In [320]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import stats
import seaborn
import sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve

# roc curve and auc
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

from collections import namedtuple

from sklearn.metrics import f1_score
from sklearn.metrics import auc
```

In [87]:
```
df = pd.read_csv("./data/dados_voz_genero.csv")
```

IA048 – Aprendizado de Máquina

Exercícios de Fixação de Conceitos (EFC) 2 – 2s2020

Classificação binária

Você dispõe de um conjunto de dados contendo 3168 amostras rotuladas. Cada amostra é descrita por 19 atributos acústicos extraídos de trechos gravados de voz, considerando a faixa de frequências de 0 a 280 Hz. A última coluna corresponde ao rótulo associado a cada padrão, sendo igual a '1' para o gênero masculino, e '0' para o gênero feminino.

a. Faça uma análise das características dos atributos de entrada considerando os respectivos histogramas e as medidas de correlação entre eles.

b. Construa, então, o modelo de regressão logística para realizar a classificação dos padrões. Para isso, reserve uma parte dos dados (e.g., 20%) para validação, usando todas as demais amostras para o treinamento do modelo. Pensem na pertinência e na possibilidade de realizar algum pré-processamento nos dados (e.g., normalização).

Apresente e discuta os seguintes resultados com relação ao conjunto de validação:

- A curva ROC;
- A curva de evolução da F -medida em função do threshold de decisão

c. Indique qual seria o valor mais adequado para o threshold de decisão e por quê. Empregando, então, esse threshold, obtenha a matriz de confusão e a acurácia do classificador para o conjunto de validação. Comente os resultados obtidos.

Classificação multi-classe

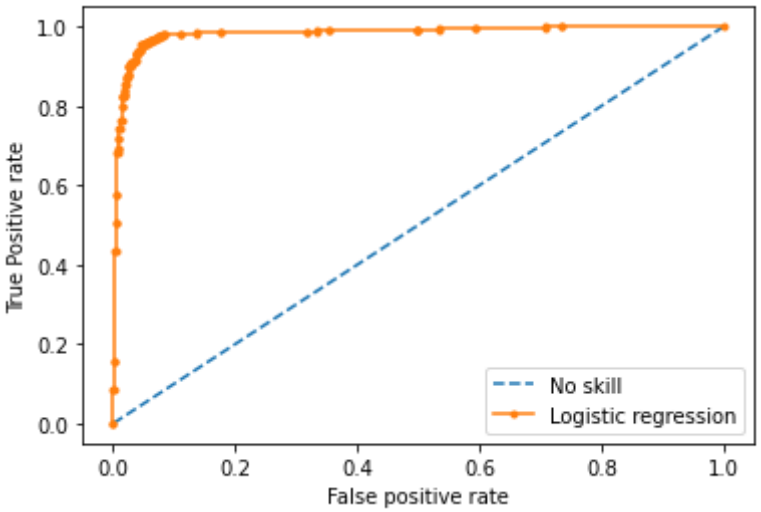
asdf

In [249]:
```
def t_test(a, b):
    N = len(a)
    #For unbiased max likelihood estimate we have to divide the var by N-1,
    var_a = a.var(ddof=1)
    var_b = b.var(ddof=1)

    #std deviation
    s = np.sqrt((var_a + var_b)/2)

    ## Calculate the t-statistics
    t = (a.mean() - b.mean())/(s*np.sqrt(2/N))

    ## Compare with the critical t-value
    #Degrees of freedom
    df = 2*N - 2

    #p-value after comparison with the t
    p = 1 - stats.t.cdf(t,df=df)
    return p

def histogram_intersection(a, b):
    return t_test(a, b)

def plot_histogram(df, column='dfrange'):
    mapping = { 0: 'Female', 1: 'Male' }
    seaborn.histplot(df.replace({ 'label': mapping }), x=column, alpha=0.5,
    return abs(1 - t_test(df[df['label'] == 1][column], df[df['label'] == 0

def plot_matrix(df):
    correlation = df.corr().abs()
    np.fill_diagonal(correlation.values, 0)

    figure = plt.figure(figsize=(24, 24))
    plt.matshow(correlation, fignum=figure.number)

    xticks = plt.xticks(range(df.shape[1]), df.columns, fontsize=14, rotati
    yticks = plt.yticks(range(df.shape[1]), df.columns, fontsize=14)
    colorbar = plt.colorbar()

    serie = correlation.unstack().sort_values(ascending=False, kind="quicks
    highest_correlations = serie.drop_duplicates().where(serie > 0.8).dropn

    return highest_correlations
```

In [343]:
```
def logistic_regression(df, test_size=0.2):
    model = LogisticRegression()
    X = df[['meanfun']]
    y = df['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=tes
    model.fit(X_train, y_train)

    return model.score(X_test, y_test)

def roc_score(df):
    X = df[['meanfun']]
    y = df['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5
    ns_probs = [0 for _ in range(len(y_test))]
    model = LogisticRegression(solver='lbfgs')
    model.fit(X_train, y_train)

    lr_probs = model.predict_proba(X_test)
    lr_probs = lr_probs[:, 1]
    ns_auc = roc_auc_score(testy, ns_probs)
    lr_auc = roc_auc_score(testy, lr_probs)
    Score = namedtuple('Score', ['none', 'logistic_regression'])

    return Score(ns_auc, lr_auc)

def roc_plot(df):
    X = df[['meanfun']]
    y = df['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5

    model = LogisticRegression(solver='lbfgs')
    model.fit(X_train, y_train)

    ns_probs = [0 for _ in range(len(y_test))]
    ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)

    lr_probs = model.predict_proba(X_test)
    lr_probs = lr_probs[:, 1]
    lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)

    plt.plot(ns_fpr, ns_tpr, linestyle='--', label='No skill')
    plt.plot(lr_fpr, lr_tpr, marker='.', label='Logistic regression')
    plt.xlabel('False positive rate')
    plt.ylabel('True Positive rate')
    plt.legend()
    plt.show()

    ns_auc = roc_auc_score(y_test, ns_probs)
    lr_auc = roc_auc_score(y_test, lr_probs)
    Score = namedtuple('Score', ['none', 'logistic_regression'])

    return Score(ns_auc, lr_auc)
```

```
print(roc_score(df))
roc_plot(df)
```

```
Score(none=0.5, logistic_regression=0.9835209612161285)
```

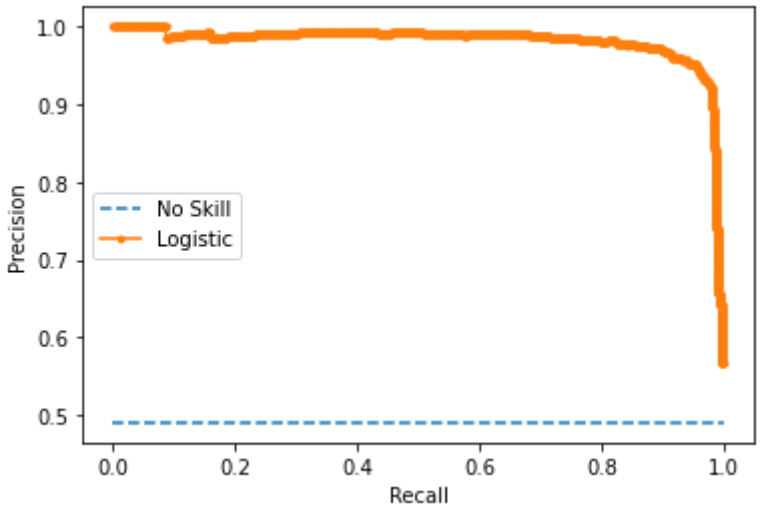


Out[343]:
```
Score(none=0.5, logistic_regression=0.9835209612161285)
```

In [251]:
```
plot_matrix(df)
```

Out[251]:
```
maxdom    dfrange     0.999838
kurt      skew        0.977020
centroid  median      0.925445
Q25       centroid    0.911416
sd        IQR         0.874660
IQR       Q25         0.874189
sfm       sp.ent      0.866411
sd        Q25         0.846931
sfm       sd          0.838086
label     meanfun     0.833921
meandom   maxdom      0.812838
          dfrange     0.811304
dtype: float64
```

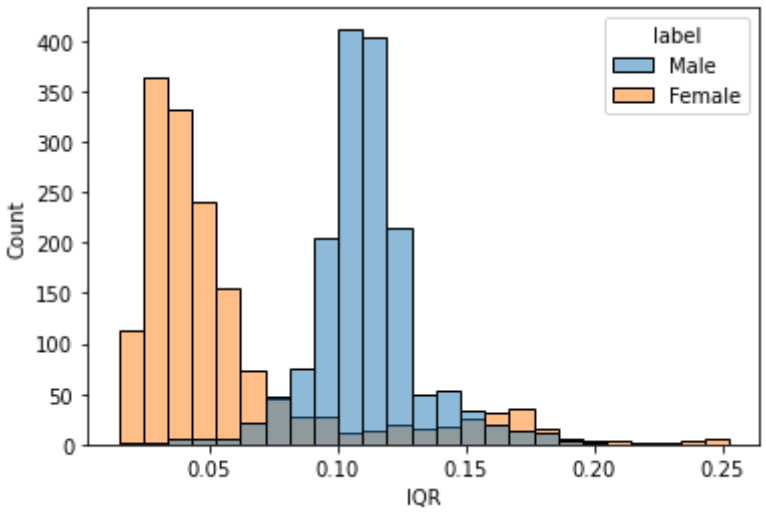
In [347]:
```
def precision(df):
    trainX, testX, trainy, testy = train_test_split(X, y, test_size=0.5, ra
    model = LogisticRegression(solver='lbfgs')
    model.fit(trainX, trainy)

    lr_probs = model.predict_proba(testX)
    lr_probs = lr_probs[:, 1]
    yhat = model.predict(testX)
    lr_precision, lr_recall, _ = precision_recall_curve(testy, lr_probs)
    lr_f1, lr_auc = f1_score(testy, yhat), auc(lr_recall, lr_precision)

    no_skill = len(testy[testy==1]) / len(testy)
    plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill'
    plt.plot(lr_recall, lr_precision, marker='.', label='Logistic')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.legend()
    plt.show()

    Logistic = namedtuple('Logistic', ['f1', 'auc'])
    return Logistic(lr_f1, lr_auc)

print(precision(df))
```




```
Logistic(f1=0.9431396786155747, auc=0.9811288957722084)
```

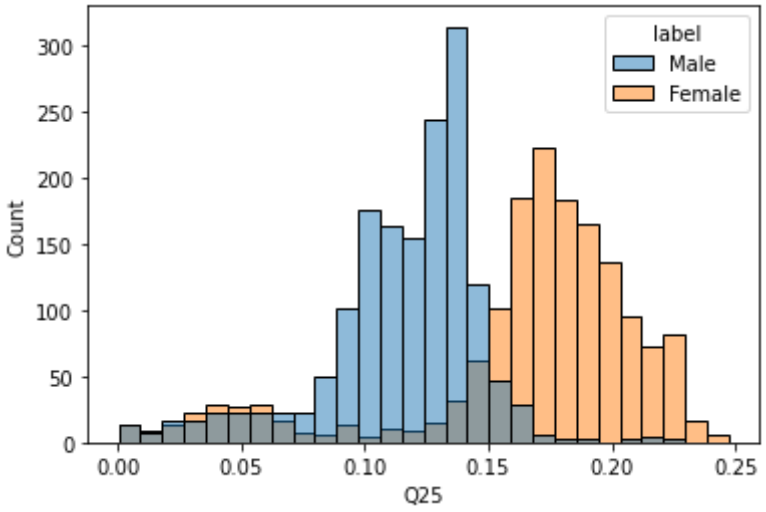
In [252]:
```
print(plot_histogram(df, column='IQR'))
```

