Recurssive Feature Elimination.

- As it name suggesting that it recurssively removing the features and build a model using the remaining attributes and then again calculate model accuracy.
- Morever it how will do ?
- It train the model first on the all datasets,all the features and then it tries to remove least performing features and then it again train the model and find out the feature impotant among the remaing features and then it try to train the model since it is kind of the recurssive training of the model and recurssive feature elimination so it try to eliminate the feature recurssively one by one.
- Sckit learn does more the heavy lifting.
- RFE model is avalible in the feature selection we can pass any classifier under the REF module then it execute based on the estimator.
- By using the familar syntex method fit.() we can train the model and finally we can select the feature

Lets Go With An Example.

In [21]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
```

In [22]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [23]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
from sklearn.feature_selection import RFE,SelectFromModel
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

In [24]:
```
data = pd.read_csv('adult.csv')
```

In [25]:
```
data.head()
```

Out[25]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.p
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	

In [26]:
```
data['occupation'].value_counts()
```

Out[26]:
```
Prof-specialty       4140
Craft-repair         4099
Exec-managerial      4066
Adm-clerical         3770
Sales                3650
Other-service        3295
Machine-op-inspct    2002
?                    1843
Transport-moving     1597
Handlers-cleaners    1370
Farming-fishing       994
Tech-support          928
Protective-serv       649
Priv-house-serv       149
Armed-Forces            9
Name: occupation, dtype: int64
```

In [27]:
```
from sklearn.preprocessing import LabelEncoder
```

In [28]:
```
Lable = LabelEncoder()
data['workclass']= Lable.fit_transform(data['workclass'])
data['education']= Lable.fit_transform(data['education'])
data['marital.status']= Lable.fit_transform(data['marital.status'])
data['occupation']= Lable.fit_transform(data['occupation'])
```

In [29]:
```
data['relationship']= Lable.fit_transform(data['relationship'])
data['race']= Lable.fit_transform(data['race'])
data['sex']= Lable.fit_transform(data['sex'])
```

In [30]:
```
data.head()
```

Out[30]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.w
0	90	0	77053	11	9	6	0	1	4	0	0	4356	
1	82	4	132870	11	9	6	4	1	4	0	0	4356	
2	66	0	186061	15	10	6	0	4	2	0	0	4356	
3	54	4	140359	5	4	0	7	4	4	0	0	3900	
4	41	4	264663	15	10	5	10	3	4	0	0	3900	

In [31]:
```
data['native.country']=Lable.fit_transform(data['native.country'])
```

In [32]:
```
data['income'].value_counts()
```

Out[32]:
```
<=50K    24720
>50K      7841
Name: income, dtype: int64
```

In [33]:
```
data['income']=Lable.fit_transform(data['income'])
```

In [34]:
```
data.head()
```

Out[34]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.w
0	90	0	77053	11	9	6	0	1	4	0	0	4356	
1	82	4	132870	11	9	6	4	1	4	0	0	4356	
2	66	0	186061	15	10	6	0	4	2	0	0	4356	
3	54	4	140359	5	4	0	7	4	4	0	0	3900	
4	41	4	264663	15	10	5	10	3	4	0	0	3900	

In [35]:
```
data.tail()
```

Out[35]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.p
32556	22	4	310152	15	10	4	11	1	4	1	0	0	
32557	27	4	257302	7	12	2	13	5	4	0	0	0	
32558	40	4	154374	11	9	2	7	0	4	1	0	0	
32559	58	4	151910	11	9	6	1	4	4	0	0	0	
32560	22	4	201490	11	9	4	1	3	4	1	0	0	

In [36]:
```
x= data.drop(columns=['income'])
y = data['income']
```

In [37]:
```
x.shape,y.shape
```

Out[37]:
```
((32561, 14), (32561,))
```

In [38]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,train_size=0.80,stratify=y,random_state=42)
```

In [39]:
```
x_train.shape,y_train.shape
```

Out[39]:
```
((26048, 14), (26048,))
```

Select the Feature On It's Importance.

