In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")
```

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

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

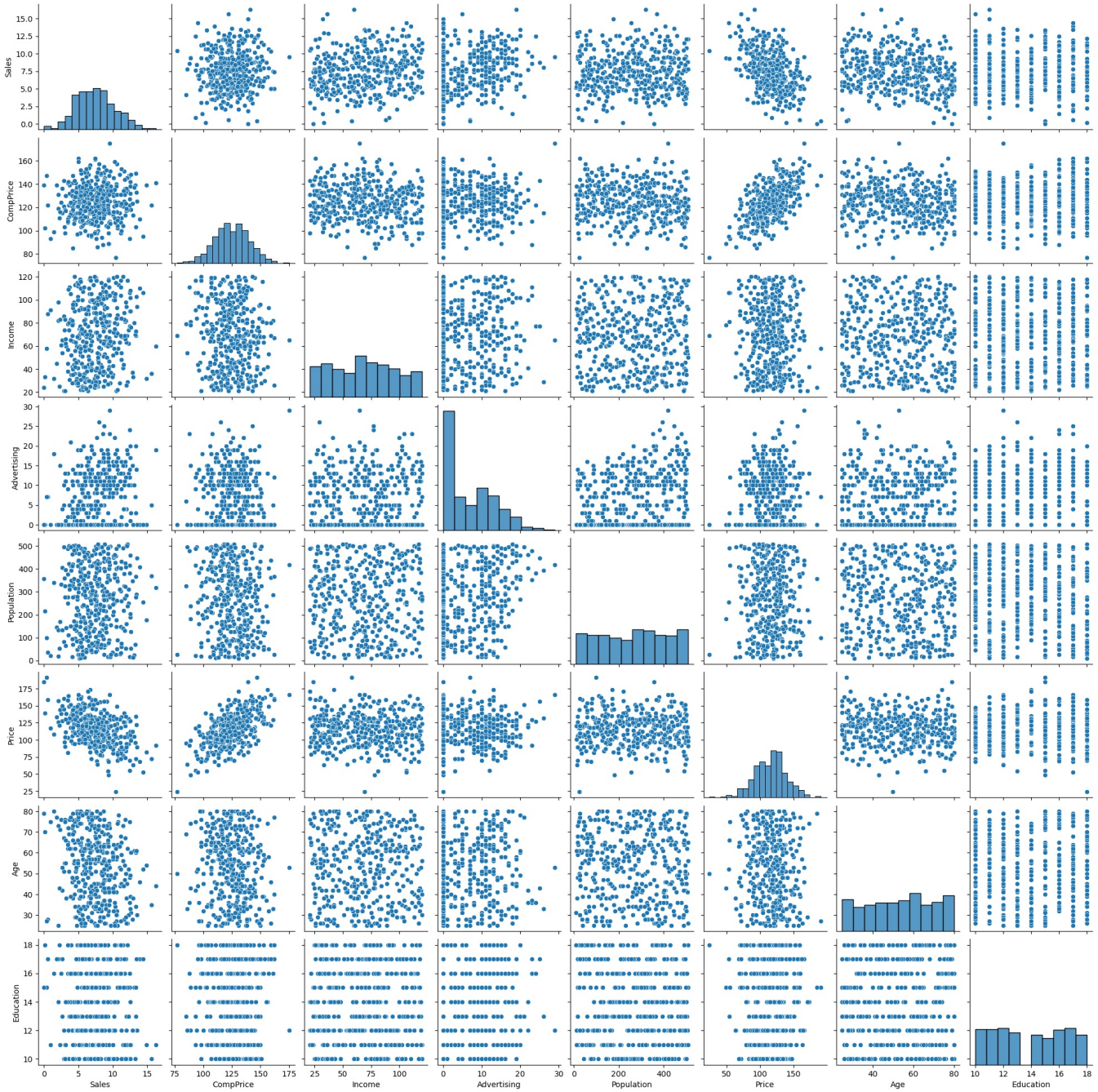
400 rows × 11 columns

In [3]:
```
data.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 [4]:
```
sns.pairplot(data)
```

Out[4]:
```
<seaborn.axisgrid.PairGrid at 0x2b2ccef6370>
```



In [5]:
```
sns.heatmap(data.isnull(),cmap='Blues')
```

Out[5]:
```
<AxesSubplot:>
```

In [6]:
```
plt.figure(figsize=(20,10))
sns.heatmap(data.corr(),annot=True)
```

Out[6]:
```
<AxesSubplot:>
```

In [7]:
```
get_ipython().system('pip install category_encoders')
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
     ---------------------------------------- 81.9/81.9 kB 1.5 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\rohit\anaconda3\lib\site-packages (from category_encoders) (1.0.2)
Requirement already satisfied: numpy>=1.14.0 in c:\users\rohit\anaconda3\lib\site-packages (from category_encoders) (1.21.5)
Requirement already satisfied: scipy>=1.0.0 in c:\users\rohit\anaconda3\lib\site-packages (from category_encoders) (1.9.1)
Requirement already satisfied: pandas>=1.0.5 in c:\users\rohit\anaconda3\lib\site-packages (from category_encoders) (1.4.4)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\rohit\anaconda3\lib\site-packages (from category_encoders) (0.13.2)
Requirement already satisfied: patsy>=0.5.1 in c:\users\rohit\anaconda3\lib\site-packages (from category_encoders) (0.5.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\rohit\anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\rohit\anaconda3\lib\site-packages (from pandas>=1.0.5->category_encoders) (2022.1)
Requirement already satisfied: six in c:\users\rohit\anaconda3\lib\site-packages (from patsy>=0.5.1->category_encoders) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rohit\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\rohit\anaconda3\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (1.1.0)
Requirement already satisfied: packaging>=21.3 in c:\users\rohit\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\rohit\anaconda3\lib\site-packages (from packaging>=21.3->statsmodels>=0.9.0->category_encoders) (3.0.9)
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

In [8]:
```
from category_encoders import OrdinalEncoder
encoder = OrdinalEncoder(cols=["ShelveLoc", "Urban", "US"])
sales = encoder.fit_transform(data)
```

In [16]:
```
data["sale_val"] = ["low" if value <= 7.49 else "high" for value in data["Sales"]]
```

In [21]:
```
sale_val = []
for value in data['Sales']:
      if value <= 7.49:
          sale_val.append("low")
      else:
          sale_val.append("high")
sales["sale_val"]= sale_val
```

In [22]:
```
sales
```

Out[22]:

	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

In [23]:
```
x = sales.drop(['sale_val', 'Sales'],axis=1)
```

In [24]:
```
x
```

Out[24]:

	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 [25]:
```
y= sales['sale_val']
```

In [26]:
```
y
```

Out[26]:
```
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
```

In [27]:
```
kfold = KFold(n_splits=10,shuffle = True, random_state = 8)
cart = DecisionTreeClassifier()
num_trees = 100
model = BaggingClassifier(base_estimator=cart, n_estimators=num_trees, random_state=8)
results = cross_val_score(model, x,y, cv=kfold)
print(results.mean())
```

```
0.82
```

In [28]:
```
num_trees = 100
max_features = 3
kfold = KFold(n_splits=10, shuffle= True ,random_state=8)
model = RandomForestClassifier(n_estimators=num_trees, max_features=max_features)
results = cross_val_score(model, x, y, cv=kfold)
print(results.mean())
```

```
0.8225
```

In [29]:
```
num_trees = 100
seed=8
kfold = KFold(n_splits=100, shuffle = True, random_state=seed)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = cross_val_score(model, x,y, cv=kfold)
print(results.mean())
```

```
0.8275
```

In [30]:
```
kfold = KFold(n_splits=10,shuffle=True, random_state=8)
estimators = []
model1 = LogisticRegression(max_iter=100)
estimators.append(('logistic', model1))
model2 = DecisionTreeClassifier()
estimators.append(('cart', model2))
model3 = SVC()
estimators.append(('svm', model3))
ensemble = VotingClassifier(estimators)
results = cross_val_score(ensemble, x, y, cv=kfold)
print(results.mean())
```

```
0.7899999999999999
```

In [ ]:
```
```