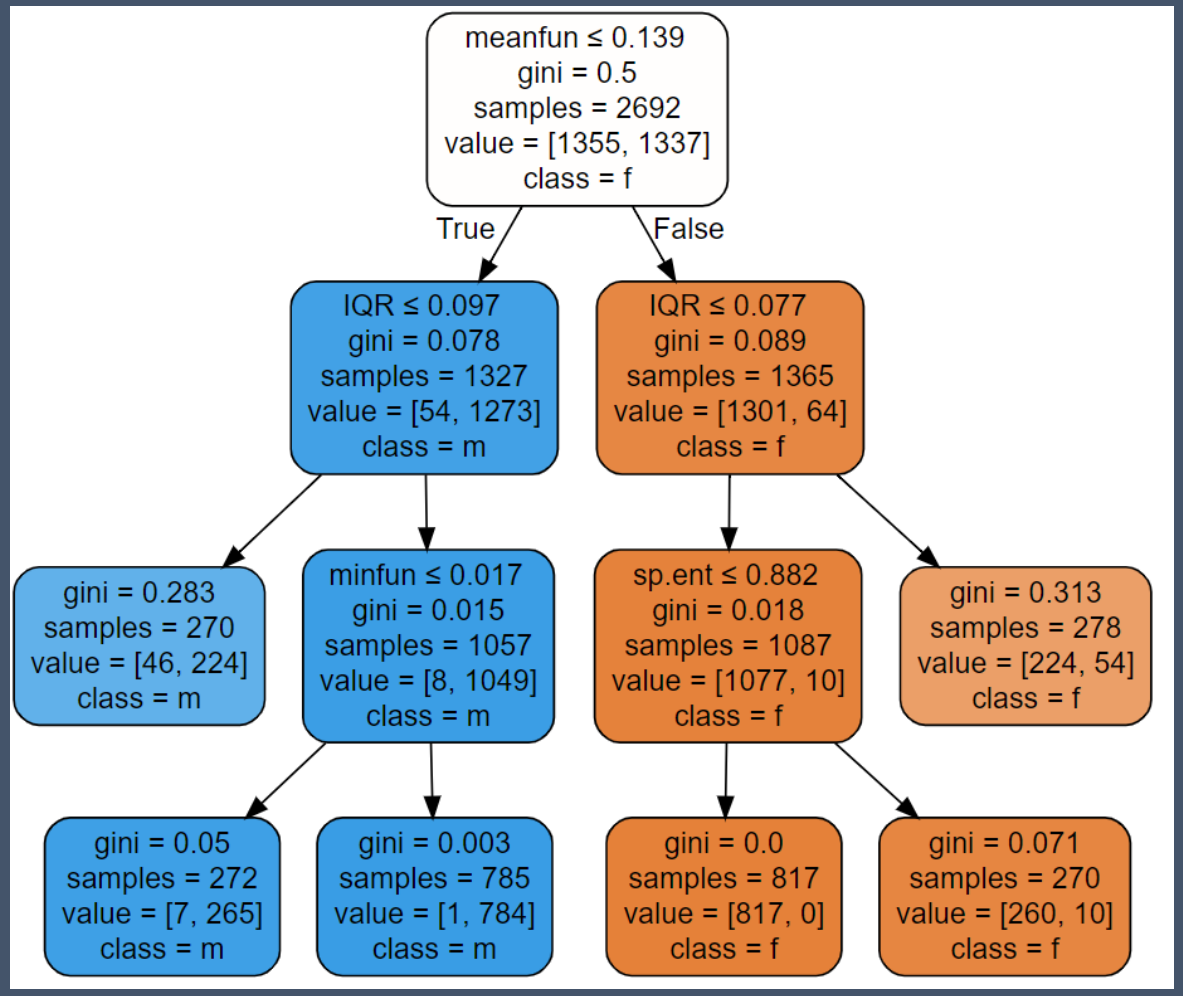
```
0.9621848739495799
```

In [40]:
```
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       229
           1       0.97      0.96      0.96       247

   micro avg       0.96      0.96      0.96       476
   macro avg       0.96      0.96      0.96       476
weighted avg       0.96      0.96      0.96       476
```

Como uma árvore de decisão dividiria o dataset?

Árvore Parametrizada (max_depth=3, min_samples_leaf=0.1)



Como uma árvore de decisão dividiria o dataset?

Árvore Parametrizada (max_depth=3, min_samples_leaf=0.1)

In [44]:
```
pred = tree_limited.predict(x_test)
```

In [45]:
```
print(confusion_matrix(y_test, pred))
```

```
[[218  11]
 [ 18 229]]
```

In [46]:
```
print(accuracy_score(y_test, pred))
```

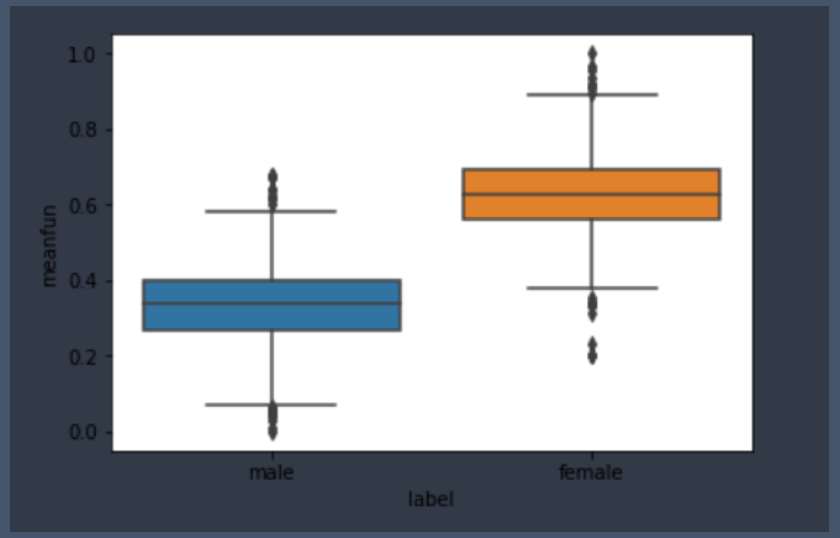
```
0.9390756302521008
```

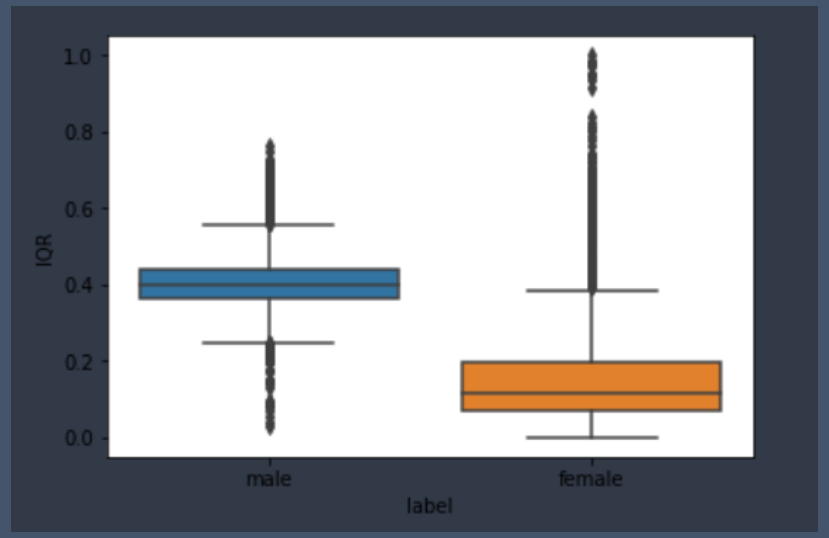
In [47]:
```
print(classification_report(y_test, pred))
```

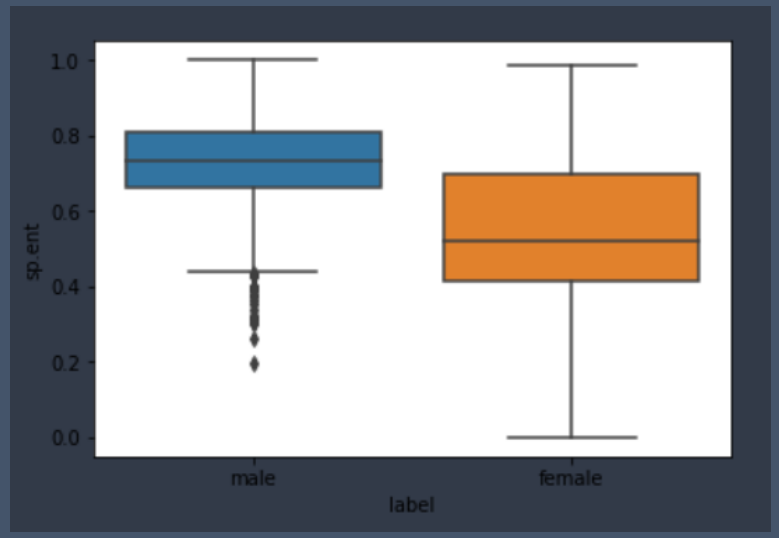
```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       229
           1       0.95      0.93      0.94       247

   micro avg       0.94      0.94      0.94       476
   macro avg       0.94      0.94      0.94       476
weighted avg       0.94      0.94      0.94       476
```

Como uma árvore de decisão dividiria o dataset?

