Random Forests Assignment

Data Set - Company_Data

1. Import Necessary libraries

In [1]:
```python
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt
import seaborn as sns

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

2. Import Data

In [2]:
```python
company_details = pd.read_csv('Company_Data.csv')
company_details
```

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

3. Data Understanding

In [3]:
```python
company_details.head()
```

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

In [4]:
```python
company_details.shape
```

Out[4]:
```
(400, 11)
```

In [5]:
```python
company_details.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 [6]:
```python
company_details.isna().sum()
```

Out[6]:
```
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64
```

In [7]:
```python
company_details.describe()
```

Out[7]:

	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 [8]:
```python
company_details.dtypes
```

Out[8]:
```
Sales          float64
CompPrice        int64
Income           int64
Advertising      int64
Population       int64
Price            int64
ShelveLoc       object
Age              int64
Education        int64
Urban           object
US              object
dtype: object
```

In [9]:
```python
company_details.columns
```

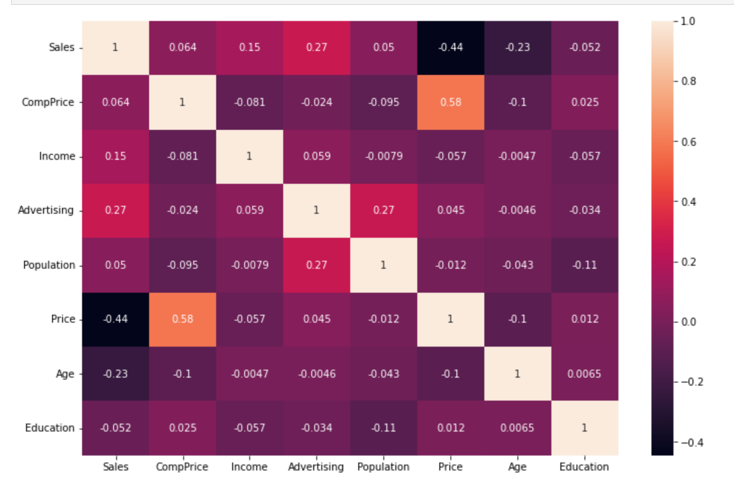
Out[9]:
```
Index(['Sales', 'CompPrice', 'Income', 'Advertising', 'Population', 'Price',
       'ShelveLoc', 'Age', 'Education', 'Urban', 'US'],
      dtype='object')
```

