—---------import libraries—----------
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn import datasets
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

—---------read data—----------
```
data = pd.read_csv('Downloads/Company_Data.csv')
data
```

	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

—---------dummies—----------
```
df = pd.get_dummies(data,columns=['Urban','US'])
df.head(6)
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban_No	Urban_Yes	US_No	US_Yes
0	9.50	138	73	11	276	120	Bad	42	17	0	1	0	1
1	11.22	111	48	16	260	83	Good	65	10	0	1	0	1
2	10.06	113	35	10	269	80	Medium	59	12	0	1	0	1
3	7.40	117	100	4	466	97	Medium	55	14	0	1	0	1
4	4.15	141	64	3	340	128	Bad	38	13	0	1	1	0
5	10.81	124	113	13	501	72	Bad	78	16	1	0	0	1

—----------map—----------
```
df['ShelveLoc'] = df['ShelveLoc'].map({'Good':1,'Medium':2,'Bad':0})
df.head(6)
```

	Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban_No	Urban_Yes	US_No	US_Yes
0	9.50	138	73	11	276	120	0	42	17	0	1	0	1
1	11.22	111	48	16	260	83	1	65	10	0	1	0	1
2	10.06	113	35	10	269	80	2	59	12	0	1	0	1
3	7.40	117	100	4	466	97	2	55	14	0	1	0	1
4	4.15	141	64	3	340	128	0	38	13	0	1	1	0
5	10.81	124	113	13	501	72	0	78	16	1	0	0	1

—————divide————————————
x = df.iloc[:,0:6]
y = df['ShelveLoc']

```
df['ShelveLoc'].unique()
```

```
array([0, 1, 2], dtype=int64)
```

```
colnames = list(df.columns)
colnames
```

```
['Sales',
 'CompPrice',
 'Income',
 'Advertising',
 'Population',
 'Price',
 'ShelveLoc',
 'Age',
 'Education',
 'Urban_No',
 'Urban_Yes',
 'US_No',
 'US_Yes']
```

labels = np.array(df['Income'])
features = df.drop('Income',axis=1)
featurelist = list(df.columns)
features = np.array(df)

—————traintest————————————
train_features,test_features,train_labels,test_labels =
train_test_split(features,labels,test_size=0.3)

train_features.shape,test_features.shape,train_labels.shape,test_labels.shape

```
((280, 13), (120, 13), (280,), (120,))
```

—————randomforest————————————
rf = RandomForestRegressor(n_estimators=1000,random_state=30)
rf.fit(train_features,train_labels)

```
RandomForestRegressor(n_estimators=1000, random_state=30)
```

```
predictions = rf.predict(test_features)
errors = abs(predictions-test_labels)
print('error',round(np.mean(errors),2))
```

```
error 0.2
```

```
mape = 100*(errors/test_labels)
a = 100-np.mean(mape)
print('Accuracy',a)
```

```
Accuracy 99.62431147493149
```

```
—----------decisiontree—--------------------
model = DecisionTreeClassifier(criterion='entropy',max_depth=30)
model.fit(train_features,train_labels)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=30
```

```
tree.plot_tree
```

```
<function sklearn.tree._export.plot_tree(decision_tree, *, max_depth=None, feature_names=None, class_names=None, label='all', filled=False, impurity=True, node_ids=False, proportion=False, rotate='deprecated', rounded=False, precision=3, ax=None, fontsize=None)>
```

```
predict = model.predict(test_features)
predict
```

```
array([ 28,  26,  90,  69,  94,  98,  90,  63,  80,  47,  21, 105,  81,
        80,  36,  42,  79,  43,  72,  78,  53,  88,  31,  72,  34,  33,
        58,  62,  83,  30,  42,  71, 117,  81,  68,  36,  60,  57, 116,
        42,  65,  35, 116,  94,  90,  26,  38,  94, 101,  48,  83, 113,
        73,  42,  69,  63,  33,  35,  96,  77,  37,  42,  28,  34,  72,
        30,  63,  44,  32,  71,  28,  91,  84,  60,  58,  30,  64, 113,
        28,  91,  79,  93,  28,  30,  47,  54,  82,  33,  48, 100,  25,
       100,  51,  50,  93,  68, 111,  78,  62,  88,  71,  73,  42,  80,
        33, 118,  84,  95,  90,  74,  40,  75,  64,  93,  91,  67, 120,
        84,  39,  65], dtype=int64)
```

```
np.mean(predict==test_labels)
```

```
0.7166666666666667
```