Uso das variáveis “meanfun”, “IQR” e “sp.ent”

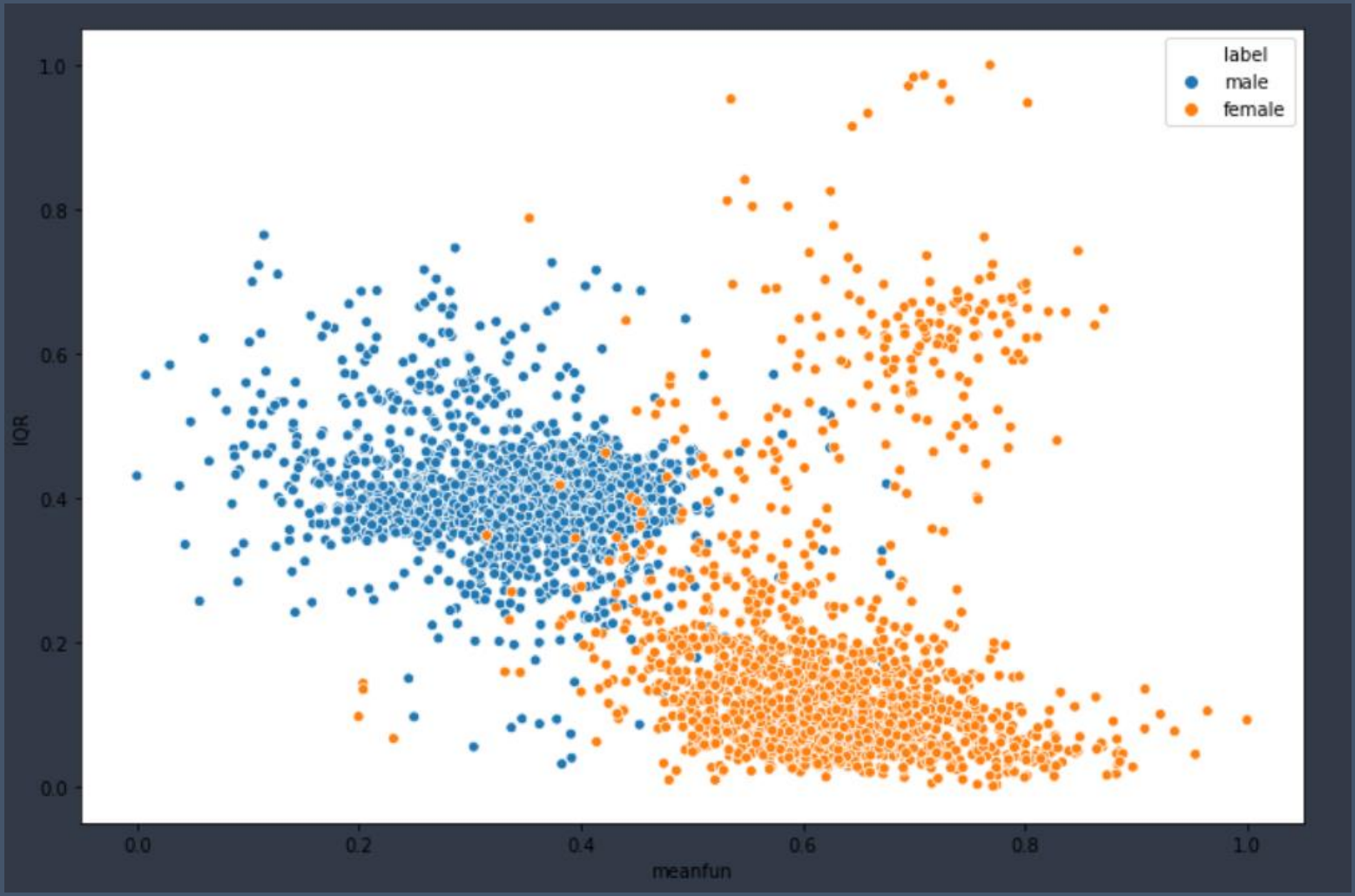






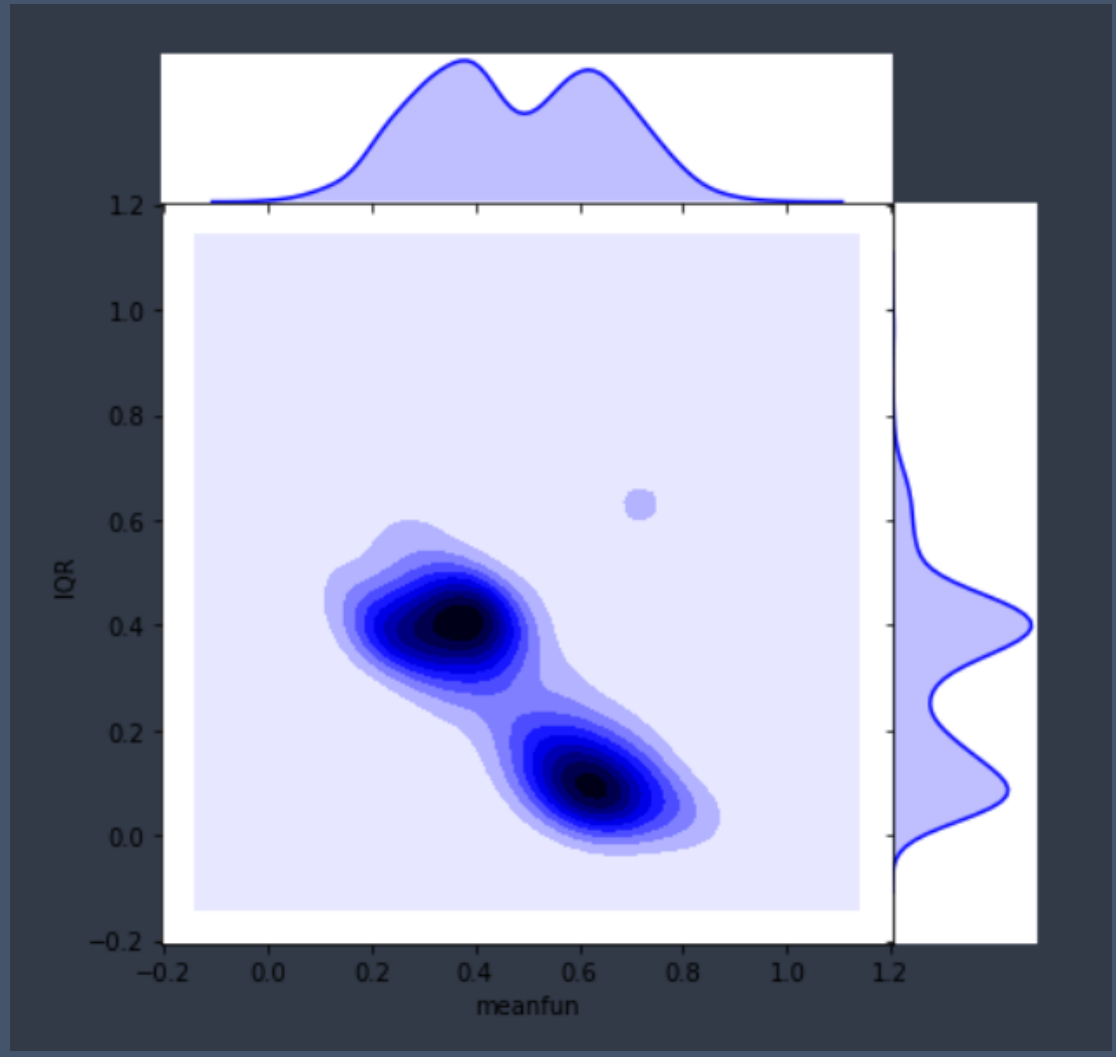
Como uma árvore de decisão dividiria o dataset?

“meanfun” x “IQR”



Como uma árvore de decisão dividiria o dataset?

"meanfun" x "IQR"