In [10]:
```python
company_details.ShelveLoc.value_counts()
```

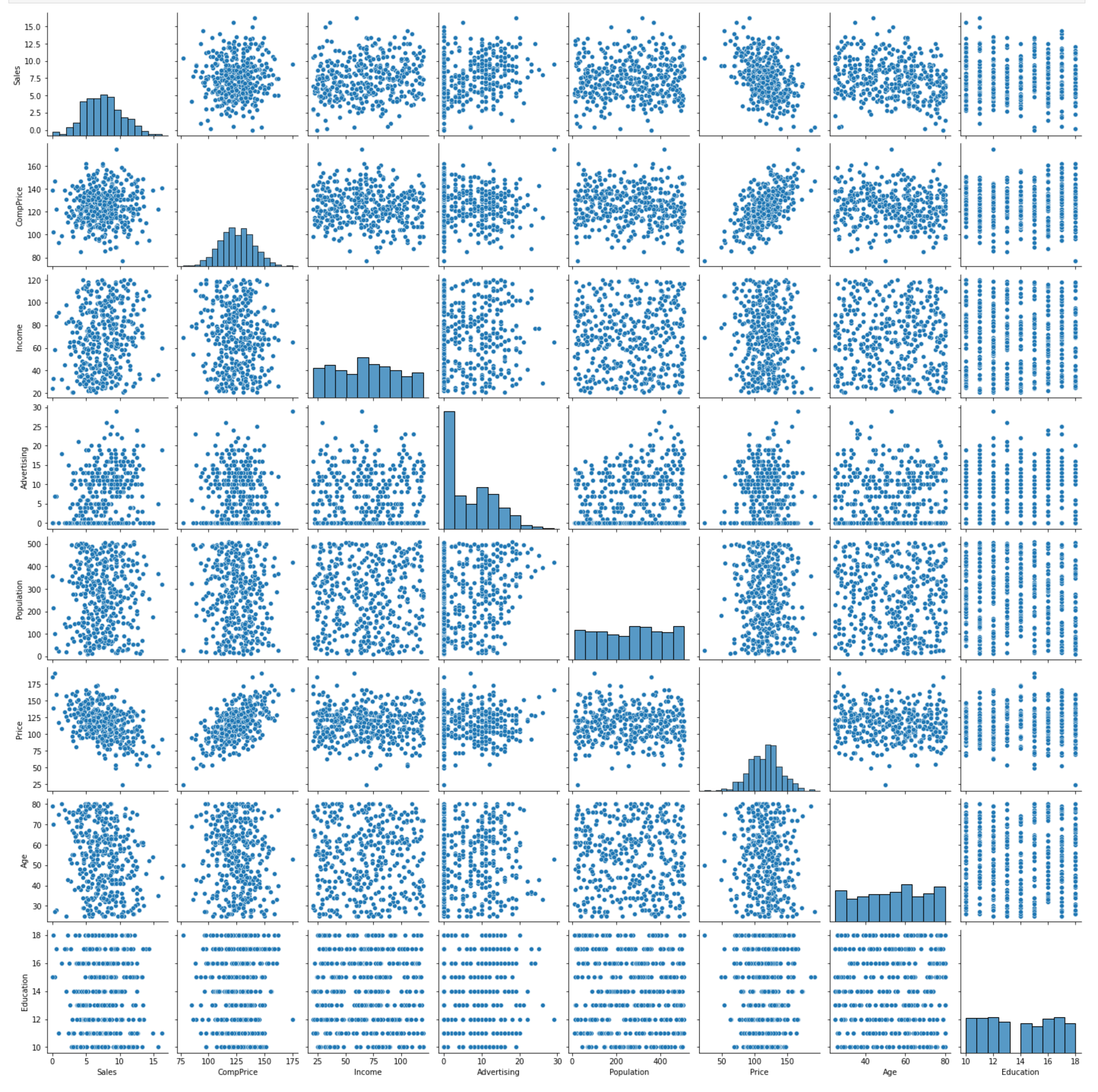
Out[10]:
```
Medium    219
Bad        96
Good       85
Name: ShelveLoc, dtype: int64
```

3.1 Correlation Matrix :

In [11]:
```python
plt.figure(figsize = (12,8))
sns.heatmap(company_details.corr(),annot = True)
plt.show()
```



In [12]:
```python
sns.pairplot(company_details)
plt.show()
```



3.2 Category Encoder :

In [13]:
```python
!pip install category_encoders
```

```
Requirement already satisfied: category_encoders in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (2.5.1.post0)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from category_encoders) (0.13.2)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from category_encoders) (1.0.2)
Requirement already satisfied: pandas>=1.0.5 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from category_encoders) (1.4.2)
Requirement already satisfied: patsy>=0.5.1 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from category_encoders) (0.5.2)
Requirement already satisfied: numpy>=1.14.0 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from category_encoders) (1.21.5)
Requirement already satisfied: scipy>=1.0.0 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from category_encoders) (1.7.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders) (2021.3)
Requirement already satisfied: six in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (1.1.0)
Requirement already satisfied: packaging>=21.3 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from statsmodels>=0.9.0->category_encoders) (21.3)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\mohammed faisal khan\anaconda3\lib\site-packages (from packaging>=21.3->statsmodels>=0.9.0->category_encoders)
(3.0.4)
```

In [14]:
```python
from category_encoders import OrdinalEncoder
from sklearn import preprocessing
```

In [15]:
```python
encoder = OrdinalEncoder(cols = ["ShelveLoc", "Urban", "US"])
sales = encoder.fit_transform(company_details)
```

