Ashpak Sheikh

DECISION TREE CLASSIFIER

BATCH : DSWDMOD 020421

Business problem: - A cloth manufacturing company is interested to know about the different attributes contributing to high sales. Build a decision tree & random forest model with Sales as target variable (first convert it into categorical variable).

About data: -

We have been given data of companies sales ,price , income and age.

Analysis with Python: -

importing required libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


#loading data set


company=pd.read_csv("D:/DataScience/Class/assignment working/DC/Company_Data.csv")
```

checking descreption

```
company.describe()
```

```
In [461]: company.describe()
Out[461]:
            Sales   CompPrice      Income  ...       Price         Age   Education
count  400.000000  400.000000  400.000000  ...  400.000000  400.000000  400.000000
mean     7.496325  124.975000   68.657500  ...  115.795000   53.322500   13.900000
std      2.824115   15.334512   27.986037  ...   23.676664   16.200297    2.620528
min      0.000000   77.000000   21.000000  ...   24.000000   25.000000   10.000000
25%      5.390000  115.000000   42.750000  ...  100.000000   39.750000   12.000000
50%      7.490000  125.000000   69.000000  ...  117.000000   54.500000   14.000000
75%      9.320000  135.000000   91.000000  ...  131.000000   66.000000   16.000000
max     16.270000  175.000000  120.000000  ...  191.000000   80.000000   18.000000

[8 rows x 8 columns]
```

checking missing data

```
company.isna().sum()
```

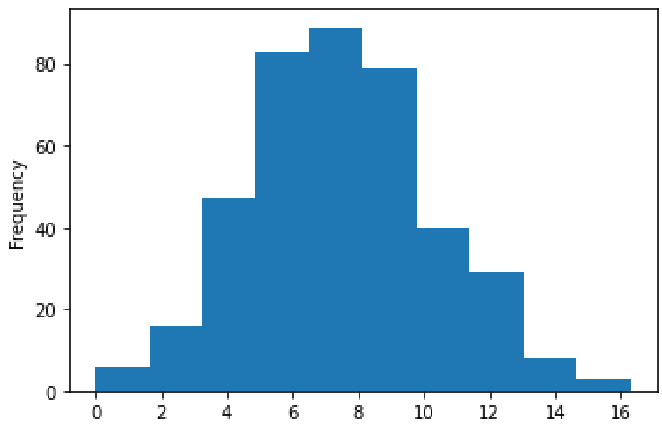
```
Out[463]:
Sales          0
CompPrice      0
Income         0
Advertising    0
Population     0
Price          0
ShelveLoc      0
Age            0
Education      0
Urban          0
US             0
dtype: int64
```

labeling categorical data

```
from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

company["ShelveLoc"]=lb.fit_transform(company["ShelveLoc"])


company=pd.get_dummies(company,columns=["Urban","US"],drop_first=True)
```

categorizing sales