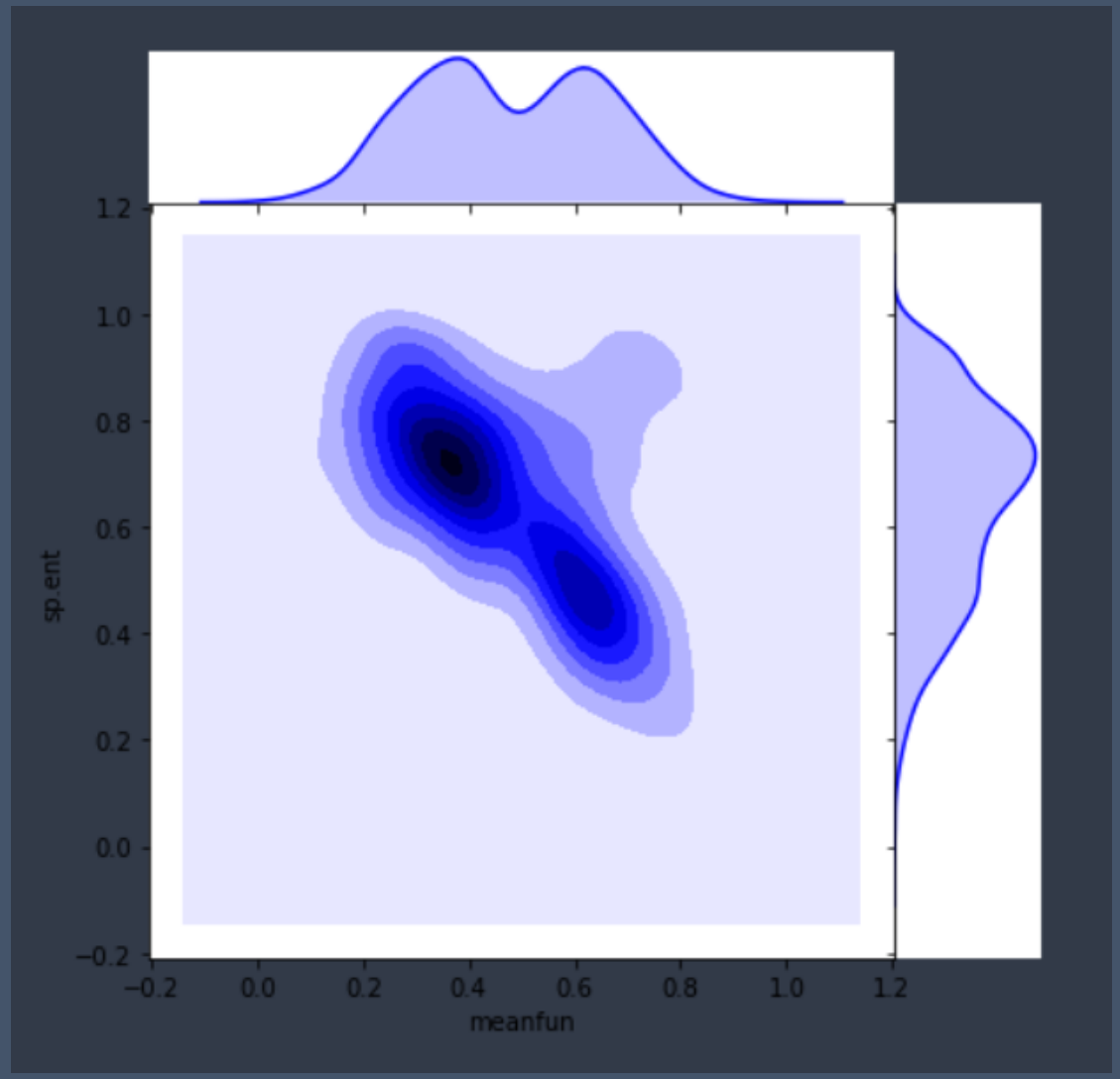


Como uma árvore de decisão dividiria o dataset?

"meanfun" x "sp.ent"

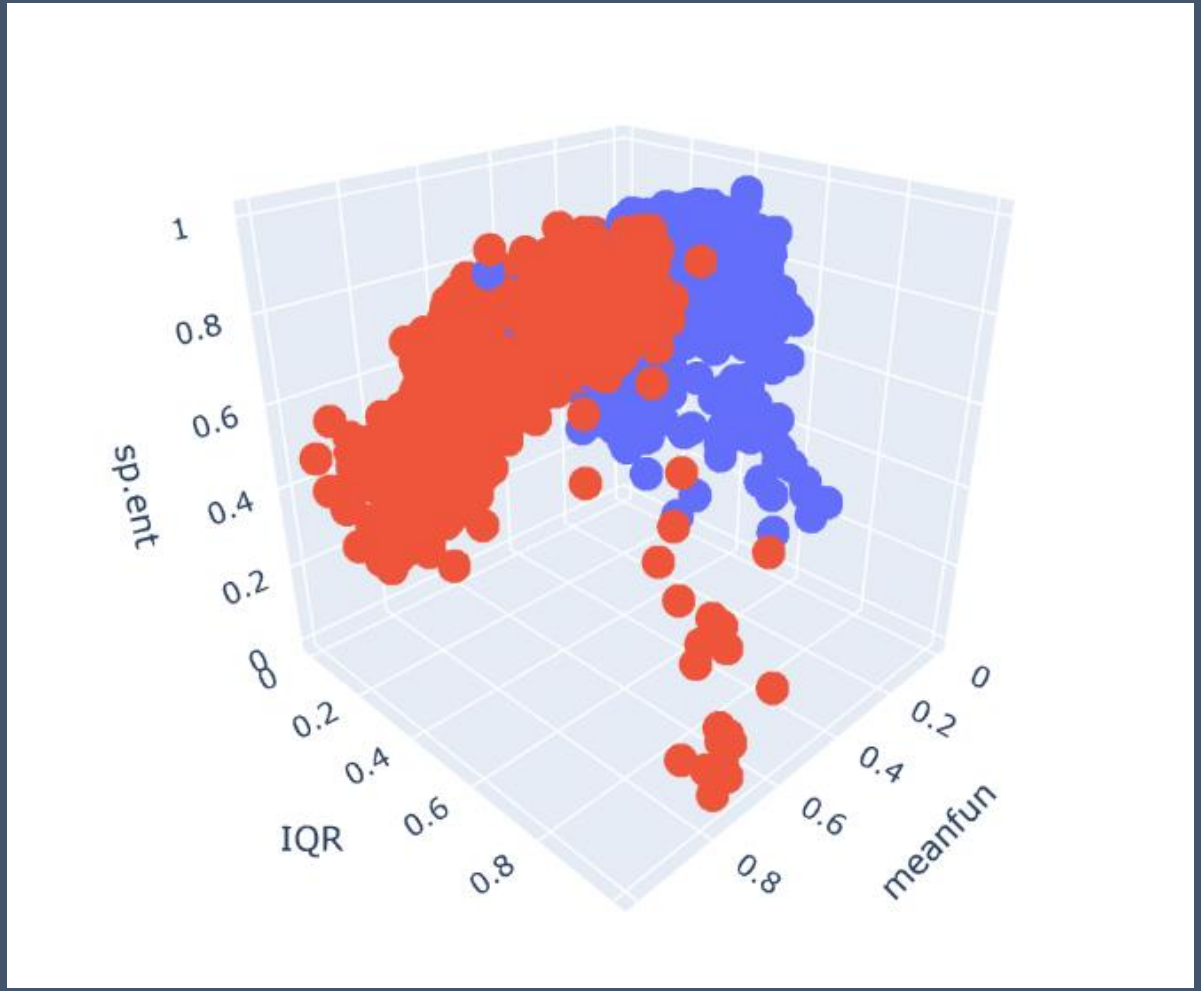
Como uma árvore de decisão dividiria o dataset?

“meanfun” x “sp.ent”



Como uma árvore de decisão dividiria o dataset?

“meanfun” x “IQR” x “sp.ent”



Método

Serão testados 4 modelos de redes neurais com as mesmas características, variando apenas a entrada

- Modelo com as 20 features disponíveis
- Modelo com 3 variáveis escolhidas
- Modelo com 3 principal components
- Modelo com a quantidade de principal components necessário para explicar, no mínimo, 95% da variação do dataset

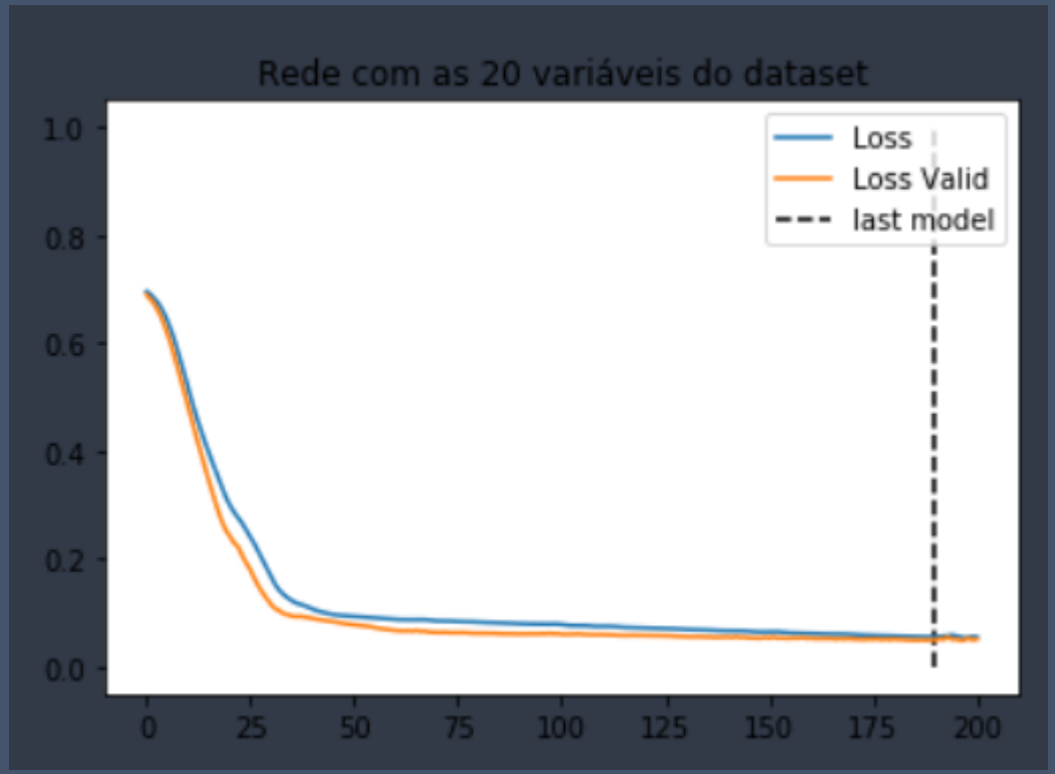
Todos serão testados com 10.000 amostras de tamanho 100 (sem reposição) com origem nos dados de teste (15%).

