In [50]:

```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.feature_selection import (
    SelectKBest,
    chi2,
    f_classif,
    VarianceThreshold,
    RFE,
)

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, LassoCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from mlxtend.feature_selection import SequentialFeatureSelector

```

In [51]:

```python
#Load wine dataset
wine = load_wine()
X= pd.DataFrame(wine.data, columns=wine.feature_names)
y= pd.Series(wine.target)
```

In [52]:

```python
X.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	no
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.2
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.2
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.3
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.2
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.3

In [53]:

```python
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
dtype: int32
```

In [54]:
```python
#disp the first fewmrows
print("sample dataframe: ")
print(X.head())
```

```
sample dataframe: 
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80   
1    13.20        1.78  2.14               11.2      100.0           2.65   
2    13.16        2.36  2.67               18.6      101.0           2.80   
3    14.37        1.95  2.50               16.8      113.0           3.85   
4    13.24        2.59  2.87               21.0      118.0           2.80   

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04   
1        2.76                  0.26             1.28             4.38  1.05   
2        3.24                  0.30             2.81             5.68  1.03   
3        3.49                  0.24             2.18             7.80  0.86   
4        2.69                  0.39             1.82             4.32  1.04   

   od280/od315_of_diluted_wines  proline  
0                          3.92   1065.0  
1                          3.40   1050.0  
2                          3.17   1185.0  
3                          3.45   1480.0  
4                          2.93    735.0  
```

In [55]:
```python
#normalise the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled = pd.DataFrame(X_scaled, columns=X.columns)
```

In [56]:
```python
#1. filter methods
print("\n === Filter Methods ===")
#info gain(anova f-value for classification)
f_scores, _ = f_classif(X_scaled, y)
info_gain = pd.Series(f_scores , index= X.columns)
print("Top features by info gain: ")
print(info_gain.sort_values(ascending=False). head())
```

```
 === Filter Methods ===
Top features by info gain: 
flavanoids                      233.925873
proline                         207.920374
od280/od315_of_diluted_wines    189.972321
alcohol                         135.077624
color_intensity                 120.664018
dtype: float64
```

In [57]:
```
#chi
chi_scores, _ = chi2(np.abs(X_scaled) ,y)
chi2_scores = pd.Series(chi_scores , index= X.columns)
print("Top features by chi sqaure test: ")
print(chi2_scores.sort_values(ascending=False).head())
```

```
Top features by chi sqaure test: 
flavanoids                      18.764102
proline                         17.681719
color_intensity                 15.980559
od280/od315_of_diluted_wines    15.975706
hue                             13.225143
dtype: float64
```

In [58]:
```
#cor coef
corre = X.corrwith(y)
print("\n Top features by Correlation Coefficient: ")
print(corre.abs().sort_values(ascending = False).head())
```

```

 Top features by Correlation Coefficient: 
flavanoids                      0.847498
od280/od315_of_diluted_wines    0.788230
total_phenols                   0.719163
proline                         0.633717
hue                             0.617369
dtype: float64
```

In [59]:
```
#variance threshold
vt= VarianceThreshold(threshold=0.01)
vt.fit(X)
var_features = X.columns[vt.get_support()]
print("\n Top features by variance threshold: ")
print(var_features)
```

```

 Top features by variance threshold: 
Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
       'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
       'proanthocyanins', 'color_intensity', 'hue',
       'od280/od315_of_diluted_wines', 'proline'],
      dtype='object')
```

In [60]:
```
#mad
mad_scores = X.apply(lambda col: np.mean(np.abs(col -np.mean(col))))
print("\n Top features by mad: ")
print(mad_scores.sort_values(ascending=False).head())

```
```
 Top features by mad: 
proline             259.332344
magnesium            10.999243
alcalinity_of_ash     2.595001
color_intensity       1.835831
malic_acid            0.920277
dtype: float64
```

In [61]:
```
#2. wrapper methods

print("\n =========  Wrappper methods =========")
```
```
 =========  Wrappper methods =========
```

In [62]:
```
#for selection
lr = LogisticRegression(max_iter=10000, random_state=0)
forward_selector = SequentialFeatureSelector(lr, k_features=5, forward=T
forward_selector = forward_selector.fit(X_scaled, y)
print("\nFeatures selected by Forward Selection:")
print(forward_selector.k_feature_names_)
```
```
Features selected by Forward Selection:
('alcohol', 'ash', 'alcalinity_of_ash', 'flavanoids', 'proline')
```

In [63]:
```
#back elimination
lr = LogisticRegression(max_iter=10000, random_state=0)
backwar_selector = SequentialFeatureSelector(lr, k_features=5, forward=F
backwar_selector = backwar_selector.fit(X_scaled, y)
print("\nFeatures selected by backward elimination:")
print(backwar_selector.k_feature_names_)
```
```
Features selected by backward elimination:
('alcohol', 'ash', 'flavanoids', 'color_intensity', 'proline')
```

In [64]:
```
#recursive feature elimination
rfe = RFE(lr, n_features_to_select=5)
rfe.fit(X_scaled, y)
print("\n features selected by rfe")
print(X.columns[rfe.support_])
```

```

 features selected by rfe
Index(['alcohol', 'flavanoids', 'color_intensity', 'hue', 'proline'], dtype='object')
```

In [66]:
```
#3. embed methods

#regularization(lasso)
lasso = LassoCV(cv = 5, random_state=0).fit(X_scaled, y)
lasso_features = X.columns[lasso.coef_ !=0]
print("\n features selected by lasso regularization")
print(lasso_features)
```

```

 features selected by lasso regularization
Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'total_phenols',
       'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins',
       'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'proline'],
      dtype='object')
```

In [67]:
```
#tree based method
rf = RandomForestClassifier(random_state=0)
rf.fit(X_scaled, y)
importances = pd.Series(rf.feature_importances_, index = X.columns)
print("\n top features by tree based method")
print(importances.sort_values(ascending=False).head())
```

```

 top features by tree based method
proline                         0.193999
flavanoids                      0.160954
color_intensity                 0.145267
alcohol                         0.110700
od280/od315_of_diluted_wines    0.109747
dtype: float64
```

In [ ]:
```

```