About The Wrapping Method.

- It to predict to predict the target variable on the basis randomly selective features and those will give us better prediction assumme that this features combination betttter is best for us.
- But it having drawbacks that is nothing but is computationally very expensive than filter method.
- But Perform the better than filter method.
- It is not always recommended to the higher dimensional datasets.

Types Of Wrapping Methods.

- Foreward Step Selection.(Recurssive)
- Backward Step Selection.(Recurssive)
- Exhaustive Feature Selection (Subset Selection)

In the Filter Method we do not use the machine learning algorithm but in the wrapper we use the machine learning algorithm while selecting Features.

- **Foreward Step Selection.**

- In this Foreward Step Selection method start to use the feature from intial level there we will use feature first time then it will possible and random combination with other feature to make the better the prediction.Select the best feature combination.

- **Backward Step Selection.**
- In this feature selection techique we generally initialized with the all features then step by step removing the features which are not so much putting the information to predict the best possible outcome.
- **Exhaustive Feature Selection**
- It is uses all possible combination all features.
- It requires Massive Comutational Power.

Let's Start With Machine Learning

In [56]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

In [57]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
```

In [58]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

In [59]:
```
data = pd.read_csv('Wine.csv')
```

In [60]:
```
data.head()
```

Out[60]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	Hue
0	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29	5.64	1.0
1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.0
2	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81	5.68	1.0
3	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18	7.80	0.8
4	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82	4.32	1.0

In [61]:
```
data.tail()
```

Out[61]:

	Alcohol	Malic_Acid	Ash	Ash_Alcanity	Magnesium	Total_Phenols	Flavanoids	Nonflavanoid_Phenols	Proanthocyanins	Color_Intensity	H
173	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.7	C
174	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	1.41	7.3	C
175	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	1.35	10.2	C
176	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	1.46	9.3	C
177	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2	C

In [62]:
```
data.columns
```

Out[62]:
```
Index(['Alcohol', 'Malic_Acid', 'Ash', 'Ash_Alcanity', 'Magnesium',
       'Total_Phenols', 'Flavanoids', 'Nonflavanoid_Phenols',
       'Proanthocyanins', 'Color_Intensity', 'Hue', 'OD280', 'Proline',
       'Customer_Segment'],
      dtype='object')
```

In [63]:
```
data.nunique()
```

Out[63]:
```
Alcohol                 126
Malic_Acid              133
Ash                      79
Ash_Alcanity             63
Magnesium                53
Total_Phenols            97
Flavanoids              132
Nonflavanoid_Phenols     39
Proanthocyanins         101
Color_Intensity         132
Hue                      78
OD280                   122
Proline                 121
Customer_Segment          3
dtype: int64
```

In [64]:
```
x = data.drop(columns=['Customer_Segment'])
y = data['Customer_Segment']
```

In [65]:
```
x.shape,y.shape
```

Out[65]:
```
((178, 13), (178,))
```

In [66]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,random_state=42,stratify=y)
```

In [67]:
```
x_train.shape,x_test.shape
```

Out[67]:
```
((142, 13), (36, 13))
```

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [69]:
```
sel = SFS(RandomForestClassifier(n_estimators=100,criterion='gini',random_state=42),k_features=8,forward=True,f
```

In [70]:
```
sel.fit(x_train,y_train)
```

Out[70]:
```
SequentialFeatureSelector(clone_estimator=True, cv=4,
                          estimator=RandomForestClassifier(bootstrap=True,
                                                           ccp_alpha=0.0,
                                                           class_weight=None,
                                                           criterion='gini',
                                                           max_depth=None,
                                                           max_features='auto',
                                                           max_leaf_nodes=None,
                                                           max_samples=None,
                                                           min_impurity_decrease=0.0,
                                                           min_impurity_split=None,
                                                           min_samples_leaf=1,
                                                           min_samples_split=2,
                                                           min_weight_fraction_leaf=0.0,
                                                           n_estimators=100,
                                                           n_jobs=None,
                                                           oob_score=False,
                                                           random_state=42,
                                                           verbose=0,
                                                           warm_start=False),
                          floating=False, forward=True, k_features=8, n_jobs=1,
                          pre_dispatch='2*n_jobs', scoring='accuracy',
                          verbose=0)
```

In [71]:
```
sel.k_feature_names_
```

Out[71]:
```
('Alcohol',
 'Malic_Acid',
 'Ash_Alcanity',
 'Magnesium',
 'Flavanoids',
 'Proanthocyanins',
 'Color_Intensity',
 'Hue')
```

In [72]:
```
sel.k_score_
```

Out[72]:
```
0.9930555555555556
```

In [73]:
```
pd.DataFrame.from_dict(sel.get_metric_dict()).T
```

Out[73]:

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
1	(9,)	[0.6944444444444444, 0.6388888888888888, 0.628...	0.683333	(Color_Intensity,)	0.090885	0.0566974	0.0327343
2	(6, 9)	[0.8888888888888888, 0.9722222222222222, 0.971...	0.950992	(Flavanoids, Color_Intensity)	0.0574777	0.0358567	0.0207019
3	(4, 6, 9)	[0.9166666666666666, 1.0, 0.9428571428571428, ...	0.964881	(Magnesium, Flavanoids, Color_Intensity)	0.0582191	0.0363193	0.0209689
4	(0, 4, 6, 9)	[0.9722222222222222, 0.9444444444444444, 0.971...	0.972024	(Alcohol, Magnesium, Flavanoids, Color_Intensity)	0.0314903	0.0196449	0.011342
5	(0, 4, 6, 8, 9)	[1.0, 0.9722222222222222, 0.9714285714285714, ...	0.985913	(Alcohol, Magnesium, Flavanoids, Proanthocyani...	0.0225862	0.0140901	0.00813492
6	(0, 1, 4, 6, 8, 9)	[1.0, 0.9722222222222222, 0.9714285714285714, ...	0.985913	(Alcohol, Malic_Acid, Magnesium, Flavanoids, P...	0.0225862	0.0140901	0.00813492
7	(0, 1, 4, 6, 8, 9, 10)	[1.0, 0.9722222222222222, 1.0, 1.0]	0.993056	(Alcohol, Malic_Acid, Magnesium, Flavanoids, P...	0.0192809	0.0120281	0.00694444
8	(0, 1, 3, 4, 6, 8, 9, 10)	[1.0, 0.9722222222222222, 1.0, 1.0]	0.993056	(Alcohol, Malic_Acid, Ash_Alcanity, Magnesium,...	0.0192809	0.0120281	0.00694444

Backward Selection Method

In [74]:
```
sel = SFS(RandomForestClassifier(n_estimators=100,criterion='gini',random_state=42),k_features=8,forward=False,
sel.fit(x_train,y_train)
```

Out[74]:
```
SequentialFeatureSelector(clone_estimator=True, cv=4,
                          estimator=RandomForestClassifier(bootstrap=True,
                                                           ccp_alpha=0.0,
                                                           class_weight=None,
                                                           criterion='gini',
                                                           max_depth=None,
                                                           max_features='auto',
                                                           max_leaf_nodes=None,
                                                           max_samples=None,
                                                           min_impurity_decrease=0.0,
                                                           min_impurity_split=None,
                                                           min_samples_leaf=1,
                                                           min_samples_split=2,
                                                           min_weight_fraction_leaf=0.0,
                                                           n_estimators=100,
                                                           n_jobs=None,
                                                           oob_score=False,
                                                           random_state=42,
                                                           verbose=0,
                                                           warm_start=False),
                          floating=False, forward=False, k_features=8, n_jobs=1,
                          pre_dispatch='2*n_jobs', scoring='accuracy',
                          verbose=0)
```

In [75]:
```
sbs = sel
```

In [76]:
```
sbs.k_feature_names_
```

Out[76]:
```
('Alcohol',
 'Malic_Acid',
 'Ash',
 'Ash_Alcanity',
 'Magnesium',
 'Total_Phenols',
 'Flavanoids',
 'Color_Intensity')
```

In [77]:
```
sbs.k_score_
```

Out[77]:
```
0.9930555555555556
```

In [78]:
```
pd.DataFrame(sbs.get_metric_dict()).T
```

Out[78]:

	feature_idx	cv_scores	avg_score	feature_names	ci_bound	std_dev	std_err
13	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)	[0.9722222222222222, 0.9722222222222222, 0.971...	0.978968	(Alcohol, Malic_Acid, Ash, Ash_Alcanity, Magne...	0.0194714	0.012147	0.00701308
12	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11)	[0.9722222222222222, 0.9722222222222222, 0.971...	0.978968	(Alcohol, Malic_Acid, Ash, Ash_Alcanity, Magne...	0.0194714	0.012147	0.00701308
11	(0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 11)	[1.0, 0.9722222222222222, 1.0, 1.0]	0.993056	(Alcohol, Malic_Acid, Ash, Ash_Alcanity, Magne...	0.0192809	0.0120281	0.00694444
10	(0, 1, 2, 3, 4, 5, 6, 7, 9, 10)	[1.0, 0.9444444444444444, 1.0, 1.0]	0.986111	(Alcohol, Malic_Acid, Ash, Ash_Alcanity, Magne...	0.0385617	0.0240563	0.0138889
9	(0, 1, 2, 3, 4, 5, 6, 7, 9)	[0.9722222222222222, 0.9722222222222222, 1.0, ...	0.986111	(Alcohol, Malic_Acid, Ash, Ash_Alcanity, Magne...	0.0222636	0.0138889	0.00801875
8	(0, 1, 2, 3, 4, 5, 6, 9)	[1.0, 0.9722222222222222, 1.0, 1.0]	0.993056	(Alcohol, Malic_Acid, Ash, Ash_Alcanity, Magne...	0.0192809	0.0120281	0.00694444

Exhaustive Feature Selection.

In [85]:
```
from mlxtend.feature_selection import ExhaustiveFeatureSelector as efs
```

In [86]:
```
efs = efs(RandomForestClassifier(n_estimators=100,criterion='gini',random_state=42),min_features=4,max_features
```

In [ ]:
```
efs.fit(x_train,y_train)
```

In [ ]:
```

```