```
company["Sales"].plot(kind="hist")
```




```
#as the data is in symmetry ,creating two categories


company["Sales"]=pd.cut(company["Sales"],bins=2,labels=("low","high"))
```

seperating target and predictors

target=company.Sales.map({"low":0,"high":1})

predictors=company.drop("Sales",axis=1)

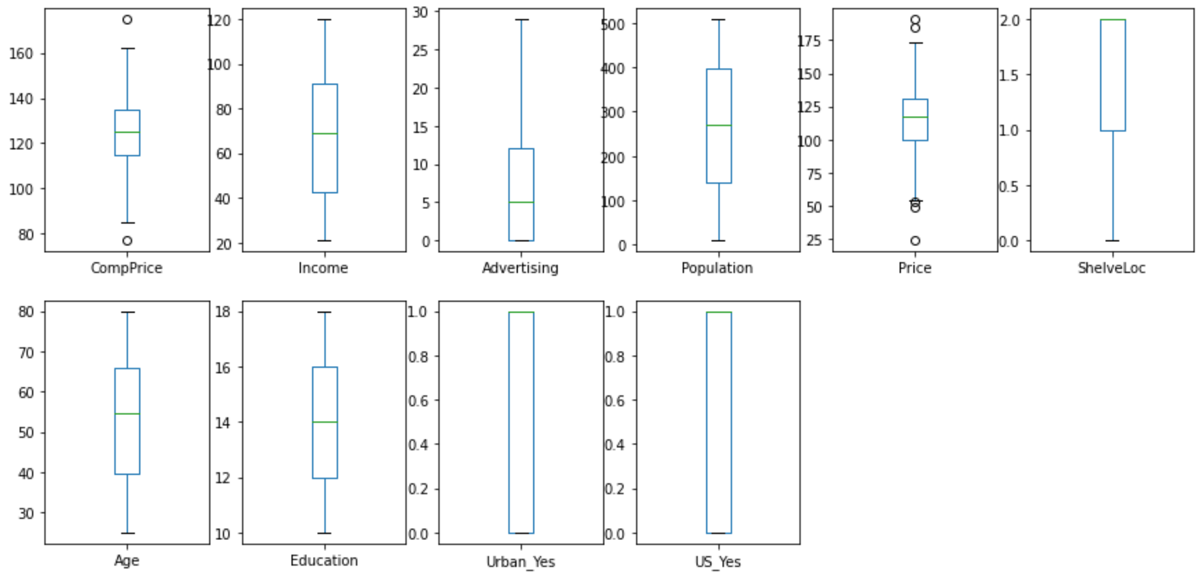
target.value_counts()

```
In [474]: target.value_counts()
Out[474]:
0    241
1    159
Name: Sales, dtype: int64
```

checking outliers

predictors.plot(figsize=(15,15),kind="box",subplots=True,layout=(4,6))

plt.show()



Ashpak Sheikh
DECISION TREE CLASSIFIER
BATCH : DSWDMOD 020421

normalizing data

writing custom function to normalize data

```
def norm(x):
   z=(x-x.min())/(x.max()-x.min())
   return z
predictor=norm(predictors)
```

splitting data into train and test

```
from sklearn.model_selection import train_test_split , GridSearchCV

x_train, x_test ,y_train , y_test =train_test_split(predictor,target,random_state=125,test_size=0.25)
```

importing decision tree classifier

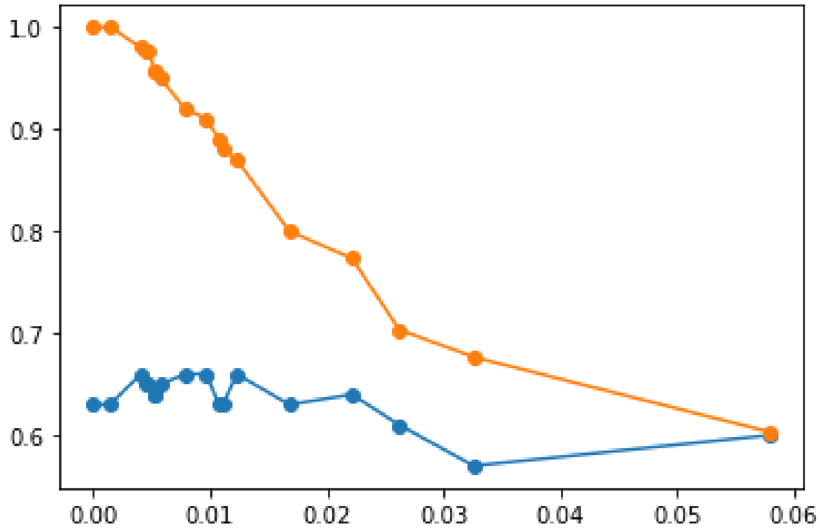
```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree

dt=DecisionTreeClassifier(  min_samples_split=3,max_depth=5, random_state=125,ccp_alpha=0.06)
dt.fit(x_train,y_train)
clf=dt.fit(x_train,y_train)

np.mean(dt.predict(x_test)==y_test)
np.mean(dt.predict(x_train)==y_train)
```

```
np.mean(dt.predict(x_test)==y_test)
0.6

np.mean(dt.predict(x_train)==y_train)
0.6033333333333334
```

```
plt.figure(figsize=(15,10))
tree.plot_tree(clf,filled=True)
plt.show()
```

Ashpak Sheikh
DECISION TREE CLASSIFIER
BATCH : DSWDMOD 020421

