In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [55]:

```
df = pd.DataFrame(pd.read_csv('winemag-data-130k-v2.csv')).drop(labels = ['Unnamed: 0'],
axis = 1)
```

In [56]:

```
df.isna().sum()
```

Out[56]:

```
country                    63
description                 0
designation             37465
points                      0
price                    8996
province                   63
region_1                21247
region_2                79460
taster_name             26244
taster_twitter_handle   31213
title                       0
variety                     1
winery                      0
dtype: int64
```

In [58]:

```
df[['country', 'points', 'price', 'province', 'taster_name']]
```

Out[58]:

	country	points	price	province	taster_name
0	Italy	87	NaN	Sicily & Sardinia	Kerin O'Keefe
1	Portugal	87	15.0	Douro	Roger Voss
2	US	87	14.0	Oregon	Paul Gregutt
3	US	87	13.0	Michigan	Alexander Peartree
4	US	87	65.0	Oregon	Paul Gregutt
...	...	...	...	...	...
129966	Germany	90	28.0	Mosel	Anna Lee C. Iijima
129967	US	90	75.0	Oregon	Paul Gregutt
129968	France	90	30.0	Alsace	Roger Voss
129969	France	90	32.0	Alsace	Roger Voss
129970	France	90	21.0	Alsace	Roger Voss

129971 rows × 5 columns

In [60]:

```
# Задание 1: Для набора данных проведите устранение пропусков для одного (произвольного)
категориального признака с использованием метода заполнения отдельной категорией для проп
ущенных значений.

# В качестве произвольного признака выберем колонку "province". Заменим пропущенные значе
```

```
ния категорией "Unknown province":

df['province'].fillna('Unknown province', inplace = True)
```

In [61]:

```
df['province'].isna().sum()
```

Out[61]:

```
0
```

In [62]:

```
df[df['province'] == 'Unknown province'][['country', 'points', 'price', 'province', 'taster_name']]
```

Out[62]:

	country	points	price	province	taster_name
913	NaN	87	30.0	Unknown province	Mike DeSimone
3131	NaN	83	NaN	Unknown province	Roger Voss
4243	NaN	88	18.0	Unknown province	Mike DeSimone
9509	NaN	92	28.0	Unknown province	Susan Kostrzewa
9750	NaN	89	28.0	Unknown province	Jeff Jenssen
...	...	...	...	...	...
124176	NaN	90	30.0	Unknown province	Jeff Jenssen
129407	NaN	89	22.0	Unknown province	Michael Schachner
129408	NaN	89	22.0	Unknown province	Michael Schachner
129590	NaN	90	30.0	Unknown province	Mike DeSimone
129900	NaN	91	32.0	Unknown province	Mike DeSimone

63 rows × 5 columns

In [15]:

```
# Видно, что все пропущенные значения были успешно заменены на "Unknown province"
```

In [34]:

```
# Задание 2: Для набора данных проведите процедуру отбора признаков (feature selection). Используйте метод обертывания (wrapper method), прямой алгоритм (sequential forward selection).

from sklearn.neighbors import KNeighborsClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

In [63]:

```
# Возьмем другой датасет:
df = pd.DataFrame(pd.read_csv('KNNAlgorithmDataset.csv')).drop(labels = ['id', 'Unnamed: 32'], axis = 1).dropna()
df[['diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean' ,'compactness_mean', 'concavity_mean', 'concave points_mean']]
```

Out[63]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19
...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00

569 rows × 9 columns

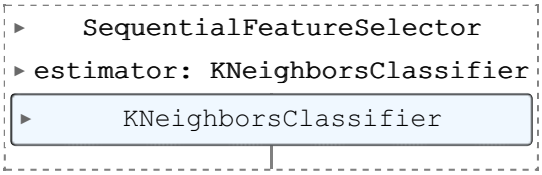
In [43]:

```
# Допустим, нам нужно предсказать диагноз, то есть diagnosis:

X = df.drop(labels = 'diagnosis', axis = 1).copy(deep = True)
Y = df['diagnosis'].copy(deep = True)

knn = KNeighborsClassifier(n_neighbors=5)

sfs = SFS(knn, forward = True, floating = False, k_features = 4)
```

In [44]:

```
sfs.fit(X, Y)
```

Out[44]:



In [45]:

```
sfs.subsets_
```

Out[45]:

```
{1: {'feature_idx': (7,),
  'cv_scores': array([0.86842105, 0.9122807 , 0.9122807 , 0.92982456, 0.90265487]),
  'avg_score': 0.9050923769600994,
  'feature_names': ('concave points_mean',)},
 2: {'feature_idx': (7, 16),
  'cv_scores': array([0.92105263, 0.93859649, 0.90350877, 0.93859649, 0.90265487]),
  'avg_score': 0.9208818506443098,
  'feature_names': ('concave points_mean', 'concavity_se')},
 3: {'feature_idx': (7, 16, 20),
  'cv_scores': array([0.85087719, 0.92105263, 0.93859649, 0.94736842, 0.9380531 ]),
  'avg_score': 0.9191895668374475,
  'feature_names': ('concave points_mean', 'concavity_se', 'radius_worst')},
 4: {'feature_idx': (7, 16, 20, 26),
  'cv_scores': array([0.92105263, 0.92982456, 0.95614035, 0.93859649, 0.94690265]),
  'avg_score': 0.9385033379909953,
  'feature_names': ('concave points_mean',
   'concavity_se',
   'radius_worst',
   'concavity_worst')}}
```

In [46]:

```
# По результатам нетрудно заметить, что лучшая точность достигается при выборе признаков
'concave points_mean','concavity_se', 'radius_worst', 'concavity_worst'
```

In [52]:

```
sns.violinplot(data = df, x = 'diagnosis', y = 'radius_mean')
```

Out[52]:

```
<Axes: xlabel='diagnosis', ylabel='radius_mean'>
```