In [40]:
```
sel = SelectFromModel(RandomForestClassifier())
sel.fit(x_train,y_train)
```

Out[40]:
```
SelectFromModel(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=None, verbose=0,
                                                 warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

In [41]:
```
feature = x_train.columns[sel.get_support()]
```

In [42]:
```
feature
```

Out[42]:
```
Index(['age', 'fnlwgt', 'education.num', 'relationship', 'capital.gain',
       'hours.per.week'],
      dtype='object')
```

In [43]:
```
len(feature)
```

Out[43]:
```
6
```

In [44]:
```
sel.estimator_.feature_importances_
```

Out[44]:
```
array([0.15087312, 0.03935164, 0.16717067, 0.03417951, 0.09064132,
       0.06592096, 0.06849939, 0.10301673, 0.01394773, 0.0130365 ,
       0.11736544, 0.03602059, 0.08284031, 0.0171361 ])
```

In [45]:
```
x_train_rfs = sel.transform(x_train)
x_test_rfs = sel.transform(x_test)
```

In [46]:
```
x_train_rfs.shape,x_test_rfs.shape
```

Out[46]:
```
((26048, 6), (6513, 6))
```

Build The Model

In [47]:
```
def RandomForest(x_train,x_test,y_train,y_test):
    clf = RandomForestClassifier(random_state=42,n_estimators=100)
    clf.fit(x_train,y_train)
    y_pred = clf.predict(x_test)
    print('Accuracy :',accuracy_score(y_test,y_pred))
```

In [48]:
```
%%time
RandomForest(x_train_rfs,x_test_rfs,y_train,y_test)
```

```
Accuracy : 0.8317211730385383
Wall time: 11.7 s
```

In [49]:
```
%%time
RandomForest(x_train,x_test,y_train,y_test)
```

```
Accuracy : 0.8582834331337326
Wall time: 11.6 s
```

Recurssive Feature Elimination With The Help Of RFE

In [50]:
```
from sklearn.feature_selection import RFE
```

In [51]:
```
sel = RFE(RandomForestClassifier(n_estimators=100,random_state=42,n_jobs=-1),n_features_to_select=6)
sel.fit(x_train,y_train)
```

Out[51]:
```
RFE(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=-1,
                                     oob_score=False, random_state=42,
                                     verbose=0, warm_start=False),
    n_features_to_select=6, step=1, verbose=0)
```

In [58]:
```
sel.get_support()
```

Out[58]:
```
array([ True, False,  True, False,  True, False, False,  True, False,
       False,  True, False,  True, False])
```

In [59]:
```
feature = x_train.columns[sel.get_support()]
```

In [60]:
```
feature
```

Out[60]:
```
Index(['age', 'fnlwgt', 'education.num', 'relationship', 'capital.gain',
       'hours.per.week'],
      dtype='object')
```

In [61]:
```
x_train_RFE = sel.transform(x_train)
x_test_RFE  = sel.transform(x_test)
```

In [62]:
```
x_train_RFE.shape,x_test_RFE.shape
```

Out[62]:
```
((26048, 6), (6513, 6))
```

In [64]:
```
%%time
RandomForest(x_train_RFE,x_test_RFE,y_train,y_test)
```