```
from sklearn.metrics import r2_score,mean_squared_error


r2_score(y_test,dt.predict(x_test))

mean_squared_error(y_test,dt.predict(x_test))


path=clf.cost_complexity_pruning_path(x_train, y_train)

ccp_alphas,impurities= path.ccp_alphas,path.impurities

clfs=[]

for ccp_alpha in ccp_alphas:

    clf=DecisionTreeClassifier(random_state=125,ccp_alpha=ccp_alpha)

    clf.fit(x_train, y_train)

    clfs.append(clf)



train_score=[clf.score(x_train, y_train) for clf in clfs]

test_score=[clf.score(x_test, y_test) for clf in clfs]


plt.figure()

plt.plot(ccp_alphas,test_score,marker="o")

plt.plot(ccp_alphas,train_score,marker='o')

plt.show()
```



grid search for DecisionTree classifier

base_learn=DecisionTreeClassifier()

paramiter_grid={"max_features" : [2,3,4,5,6,7,8,9],"min_samples_split":[2,3,4,5,6]}

grid_search=GridSearchCV(base_learn,paramiter_grid,scoring="accuracy" ,cv=5 ,n_jobs=-1 ,random_state=125)

grid_search.fit(x_train,y_train)

grid_search.best_params_

```
grid_search.best_params_
{'max_features': 2, 'min_samples_split': 5}
```

Decision tree on best parameters from grid search

from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier(max_features= 4, min_samples_split=6,ccp_alpha=0.055)

dt.fit(x_train, y_train)

test_pred=dt.predict(x_test)

np.mean(y_test==test_pred)

np.mean(y_train==dt.predict(x_train))

```
np.mean(y_test==test_pred)
0.6
```

```
np.mean(y_train==dt.predict(x_train))
0.6033333333333334
```

Random forest

from sklearn.ensemble import RandomForestClassifier

rf_clf=RandomForestClassifier(n_estimators=250, n_jobs=-1,random_state=125)

param_grid = {"max_depth": [2,3,4,5,6,7],"min_samples_split" : [2,3,4,5,6,7],"max_features":[2,3,4,5,6,7,8,9]}

g_search=GridSearchCV(rf_clf, param_grid,n_jobs=-1,cv=5,scoring="accuracy")

g_search.fit(x_train,y_train)

g_search.best_params_

```
g_search.best_params_
{'max_depth': 6, 'max_features': 3, 'min_samples_split': 4}
```

g_search.best_score_

```
g_search.best_score_
0.8266666666666668
```

random forest classification based on grid search results

rf_clf=RandomForestClassifier(n_estimators=400, n_jobs=-1,random_state=125,max_depth=5, max_features=9 , min_samples_split=4,ccp_alpha=0.098)

rf_clf.fit(x_test,y_test)

#accuracy scores of testing and training

from sklearn.metrics import accuracy_score

accuracy_score(y_test, rf_clf.predict(x_test))

accuracy_score(y_train, rf_clf.predict(x_train))

```
accuracy_score(y_test, rf_clf.predict(x_test))
0.66
```

```
accuracy_score(y_train, rf_clf.predict(x_train))
0.6533333333333333
```

#random forest giving more accuracy

Summary and inference: -

- Random forest is performing better than the Decision Tree.
- Random forest giving more accuracy and improved performance.
- Though still its not good accuracy we might need to try another classifier technique.