Tutorial da videoaula 18 - Semana 7: Comparação de Modelos Preditivos

Nesta aula, vamos comparar o desempenho dos algoritmos de classificação k-NN, SVM e RandomForest usando algumas métricas de desempenho. Vamos usar um conjunto de dados sobre a qualidade de vinhos, muito usado para avaliar algoritmos de classificação.

Sobre o Google Colab, recomendamos que, se necessário, reveja a videoaula Jupyter Notebook e Colab Google, videoaula 4 do curso COM350 - Introdução à Ciência de Dados (https://youtu.be/ZC8bfSZLl80) ou acesse a ferramenta no site https://colab.research.google.com/. Caso não tenha uma conta Google ou não queira usar, pode fazer também no Jupyter Notebook.

Qualidade de vinhos tintos

O conjunto de dados usado neste caderno está disponível em:

https://raw.githubusercontent.com/higoramario/univesp-com410-aprendizado-de-maquinas/main/vinhos-binario.csv

O conjunto de dados modificado para classificação binária está em:

https://www.kaggle.com/datasets/nareshbhat/wine-quality-binary-classification

Fonte: Naresha Bhat, Kaggle.

URL original do conjunto de dados

https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

Fonte: UCI Machine Learning Repository, Centro para Aprendizado de Máquina e Sistemas Inteligentes, Universidade da Califórnia, Irvine.

Descrição dos atributos da base de dados:

- **fixed acidity** (acidez fixa)
- **volatile acidity** (acidez volátil)
- **citric acid** (acidez cítrica)
- **residual sugar** (açúcar residual)
- **chlorides** (cloretos)
- **free sulfur dioxide** (dióxido de enxofre livre)
- **total sulfur dioxide** (dióxido de enxofre total)
- **density** (densidade)
- **pH**
- **sulphates** (sulfatos)
- **alcohol** (álcool) Atributo classe:
- **quality (score 0 or 1)** (qualidade - pontuação 0 (ruim, notas <= 5) ou 1 (bom, notas > 5))

1. Nesta atividade, usaremos as bibliotecas **scikit-learn**, **pandas** e **matplotlib**. Importe as bibliotecas.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, RocCurveDisplay
from sklearn.preprocessing import StandardScaler

plt.rcParams['figure.figsize']=[15,10]
plt.rcParams.update({'font.size': 18})
```

2. Importe a base de dados direto da URL e verifique as primeiras linhas. O arquivo contém 1599 registros.

```
url = 'https://raw.githubusercontent.com/higoramario/univesp-com410-aprendizado-de-maquinas/main/vinhos-binario.csv'
vinhos = pd.read_csv(url, sep=',')
vinhos.head(10)
```

fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0
7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	0
7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	0
11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	1
7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	0

```
vinhos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
```

```
 0  fixed acidity         1599 non-null   float64
 1  volatile acidity      1599 non-null   float64
 2  citric acid           1599 non-null   float64
 3  residual sugar        1599 non-null   float64
 4  chlorides             1599 non-null   float64
 5  free sulfur dioxide   1599 non-null   float64
 6  total sulfur dioxide  1599 non-null   float64
 7  density               1599 non-null   float64
 8  pH                    1599 non-null   float64
 9  sulphates             1599 non-null   float64
 10 alcohol               1599 non-null   float64
 11 quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

3. Separando os atributos dos rótulos e separando os dados de treinamento (90%) e teste (10%).

```
vinhos_atributos = vinhos.iloc[:,:11]
vinhos_classes = vinhos['quality']
vinhos_treino, vinhos_teste, classes_treino, classes_teste = train_test_split(vinhos_atributos, vinhos_classes,
test_size = 0.1)
```

4. Diminuindo a escala dos dados para melhorar o treinamento.

```
scaler = StandardScaler()
vinhos_treino = scaler.fit_transform(vinhos_treino)
vinhos_teste = scaler.transform(vinhos_teste)
```

5. Vamos começar com o algoritmo k-NN.

```
modelo_kNN = KNeighborsClassifier(n_neighbors = 20)
modelo_kNN.fit(vinhos_treino, classes_treino)
```

6. Verificando a acurácia de classificação do k-NN.

```
predicao_kNN = modelo_kNN.predict(vinhos_teste)
acuracia_kNN = accuracy_score(classes_teste, predicao_kNN)
print('Acurácia de classificação k-NN: {}'.format(round(acuracia_kNN, 3)*100)+'%')
```

```
Acurácia de classificação k-NN: 75.6%
```

7. A função **classification_report** traz, além da acurácia, os valores de precisão, revogação e medida F1 obtidas para cada classe.

```
print(classification_report(classes_teste, predicao_kNN))
```

```
              precision    recall  f1-score   support

           0       0.78      0.77      0.77       154
           1       0.79      0.80      0.79       166

    accuracy                           0.78       320
   macro avg       0.78      0.78      0.78       320
weighted avg       0.78      0.78      0.78       320
```

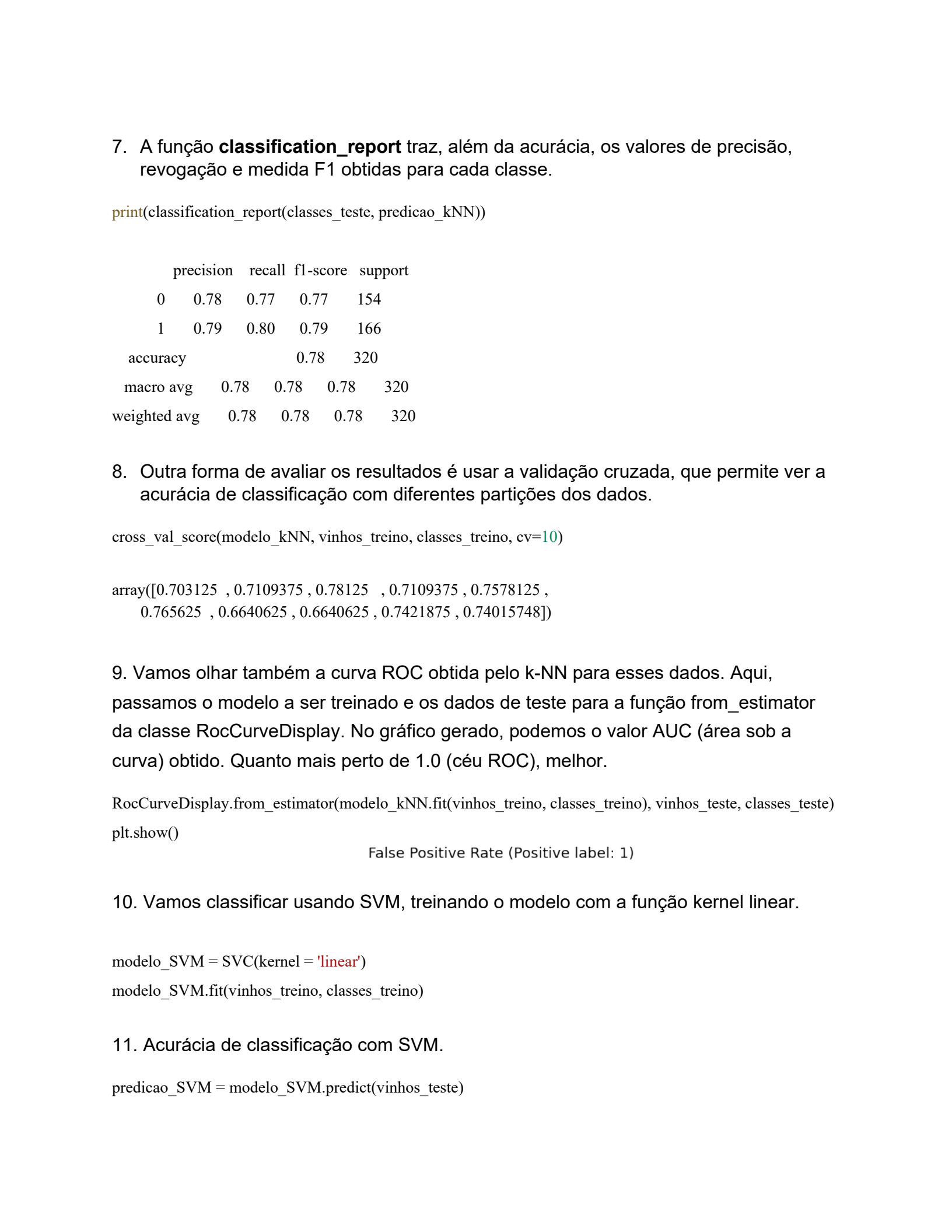
8. Outra forma de avaliar os resultados é usar a validação cruzada, que permite ver a acurácia de classificação com diferentes partições dos dados.

```
cross_val_score(modelo_kNN, vinhos_treino, classes_treino, cv=10)
```

```
array([0.703125  , 0.7109375 , 0.78125   , 0.7109375 , 0.7578125 ,
       0.765625  , 0.6640625 , 0.6640625 , 0.7421875 , 0.74015748])
```