Rede Neural com as 20 Variáveis

Treino 70%
Validação 15%
Teste 15%

Early Stopping com tolerância de 10 épocas sem redução no loss da validação

```
Model(
  (conv1): Linear(in_features=20, out_features=32, bias=True)
  (conv2): Linear(in_features=32, out_features=32, bias=True)
  (conv3): Linear(in_features=32, out_features=1, bias=True)
  (relu): ReLU()
  (sigmoid): Sigmoid()
)
```



Rede Neural com as 20 Variáveis

In [65]:
```
print('TN-FP\nFN-TP')
confusion_matrix(y_test, pred.round().detach().numpy())
```

```
TN-FP
FN-TP


array([[222,   7],
       [  4, 243]], dtype=int64)
```

In [66]:
```
accuracy_score(y_test, pred.round().detach().numpy())
```

```
0.976890756302521
```

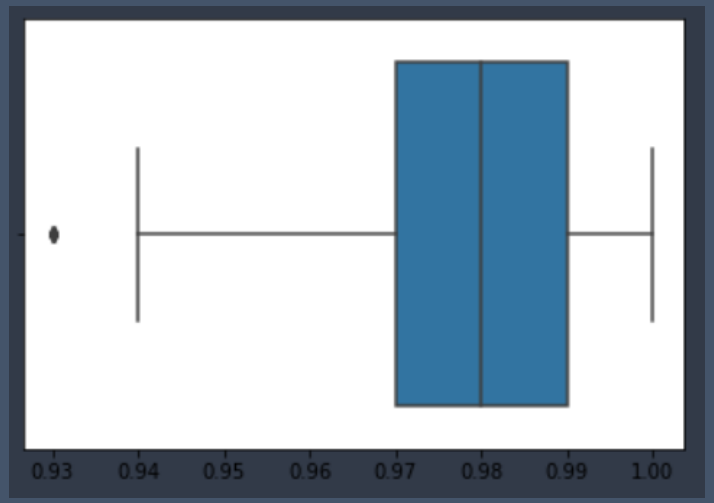
In [67]:
```
print(classification_report(y_test, pred.round().detach().numpy(),
                            target_names= ['female', 'male']))
```

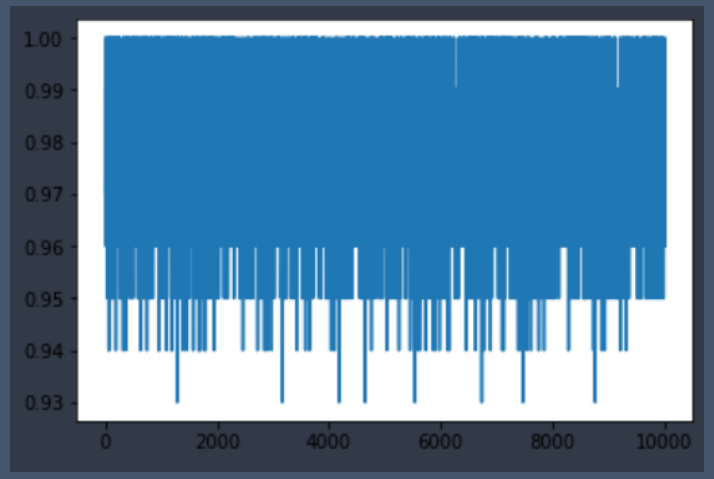
```
              precision    recall  f1-score   support

      female       0.98      0.97      0.98       229
        male       0.97      0.98      0.98       247

   micro avg       0.98      0.98      0.98       476
   macro avg       0.98      0.98      0.98       476
weighted avg       0.98      0.98      0.98       476
```

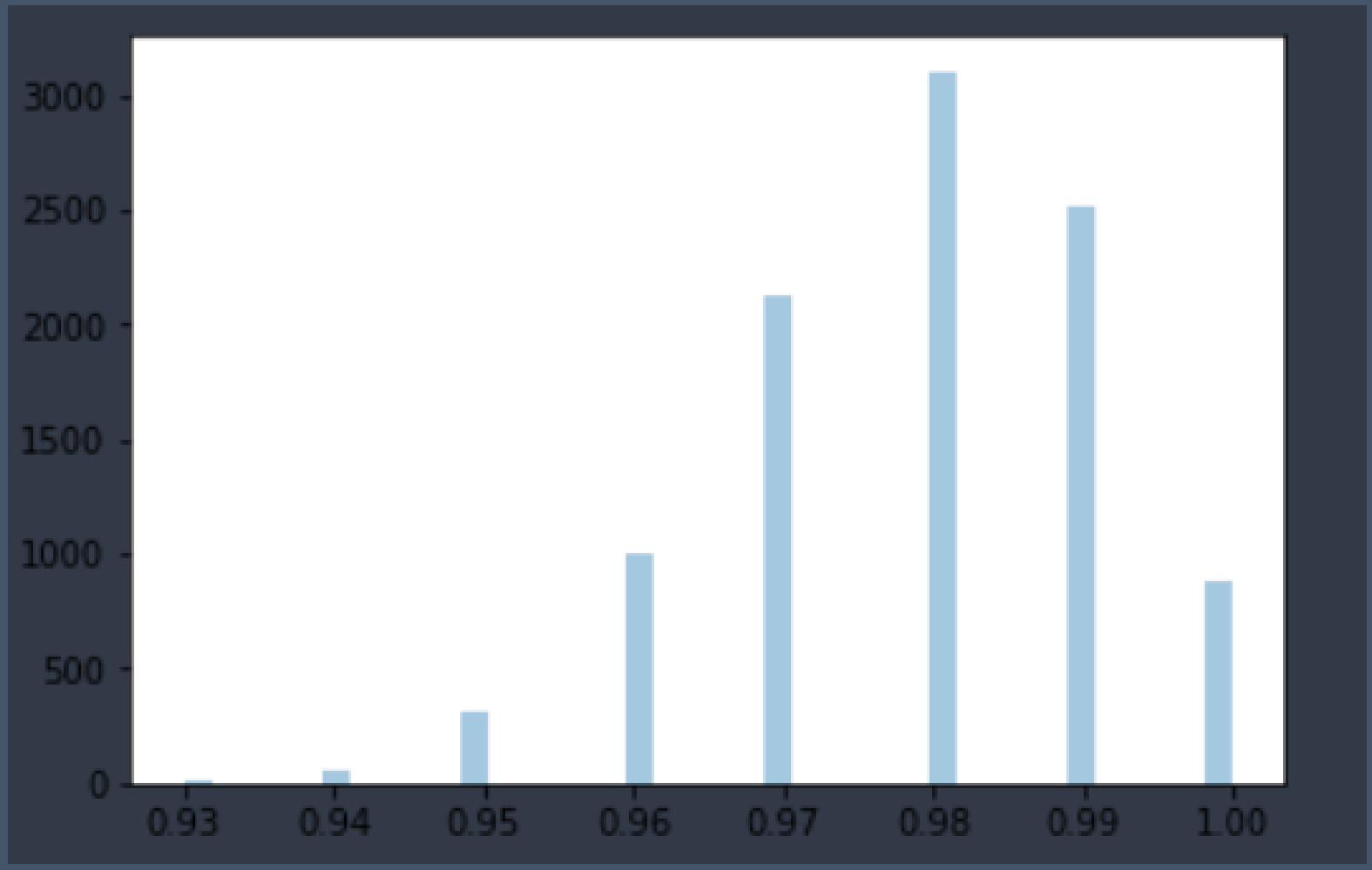
Rede Neural com as 20 Variáveis

8.356362852025466e+104 amostras diferentes de 100 elementos em 476

result_list = resultado_amostra(data, model, tam_amostra=100, quantidade_amostras=10000)





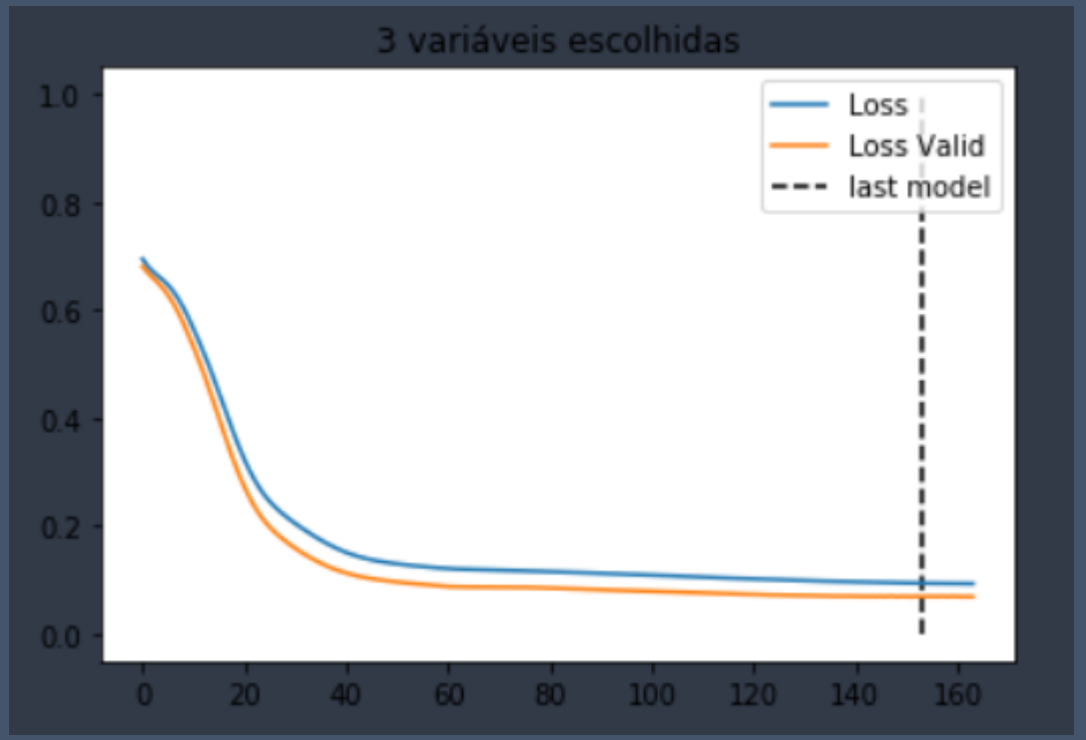


Rede Neural com as 3 Variáveis Escolhidas ('meanfun', 'IQR', 'sp.ent')