```
1.0
```



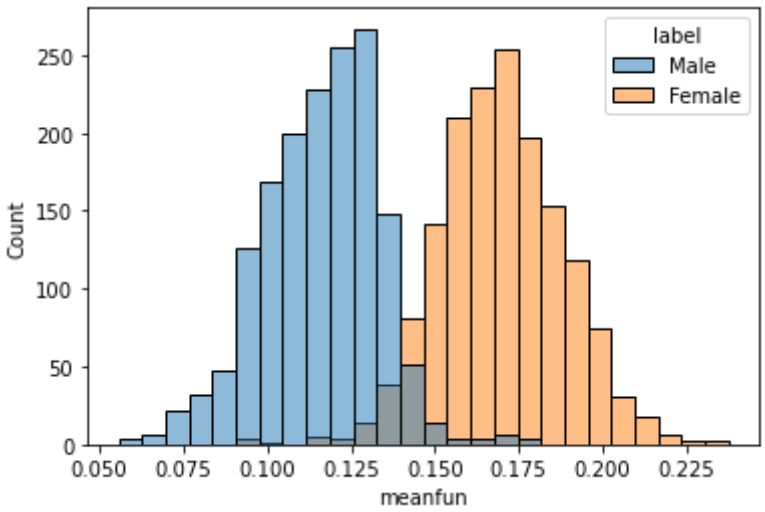
In [239]:
```
plot_histogram(df, column='Q25')
```

Out[239]:
```
<AxesSubplot:xlabel='Q25', ylabel='Count'>
```



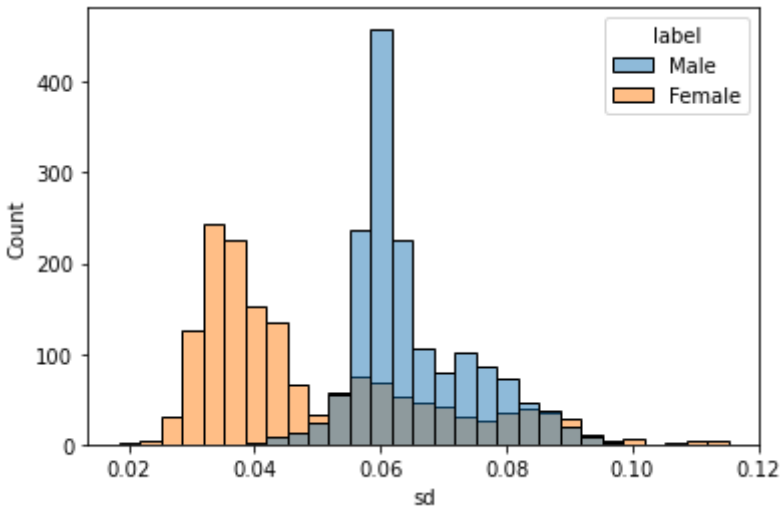
In [256]:
```
plot_histogram(df, column='meanfun')
```

Out[256]: 0.0



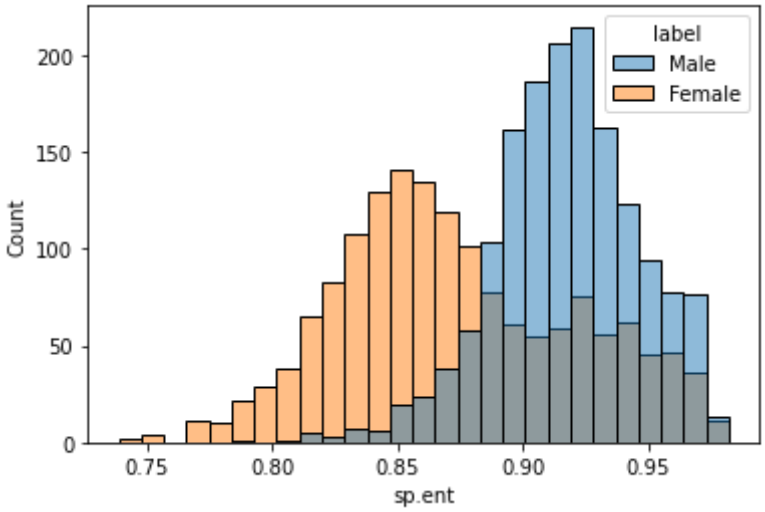
In [257]:
```
plot_histogram(df, column='sd')
```

Out[257]: 1.0



In [260]:
```
plot_histogram(df, column='sp.ent')
```

Out[260]: 1.0



In [ ]:
```

```