9. Vamos olhar também a curva ROC obtida pelo k-NN para esses dados. Aqui, passamos o modelo a ser treinado e os dados de teste para a função from_estimator da classe RocCurveDisplay. No gráfico gerado, podemos o valor AUC (área sob a curva) obtido. Quanto mais perto de 1.0 (céu ROC), melhor.

```
RocCurveDisplay.from_estimator(modelo_kNN.fit(vinhos_treino, classes_treino), vinhos_teste, classes_teste)
plt.show()
```



10. Vamos classificar usando SVM, treinando o modelo com a função kernel linear.

```
modelo_SVM = SVC(kernel = 'linear')
modelo_SVM.fit(vinhos_treino, classes_treino)
```

11. Acurácia de classificação com SVM.

```
predicao_SVM = modelo_SVM.predict(vinhos_teste)
```

```
acuracia_SVM = accuracy_score(classes_teste, predicao_SVM)
print('Acurácia de classificação SVM: {}'.format(round(acuracia_SVM,3)*100)+'%')
```

Acurácia de classificação SVM: 77.5%

12. Precisão, revogação e medida F1.

```
print(classification_report(classes_teste, predicao_SVM))
```

```
              precision    recall  f1-score   support

           0       0.82      0.79      0.80        90
           1       0.74      0.77      0.76        70

    accuracy                           0.78       160
   macro avg       0.78      0.78      0.78       160
weighted avg       0.78      0.78      0.78       160
```

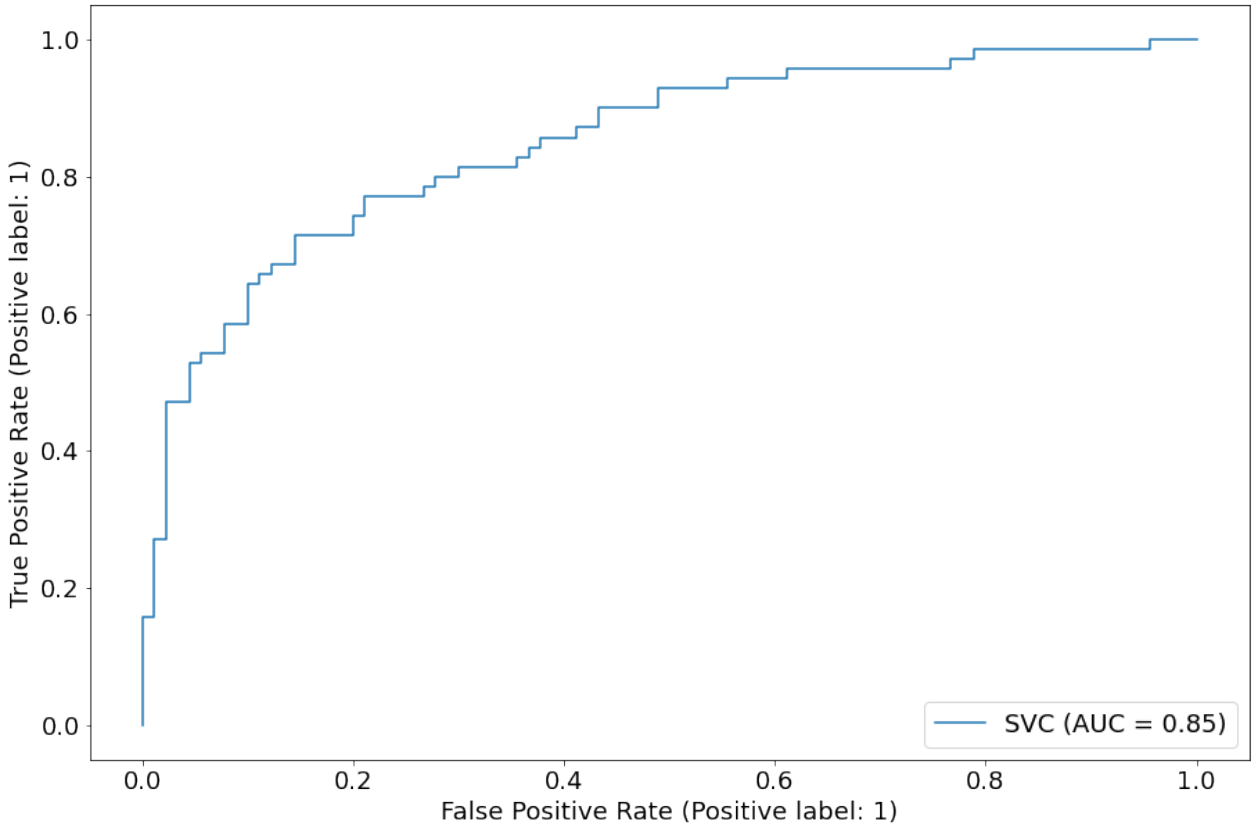
13. Usando validação cruzada.

```
cross_val_score(modelo_SVM, vinhos_treino, classes_treino, cv=10)
```

```
array([0.73611111, 0.72222222, 0.79166667, 0.72222222, 0.79166667,
       0.74305556, 0.69444444, 0.77083333, 0.76388889, 0.67132867])
```

14. Curva ROC.

```
RocCurveDisplay.from_estimator(modelo_SVM.fit(vinhos_treino, classes_treino), vinhos_teste, classes_teste)
plt.show()
```



15. Vamos fazer a classificação usando o algoritmo Random Forest.

```
modelo_RF = RandomForestClassifier()
modelo_RF.fit(vinhos_treino, classes_treino)
```

16. Acurácia de classificação com Random Forest.

```
predicao_RF = modelo_RF.predict(vinhos_teste)
acuracia_RF = accuracy_score(classes_teste, predicao_RF)
print('Acurácia de classificação RF: {}'.format(round(acuracia_RF,3)*100)+'%')
```

Acurácia de classificação RF: 82.5%

17. Precisão, revocação e medida F1.

```
print(classification_report(classes_teste, predicao_RF))
```

	precision	recall	f1-score	support

0	0.86	0.82	0.84	90
1	0.78	0.83	0.81	70
accuracy			0.82	160
macro avg	0.82	0.83	0.82	160
weighted avg	0.83	0.82	0.83	160