Treino 70%
Validação 15%
Teste 15%

Early Stopping com tolerância de 10 épocas sem redução no loss da validação

```
Model(
  (conv1): Linear(in_features=3, out_features=32,
bias=True)
  (conv2): Linear(in_features=32, out_features=32,
bias=True)
  (conv3): Linear(in_features=32, out_features=1,
bias=True)
  (relu): ReLU()
  (sigmoid): Sigmoid()
)
```



Rede Neural com as 3 Variáveis Escolhidas ('meanfun', 'IQR', 'sp.ent')

In [92]:
```
print('TN-FP\nFN-TP')
confusion_matrix(y_test, pred.round().detach().numpy())
```

```
TN-FP
FN-TP


array([[224,   5],
       [  4, 243]], dtype=int64)
```

In [93]:
```
accuracy_score(y_test, pred.round().detach().numpy())
```

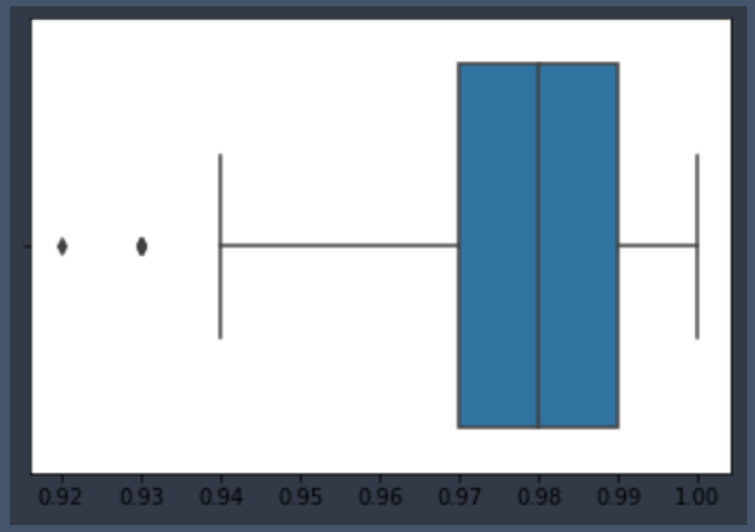
```
0.9810924369747899
```

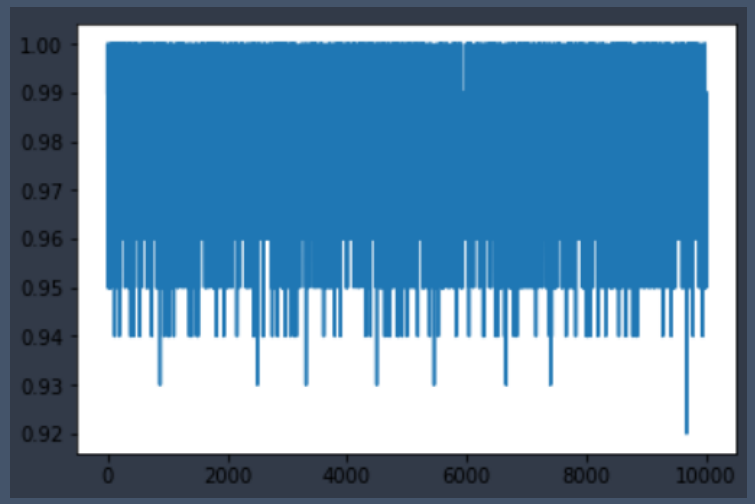
In [94]:
```
print(classification_report(y_test, pred.round().detach().numpy(),
                            target_names= ['female', 'male']))
```

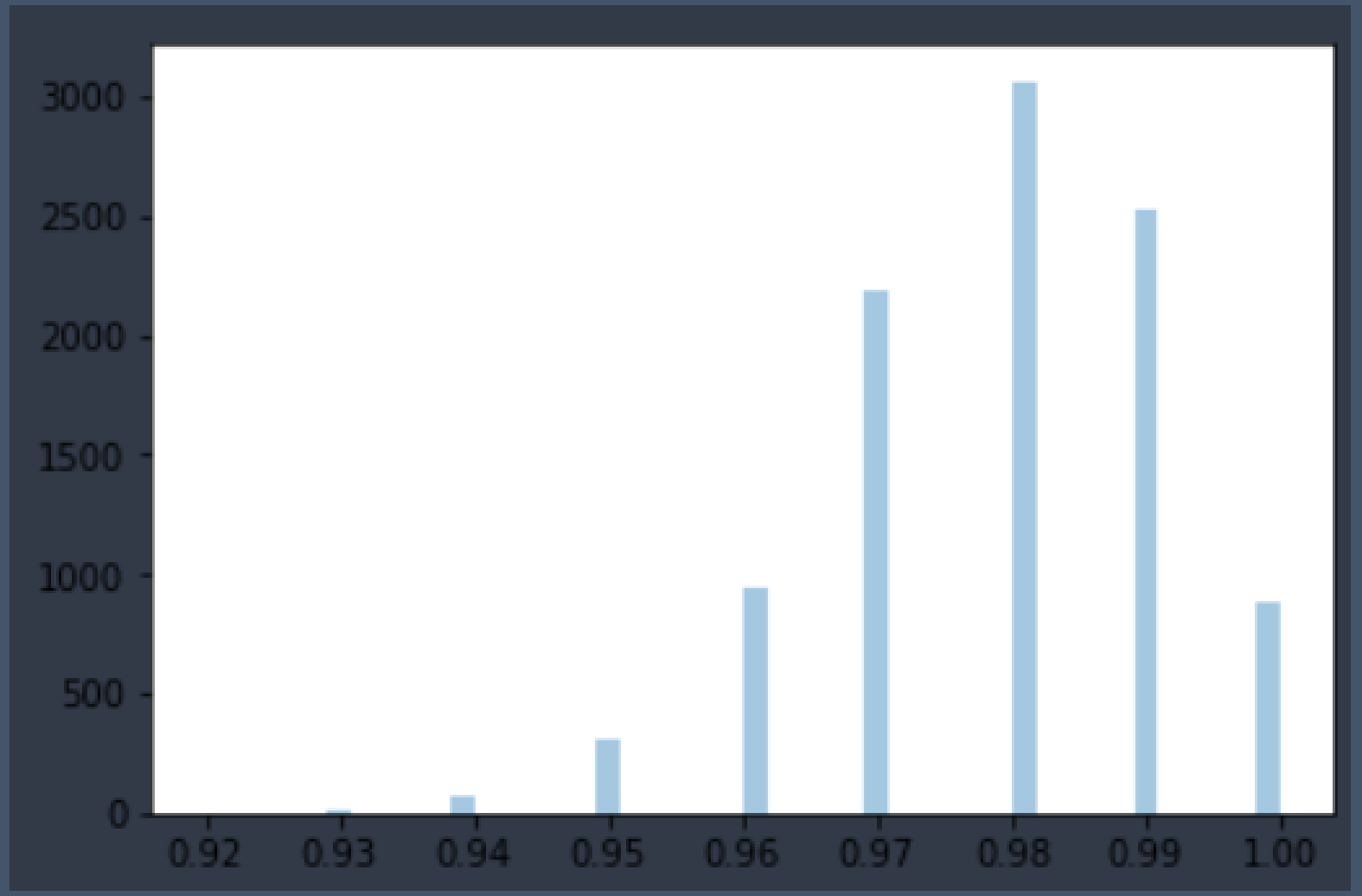
```
              precision    recall  f1-score   support

      female       0.98      0.98      0.98       229
        male       0.98      0.98      0.98       247

   micro avg       0.98      0.98      0.98       476
   macro avg       0.98      0.98      0.98       476
weighted avg       0.98      0.98      0.98       476
```

Rede Neural com as 3 Variáveis Escolhidas ('meanfun', 'IQR', 'sp.ent')

result_list = resultado_amostra(data, model, tam_amostra=100, quantidade_amostras=10000)







Rede Neural com 3 Principal Components (PCA)

In [103]:
```
pca = PCA(n_components=3)
components = pca.fit_transform(df.iloc[:,:-1])
```

In [104]:
```
pca.explained_variance_ratio_
```

```
array([0.50779229, 0.12190298, 0.08656245])
```

In [105]:
```
len(pca.explained_variance_ratio_)
```

```
3
```

In [106]:
```
np.cumsum(pca.explained_variance_ratio_)[-1]
```

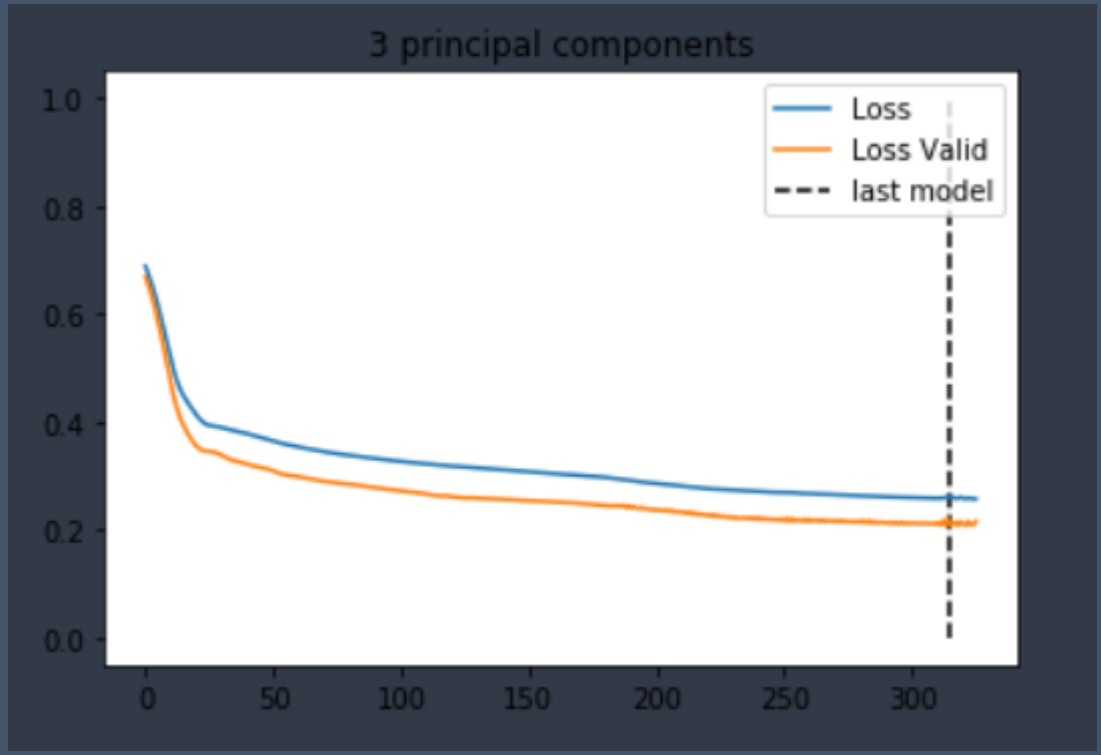
```
0.7162577187262075
```

Rede Neural com 3 Principal Components (PCA)

Treino 70%
Validação 15%
Teste 15%

Early Stopping com tolerância de 10 épocas sem redução no loss da validação

Model(
  (conv1): Linear(in_features=3, out_features=32, bias=True)
  (conv2): Linear(in_features=32, out_features=32, bias=True)
  (conv3): Linear(in_features=32, out_features=1, bias=True)
  (relu): ReLU()
  (sigmoid): Sigmoid()
)



Rede Neural com 3 Principal Components (PCA)

In [135]:
```
print('TN-FP\nFN-TP')
confusion_matrix(y_test, pred.round().detach().numpy())
```

```
TN-FP
FN-TP


array([[190,  39],
       [ 18, 229]], dtype=int64)
```

In [136]:
```
accuracy_score(y_test, pred.round().detach().numpy())
```

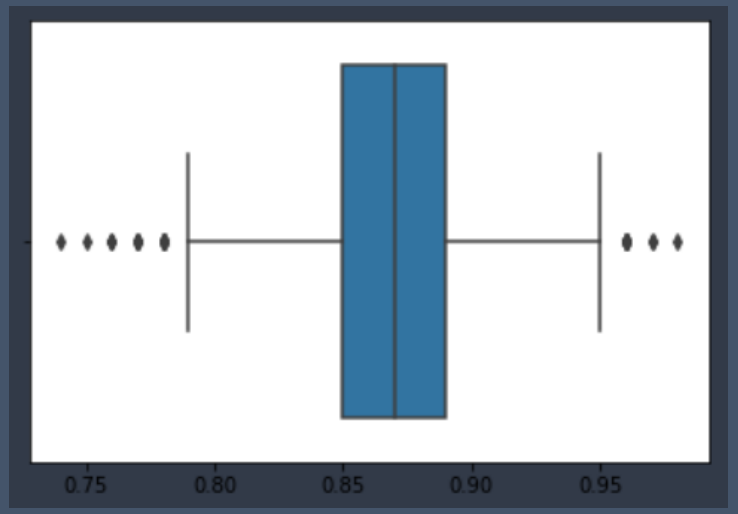
```
0.8802521008403361
```

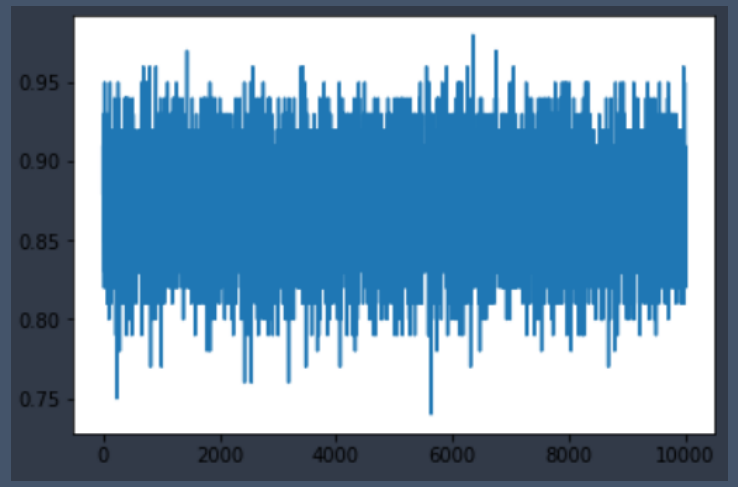
In [137]:
```
print(classification_report(y_test, pred.round().detach().numpy(),
                            target_names= ['female', 'male']))
```

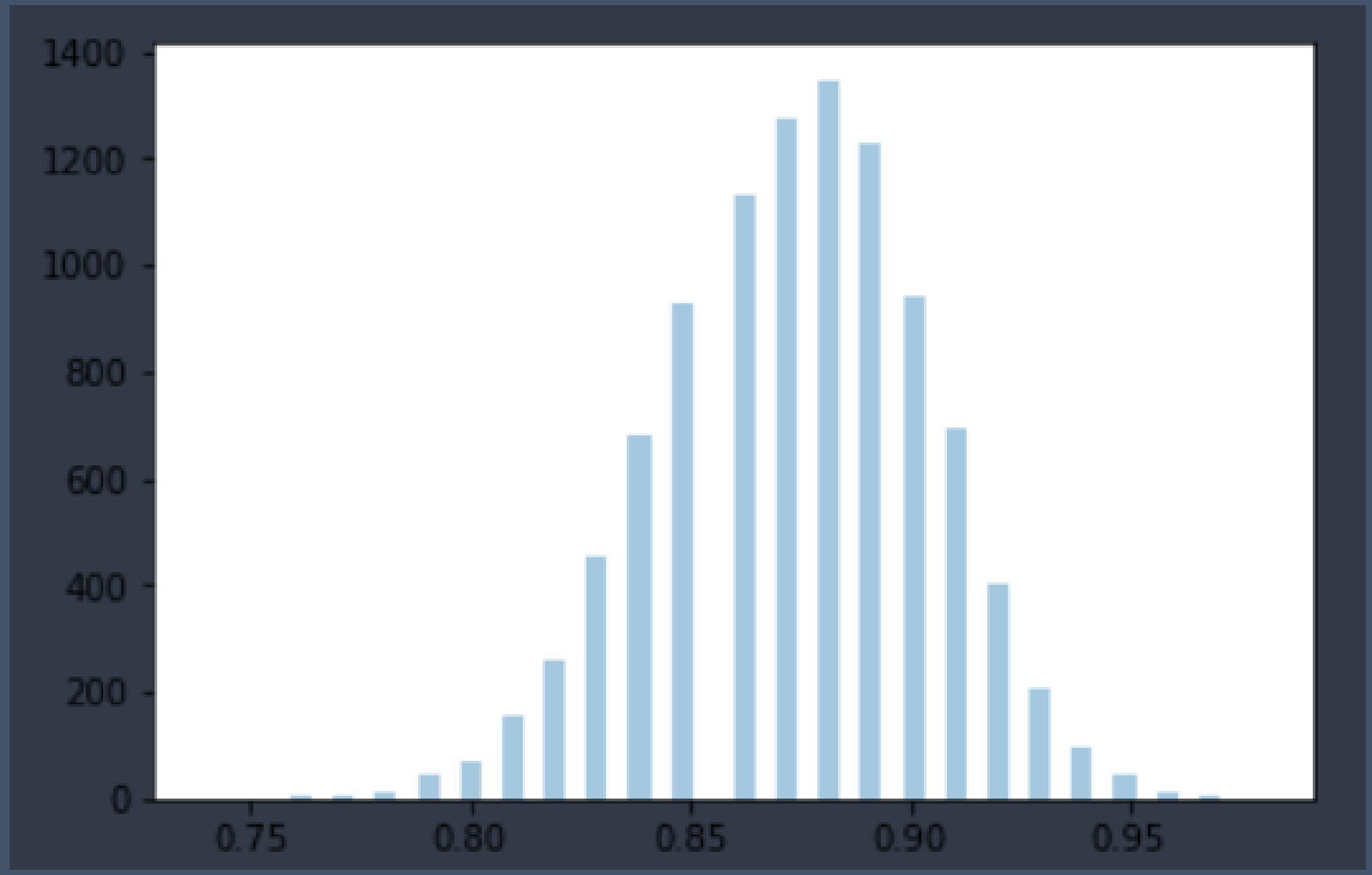
```
              precision    recall  f1-score   support

      female       0.91      0.83      0.87       229
        male       0.85      0.93      0.89       247

   micro avg       0.88      0.88      0.88       476
   macro avg       0.88      0.88      0.88       476
weighted avg       0.88      0.88      0.88       476
```

Rede Neural com 3 Principal Components (PCA)

result_list = resultado_amostra(data, model, tam_amostra=100, quantidade_amostras=10000)







Usando uma Quantidade de Componentes que Explique 95% da variação do Dataset

In [147]:
```
pca = PCA(n_components=.95)
components = pca.fit_transform(df.iloc[:,:-1])
```

In [148]:
```
pca.explained_variance_ratio_
```

```
array([0.50779229, 0.12190298, 0.08656245, 0.06103532, 0.05349789,
       0.03874441, 0.03069906, 0.0255996 , 0.02317785, 0.01706343])
```