In [16]:
```python
sale_val = []

for value in company_details['Sales']:

    if value <= 7.49:
        sale_val.append("low")
    else:
        sale_val.append("high")
        sales['sale_val'] = pd.Series(sale_val)
```

In [17]:
```python
sales
```

Out[17]:

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US	sale_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
...	...	...	...	...	...	...	...	...	...	...	...	...
395	12.57	138	108	17	203	128	2	33	14	1	1	high
396	6.14	139	23	3	37	120	3	55	11	2	1	low
397	7.41	162	26	12	368	159	3	40	18	1	1	low
398	5.94	100	79	7	284	95	1	50	12	1	1	low
399	9.71	134	37	0	27	120	2	49	16	1	1	high

400 rows × 12 columns

3.3 Splitting in x and y :

In [18]:
```python
x = sales.drop(['sale_val', 'Sales'],axis = 1)
x
```

Out[18]:

	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 [19]:
```python
y = sales['sale_val']
y
```

Out[19]:
```
0      high
1      high
2      high
3       low
4       low
       ...
395    high
396     low
397     low
398     low
399    high
Name: sale_val, Length: 400, dtype: object
```

4. Bagged Decision Trees for Classification

In [20]:
```python
from sklearn.model_selection import KFold
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.model_selection import cross_val_score
```

In [21]:
```python
cart_1 = DecisionTreeClassifier()
```

In [22]:
```python
num_trees_1 = 100
```

In [23]:
```python
k_fold_1 = KFold(n_splits = 10,shuffle = True, random_state = 8)
model_1 = BaggingClassifier(base_estimator = cart_1, n_estimators = num_trees_1, random_state = 8)
```

In [24]:
```python
results = cross_val_score(model_1, x,y, cv = k_fold_1)
print(results.mean())
```

```
0.82
```

5. Random Forest Classification

In [25]:
```python
from sklearn.ensemble import RandomForestClassifier
```

In [26]:
```python
cart_2 = RandomForestClassifier()
```

In [27]:
```python
num_trees_2 = 100
max_features = 3
```

In [28]:
```python
k_fold_2 = KFold(n_splits = 10, shuffle = True ,random_state = 8)
model_2 = RandomForestClassifier(n_estimators = num_trees_2, max_features = max_features)
```

In [29]:
```python
results = cross_val_score(model_2, x, y, cv = k_fold_2)
print(results.mean())
```

```
0.8099999999999999
```

6. Boost Classification

In [30]:
```python
from sklearn.ensemble import AdaBoostClassifier
```

In [31]:
```python
cart_3 = AdaBoostClassifier()
```

In [32]:
```python
seed = 8
num_trees_3 = 100
```

In [33]:
```python
k_fold_3 = KFold(n_splits = 100, shuffle = True, random_state = seed)
model_3 = AdaBoostClassifier(n_estimators = num_trees_3, random_state = seed)
```

In [34]:
```python
results = cross_val_score(model_3, x,y, cv = k_fold_3)
print(results.mean())
```

```
0.8275
```

7. Stacking Ensemble for Classification

In [35]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
```

In [36]:
```python
k_fold_4 = KFold(n_splits = 10,shuffle = True, random_state = 8)
estimators = []
```

Create the sub models

In [37]:
```python
model_4 = LogisticRegression(max_iter = 100)
estimators.append(('logistic', model_4))
```

In [38]:
```python
model_5 = DecisionTreeClassifier()
estimators.append(('cart', model_5))
```

In [39]:
```python
model_6 = SVC()
estimators.append(('svm', model_6))
```

Create the ensemble model

In [40]:
```python
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, x, y, cv = k_fold_4)
print(results.mean())
```

```
0.79
```

Conclusion :

```
a) Model Accuracy for Decision Tree Classifier : 0.82
b) Model Accuracy for Random Forest Classifier : 0.8099
c) Model Accuracy for Ada Boost Classifier     : 0.8275
d) Model Accuracy for Stacking Ensemble        : 0.79
```