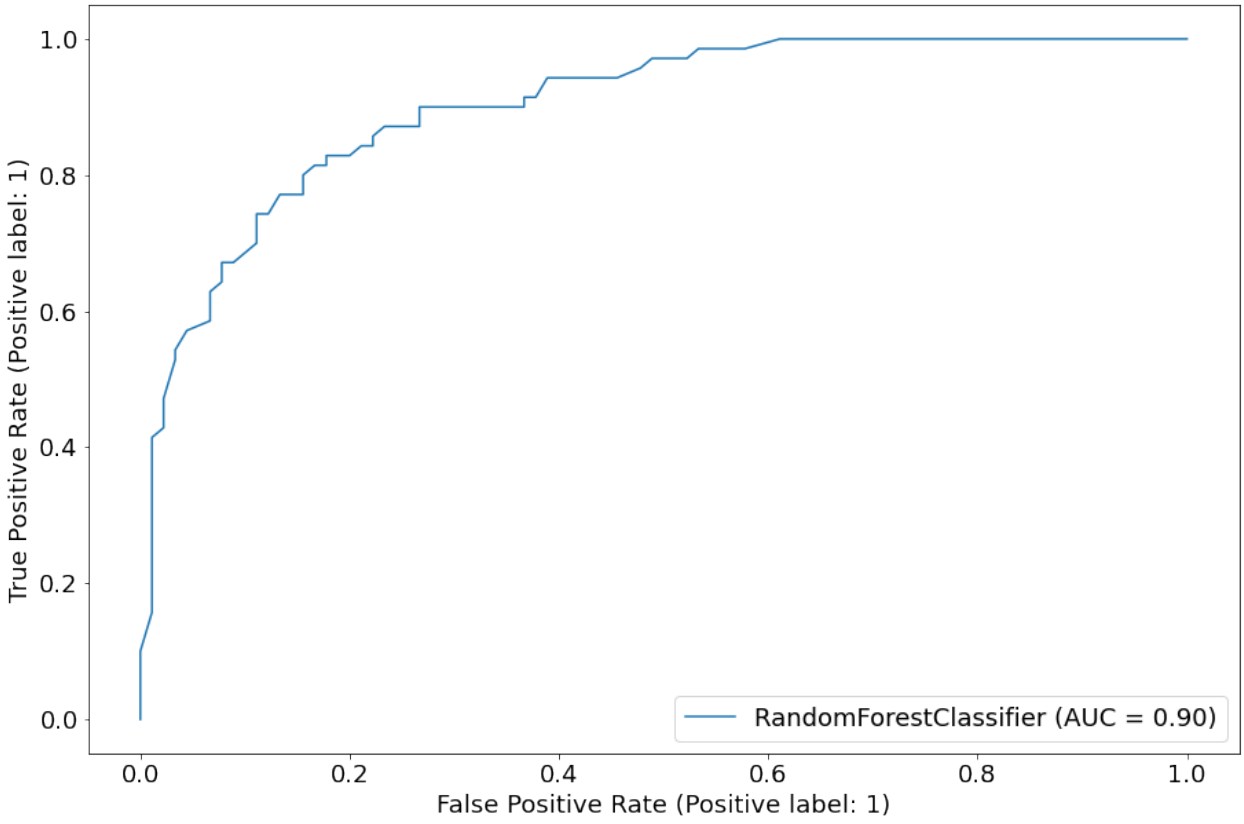
18. Validação cruzada.

```
cross_val_score(modelo_RF, vinhos_treino, classes_treino, cv=10)
```

```
array([0.83333333, 0.80555556, 0.82638889, 0.81944444, 0.83333333,
       0.8125    , 0.84027778, 0.81944444, 0.8125    , 0.81818182])
```

19. Curva ROC.

```
RocCurveDisplay.from_estimator(modelo_RF.fit(vinhos_treino, classes_treino), vinhos_teste, classes_teste)
plt.show()
```



Conclusão

De forma geral, os resultados obtidos pelos classificadores para as diferentes métricas de desempenho foram parecidos para este conjunto de dados.

Experimente alterar os parâmetros dos algoritmos para ver se consegue melhores resultados.

Versões das bibliotecas

Esse tutorial está usando as seguintes versões de bibliotecas:

matplotlib==3.2.2
pandas==1.3.5
scikit-learn==1.0.2