In [149]:
```
len(pca.explained_variance_ratio_)
```

```
10
```

In [150]:
```
np.cumsum(pca.explained_variance_ratio_)[-1]
```

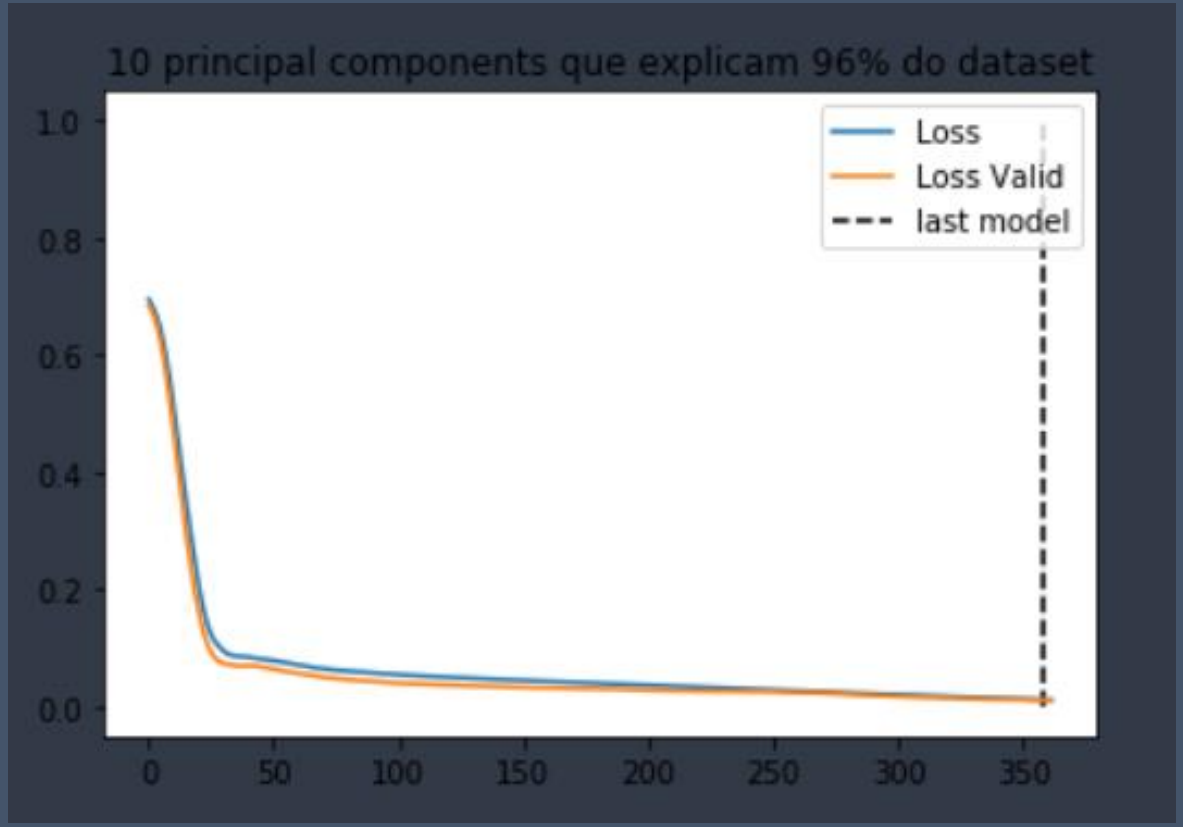
```
0.9660752899550968
```

Usando uma Quantidade de Componentes que Explique 95% da variação do Dataset

Treino 70%
Validação 15%
Teste 15%

Early Stopping com tolerância de 10 épocas sem redução no loss da validação

```
Model(
  (conv1): Linear(in_features=10, out_features=32, bias=True)
  (conv2): Linear(in_features=32, out_features=32, bias=True)
  (conv3): Linear(in_features=32, out_features=1, bias=True)
  (relu): ReLU()
  (sigmoid): Sigmoid()
)
```



Usando uma Quantidade de Componentes que Explique 95% da variação do Dataset

In [177]:
```
print('TN-FP\nFN-TP')
confusion_matrix(y_test, pred.round().detach().numpy())
```

```
TN-FP
FN-TP


array([[225,   4],
       [  7, 240]], dtype=int64)
```

In [178]:
```
accuracy_score(y_test, pred.round().detach().numpy())
```

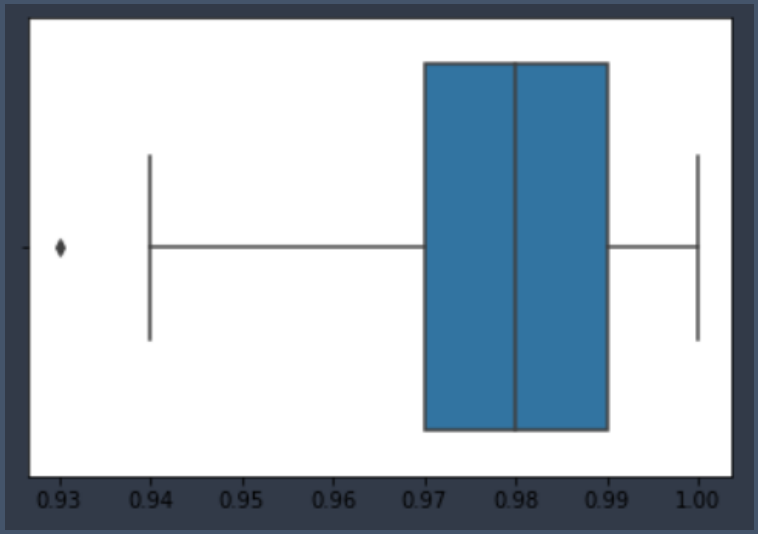
```
0.976890756302521
```

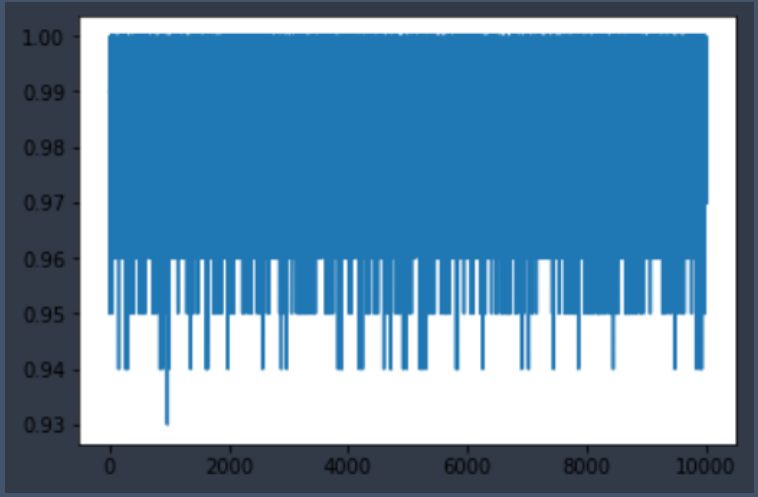
In [179]:
```
print(classification_report(y_test, pred.round().detach().numpy(),
                            target_names= ['female', 'male']))
```

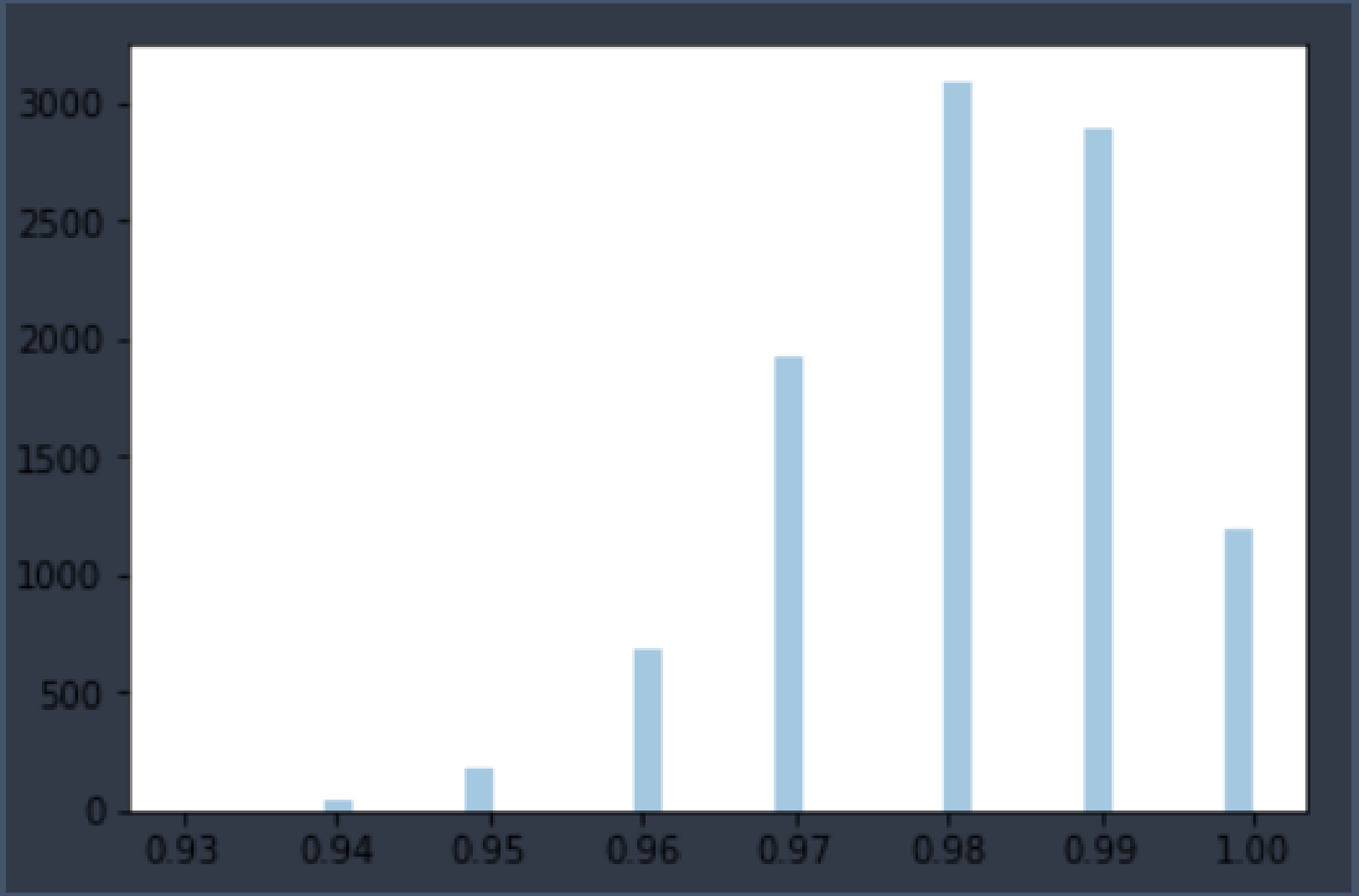
```
              precision    recall  f1-score   support

      female       0.97      0.98      0.98       229
        male       0.98      0.97      0.98       247

   micro avg       0.98      0.98      0.98       476
   macro avg       0.98      0.98      0.98       476
weighted avg       0.98      0.98      0.98       476
```

Usando uma Quantidade de Componentes que Explique 95% da variação do Dataset

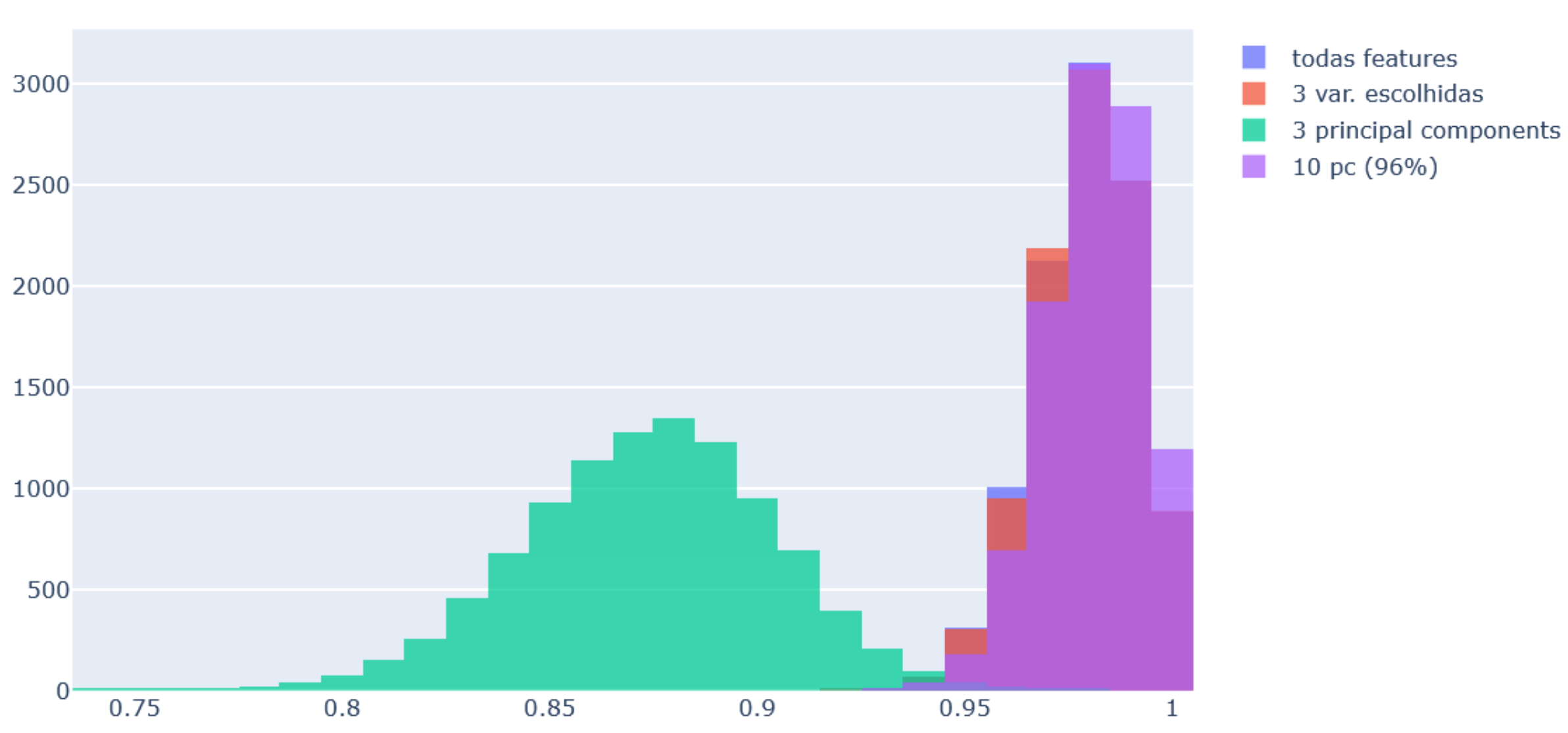
result_list = resultado_amostra(data, model, tam_amostra=100, quantidade_amostras=10000)



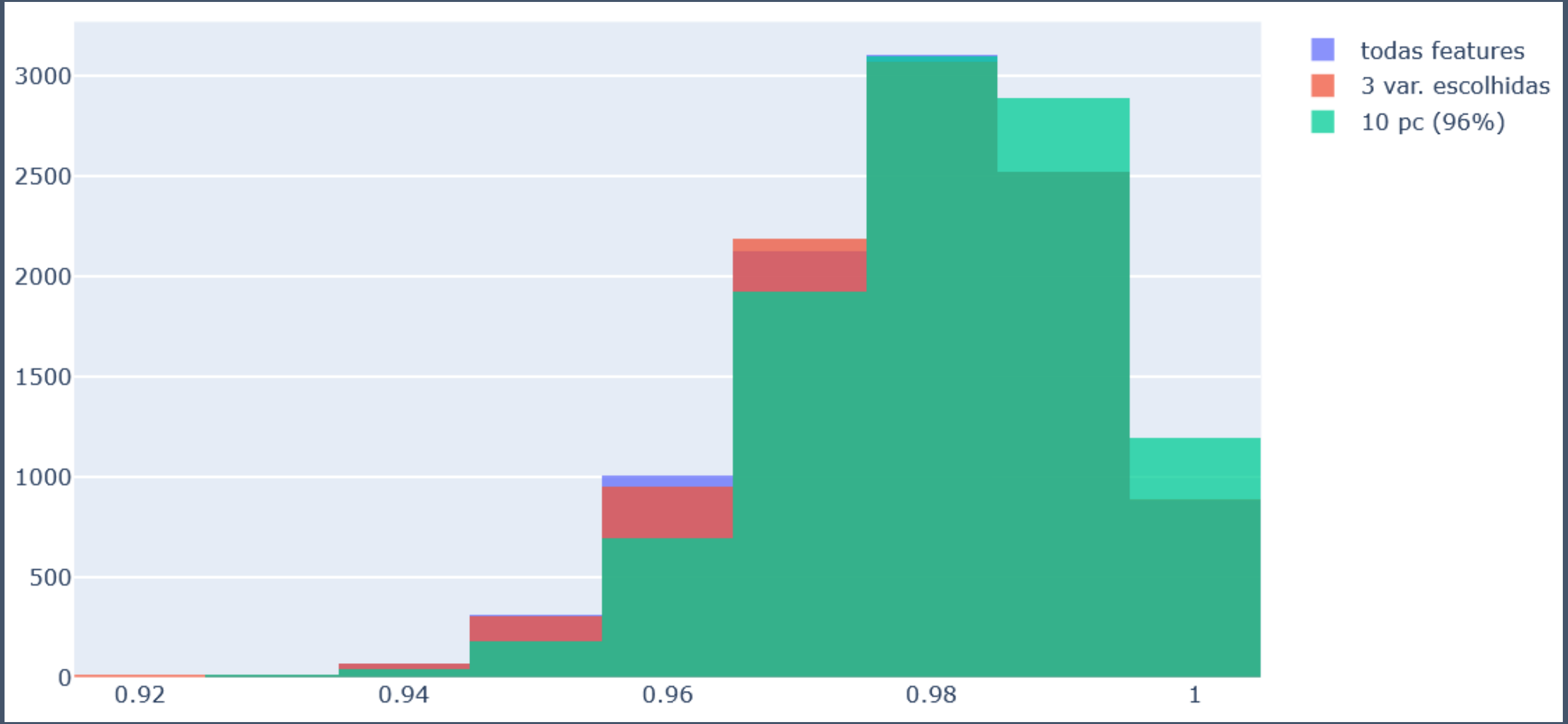




Comparando os Modelos



Comparando os Modelos



Conclusão

De fato, não há prejuízo em eliminar outras features ou fazer redução de dimensionalidade, desde que as features escolhidas representem bem a divisão que se deseja fazer no dataset e a redução de dimensionalidade não elimine muito da variação dos dados.

Mini Projetos

- Transfer Learning: https://github.com/omboido/transflearn
- Sentiment Analysis: https://github.com/hmaas00/mini_projeto_sentiment_analysis
- Style Transfer: https://github.com/omboido/styletransfer

Fim