In [1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv("/Users/SAURABH/Saurabh patil/DATA SCIENCE/random Forest/Company_Data.csv")
df
```

Out[2]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
1	11.22	111	48	16	260	83	Good	65	10	Yes	Yes
2	10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
3	7.40	117	100	4	466	97	Medium	55	14	Yes	Yes
4	4.15	141	64	3	340	128	Bad	38	13	Yes	No
...	...	...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	Good	33	14	Yes	Yes
396	6.14	139	23	3	37	120	Medium	55	11	No	Yes
397	7.41	162	26	12	368	159	Medium	40	18	Yes	Yes
398	5.94	100	79	7	284	95	Bad	50	12	Yes	Yes
399	9.71	134	37	0	27	120	Good	49	16	Yes	Yes

400 rows × 11 columns

In [3]:
```
df.Sales.describe()
```

Out[3]:
```
count    400.000000
mean       7.496325
std        2.824115
min        0.000000
25%        5.390000
50%        7.490000
75%        9.320000
max       16.270000
Name: Sales, dtype: float64
```

In [4]:
```
#Changing the categorical variables into dummies.
df1 = pd.get_dummies(df)

#Converting the Target variable i.e. Sales into Categorical
df1['Category'] = pd.cut(df1['Sales'],
                         bins=[0,10, np.inf],
                         labels=['Low/Mid','High'],
                         include_lowest=True)
df1
```

Out[4]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education	ShelveLoc_Bad	ShelveLoc_Good	ShelveLoc_Medium	Urban_No	Urban_Yes	US_No	US_Yes	Category
0	9.50	138	73	11	276	120	42	17	1	0	0	0	1	0	1	Low/Mid
1	11.22	111	48	16	260	83	65	10	0	1	0	0	1	0	1	High
2	10.06	113	35	10	269	80	59	12	0	0	1	0	1	0	1	High
3	7.40	117	100	4	466	97	55	14	0	0	1	0	1	0	1	Low/Mid
4	4.15	141	64	3	340	128	38	13	1	0	0	0	1	1	0	Low/Mid
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	33	14	0	1	0	0	1	0	1	High
396	6.14	139	23	3	37	120	55	11	0	0	1	1	0	0	1	Low/Mid
397	7.41	162	26	12	368	159	40	18	0	0	1	0	1	0	1	Low/Mid
398	5.94	100	79	7	284	95	50	12	1	0	0	0	1	0	1	Low/Mid
399	9.71	134	37	0	27	120	49	16	0	1	0	0	1	0	1	Low/Mid

400 rows × 16 columns

In [5]:
```
# Random Forest Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

In [6]:
```
array = df1.values
X = array[:,1:15]
Y = array[:,15]
```

In [7]:
```
num_trees = 100
max_features = 4
kfold = KFold(n_splits=10, random_state=7)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean()*100)
```

```
86.75
```

Let's try if we can increase thee cv score using ensemble techniques

Bagging

In [8]:
```
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier

seed = 7

cart = DecisionTreeClassifier()
num_trees = 100
model1 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results1 = cross_val_score(model1, X, Y, cv=kfold)
print(results1.mean()*100)
```

```
86.25
```

Boosting

In [9]:
```
# AdaBoost Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier

model2 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results2 = cross_val_score(model2, X, Y, cv=kfold)
print(results2.mean()*100)
```

```
89.25
```

Stacking¶

In [10]:
```
# Stacking Ensemble for Classification
from pandas import read_csv
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

Iteration -1

In [11]:
```
# create the sub models
estimators = []
model3 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model3))
model4 = DecisionTreeClassifier()
estimators.append(('cart', model4))
model5 = SVC()
estimators.append(('svm', model5))
model6 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
estimators.append(('bagging', model6))
model7 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model7))

# create the ensemble modelItrer
ensemble = VotingClassifier(estimators)
results3 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results3.mean()*100)
```

```
90.5
```

Iteration-2

In [12]:
```
# create the sub models
estimators = []
model8 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model8))
model9 = DecisionTreeClassifier()
estimators.append(('cart', model9))
model10 = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
estimators.append(('bagging', model10))
model11 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model11))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results4 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results4.mean()*100)
```

```
89.49999999999999
```

Iteration -3

In [13]:
```
# create the sub models
estimators = []
model12 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model12))
model13 = DecisionTreeClassifier()
estimators.append(('cart', model13))
model14 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model14))

# create the ensemble modSel
ensemble = VotingClassifier(estimators)
results5 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results5.mean()*100)
```

```
90.5
```

Iteration-4

In [14]:
```
# create the sub models
estimators = []
model15 = DecisionTreeClassifier()
estimators.append(('cart', model15))
model16 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model16))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results6 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results6.mean()*100)
```

```
85.75
```

Iteration-5

In [15]:
```
# create the sub models
estimators = []
model17 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model17))
model18 = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
estimators.append(('boosting', model18))

# create the ensemble model
ensemble = VotingClassifier(estimators)
results6 = cross_val_score(ensemble, X, Y, cv=kfold)
print(results6.mean()*100)
```

```
90.0
```

Since the cv score for iteration 3 was the max, so we can consider it to be our final model

In [ ]:
```
```