```
Accuracy : 0.8317211730385383
Wall time: 11.5 s
```

Select the Feature By using the GradientBoost Algorithm.

In [65]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [66]:
```
sel = SelectFromModel(GradientBoostingClassifier(learning_rate=0.01,n_estimators=100,random_state=42))
sel.fit(x_train,y_train)
```

Out[66]:
```
SelectFromModel(estimator=GradientBoostingClassifier(ccp_alpha=0.0,
                                                     criterion='friedman_mse',
                                                     init=None,
                                                     learning_rate=0.01,
                                                     loss='deviance',
                                                     max_depth=3,
                                                     max_features=None,
                                                     max_leaf_nodes=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_iter_no_change=None,
                                                     presort='deprecated',
                                                     random_state=42,
                                                     subsample=1.0, tol=0.0001,
                                                     validation_fraction=0.1,
                                                     verbose=0,
                                                     warm_start=False),
                max_features=None, norm_order=1, prefit=False, threshold=None)
```

In [67]:
```
Feature = x_train.columns[sel.get_support()]
```

In [68]:
```
Feature
```

Out[68]:
```
Index(['education.num', 'relationship', 'capital.gain'], dtype='object')
```

In [69]:
```
len(Feature)
```

Out[69]:
```
3
```

In [70]:
```
x_train_sel = sel.transform(x_train)
x_test_sel = sel.transform(x_test)
```

In [71]:
```
x_train_sel.shape,x_test_sel.shape
```

Out[71]:
```
((26048, 3), (6513, 3))
```

In [72]:
```
def GradientBoost(x_train,x_test,y_train,y_test):
    model = GradientBoostingClassifier(learning_rate=0.1,random_state=42)
    model.fit(x_train,y_train)
    y_pred = model.predict(x_test)
    print('Accuarcy :',accuracy_score(y_test,y_pred))
```

In [74]:
```
%%time
GradientBoost(x_train_sel,x_test_sel,y_train,y_test)
```

```
Accuarcy : 0.8486104713649624
Wall time: 3.37 s
```

Go with RFE.

In [75]:
```
sel = RFE(GradientBoostingClassifier(learning_rate=0.1,n_estimators=100),n_features_to_select=7)
sel.fit(x_train,y_train)
```

Out[75]:
```
RFE(estimator=GradientBoostingClassifier(ccp_alpha=0.0,
                                         criterion='friedman_mse', init=None,
                                         learning_rate=0.1, loss='deviance',
                                         max_depth=3, max_features=None,
                                         max_leaf_nodes=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_iter_no_change=None,
                                         presort='deprecated',
                                         random_state=None, subsample=1.0,
                                         tol=0.0001, validation_fraction=0.1,
                                         verbose=0, warm_start=False),
    n_features_to_select=7, step=1, verbose=0)
```

In [76]:
```
sel.get_support()
```

Out[76]:
```
array([ True, False, False, False,  True, False,  True,  True, False,
       False,  True,  True,  True, False])
```

In [77]:
```
feature = x_train.columns[sel.get_support()]
```

In [78]:
```
feature
```

Out[78]:
```
Index(['age', 'education.num', 'occupation', 'relationship', 'capital.gain',
       'capital.loss', 'hours.per.week'],
      dtype='object')
```

In [79]:
```
x_train_rfe = sel.transform(x_train)
x_test_rfe = sel.transform(x_test)
```

In [80]:
```
%%time
GradientBoost(x_train_rfe,x_test_rfe,y_train,y_test)
```

```
Accuarcy : 0.8619683709504069
Wall time: 6.69 s
```

In [81]:
```
for index in range(1,10):
    sel= RFE(RandomForestClassifier(n_estimators =100,random_state=42),n_features_to_select=index)
    sel.fit(x_train,y_train)
    x_train_rfe= sel.transform(x_train)
    x_test_rfe = sel.transform(x_test)
    print('Selected Feature :',index)
    RandomForest(x_train_rfe,x_test_rfe,y_train,y_test)
    print()
```

```
Selected Feature : 1
Accuracy : 0.6810993397819746

Selected Feature : 2
Accuracy : 0.7371411024105635

Selected Feature : 3
Accuracy : 0.777214801166897

Selected Feature : 4
Accuracy : 0.7767541839398127

Selected Feature : 5
Accuracy : 0.82081989866421

Selected Feature : 6
Accuracy : 0.8317211730385383

Selected Feature : 7
Accuracy : 0.8429295255642562

Selected Feature : 8
Accuracy : 0.8458467680024566

Selected Feature : 9
Accuracy : 0.8475356978350991
```

In [ ]: