In [1]:

```
!pip install category_encoders
import category_encoders as ce
import pandas as pd
from sklearn import datasets
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
import warnings
warnings.filterwarnings("ignore")
```

```
Requirement already satisfied: category_encoders in c:\users\acer\anaconda3\lib\site-packages (2.2.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\acer\anaconda3\lib\site-packages (from category_e
ncoders) (1.5.2)
Requirement already satisfied: pandas>=0.21.1 in c:\users\acer\anaconda3\lib\site-packages (from category
_encoders) (1.1.3)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\acer\anaconda3\lib\site-packages (from ca
tegory_encoders) (0.23.2)
Requirement already satisfied: patsy>=0.5.1 in c:\users\acer\anaconda3\lib\site-packages (from category_e
ncoders) (0.5.1)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\acer\anaconda3\lib\site-packages (from cate
gory_encoders) (0.12.0)
Requirement already satisfied: numpy>=1.14.0 in c:\users\acer\anaconda3\lib\site-packages (from category_
encoders) (1.19.2)
Requirement already satisfied: pytz>=2017.2 in c:\users\acer\anaconda3\lib\site-packages (from pandas>=0.
21.1->category_encoders) (2020.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\acer\anaconda3\lib\site-packages (from
pandas>=0.21.1->category_encoders) (2.8.1)
Requirement already satisfied: joblib>=0.11 in c:\users\acer\anaconda3\lib\site-packages (from scikit-lea
rn>=0.20.0->category_encoders) (0.17.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\acer\anaconda3\lib\site-packages (from sc
ikit-learn>=0.20.0->category_encoders) (2.1.0)
Requirement already satisfied: six in c:\users\acer\anaconda3\lib\site-packages (from patsy>=0.5.1->categ
ory_encoders) (1.15.0)
```

In [2]:

```
# import company data set
sales = pd.read_csv('/Users/acer/Sandesh Pal/Data Science Assgn/random Forest/Company_Data.csv')
```

In [3]:

```
sales
```

Out[3]:

	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 [4]:

```
# checking for null values
sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Sales        400 non-null    float64
 1   CompPrice    400 non-null    int64
 2   Income       400 non-null    int64
 3   Advertising  400 non-null    int64
 4   Population   400 non-null    int64
 5   Price        400 non-null    int64
 6   ShelveLoc    400 non-null    object
 7   Age          400 non-null    int64
 8   Education    400 non-null    int64
 9   Urban        400 non-null    object
 10  US           400 non-null    object
dtypes: float64(1), int64(7), object(3)
memory usage: 34.5+ KB
```

In [5]:

```
sales.describe()
```

Out[5]:

	Sales	CompPrice	Income	Advertising	Population	Price	Age	Education
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	7.496325	124.975000	68.657500	6.635000	264.840000	115.795000	53.322500	13.900000
std	2.824115	15.334512	27.986037	6.650364	147.376436	23.676664	16.200297	2.620528
min	0.000000	77.000000	21.000000	0.000000	10.000000	24.000000	25.000000	10.000000
25%	5.390000	115.000000	42.750000	0.000000	139.000000	100.000000	39.750000	12.000000
50%	7.490000	125.000000	69.000000	5.000000	272.000000	117.000000	54.500000	14.000000
75%	9.320000	135.000000	91.000000	12.000000	398.500000	131.000000	66.000000	16.000000
max	16.270000	175.000000	120.000000	29.000000	509.000000	191.000000	80.000000	18.000000

In [6]:

```
import category_encoders as ce
# encode variables with ordinal encoding
encoder = ce.OrdinalEncoder(cols=['ShelveLoc', 'Urban', 'US'])
sales1 = encoder.fit_transform(sales)
```

In [8]:

```
# Converting the Target column i.e. Sales into Categorical value using mean of the column i.e. 7.49
sales_val = []
for value in sales["Sales"]:
    if value<=7.49:
        sales_val.append("low")
    else:
        sales_val.append("high")

sales1["sales_val"]= sales_val
```

In [9]:

```
sales1.head()
```

Out[9]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US	sales_val
0	9.50	138	73	11	276	120	1	42	17	1	1	high
1	11.22	111	48	16	260	83	2	65	10	1	1	high
2	10.06	113	35	10	269	80	3	59	12	1	1	high
3	7.40	117	100	4	466	97	3	55	14	1	1	low
4	4.15	141	64	3	340	128	1	38	13	1	2	low

In [10]:

```
x = sales1.drop(['sales_val','Sales'], axis =1)
y = sales1['sales_val']
```

In [11]:

```
x
```

Out[11]:

	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
0	138	73	11	276	120	1	42	17	1	1
1	111	48	16	260	83	2	65	10	1	1
2	113	35	10	269	80	3	59	12	1	1
3	117	100	4	466	97	3	55	14	1	1
4	141	64	3	340	128	1	38	13	1	2
...	...	...	...	...	...	...	...	...	...	...
395	138	108	17	203	128	2	33	14	1	1
396	139	23	3	37	120	3	55	11	2	1
397	162	26	12	368	159	3	40	18	1	1
398	100	79	7	284	95	1	50	12	1	1
399	134	37	0	27	120	2	49	16	1	1

400 rows × 10 columns

In [12]:

```
y.head(10)
```

Out[12]:

```
0    high
1    high
2    high
3     low
4     low
5    high
6     low
7    high
8     low
9     low
Name: sales_val, dtype: object
```

Random Forest Classification

In [13]:

```
num_trees = 200
max_features = 4
kfold = KFold(n_splits=15)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.8002849002849003
```

lets use the various ensemble techniques to check the accuracy %

Bagging

In [14]:

```
# Bagged Decision Trees for Classification
from sklearn.ensemble import BaggingClassifier
seed = 7
kfold = KFold(n_splits=15, random_state=seed)
cart = DecisionTreeClassifier()
num_trees = 200
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.8176638176638177
```

Boosting

In [15]:

```
# AdaBoost Classification
from sklearn.ensemble import AdaBoostClassifier
num_trees = 200
seed=7
```

```
kfold = KFold(n_splits=15, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.8375118708452043
```

Stacking

In [16]:

```
# Stacking Ensemble for Classification
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

In [17]:

```
# create the sub models
estimators = []
model1 = LogisticRegression(max_iter=500)
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
# create the ensemble model
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, x, y, cv=kfold)
print(results.mean())
```

```
0.7899335232668566
```

In [ ]: