Ashpak Sheikh

DECISION TREE CLASSIFIER

BATCH : DSWDMOD 020421

Business problem: -

Build a Decision Tree & Random Forest model on the fraud data. Treat those who have taxable_income <= 30000 as Risky and others as Good (discretize the taxable income)

About data: - we have been given data about marital status, city population, and income of individuals

Analysis with Python: -

importing required libraries to handle and manipulate data

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import sklearn.tree as tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import GridSearchCV
```

```
#importing data set

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

checking descreption of data

```
fraud.describe(include="all")
```

```
In [587]: fraud.describe()
Out[587]:
       Taxable.Income  City.Population  Work.Experience
count      600.000000       600.000000       600.000000
mean     55208.375000    108747.368333        15.558333
std      26204.827597     49850.075134         8.842147
min      10003.000000     25779.000000         0.000000
25%      32871.500000     66966.750000         8.000000
50%      55074.500000    106493.500000        15.000000
75%      78611.750000    150114.250000        24.000000
max      99619.000000    199778.000000        30.000000
```

Ashpak Sheikh
DECISION TREE CLASSIFIER
BATCH : DSWDMOD 020421

checking null/missing values

fraud.isna().sum()

```
In [588]: fraud.isna().sum()
Out[588]:
Undergrad          0
Marital.Status     0
Taxable.Income     0
City.Population    0
Work.Experience    0
Urban              0
dtype: int64
```

treating categorical columns

fraud["Undergrad"]=fraud["Undergrad"].map({"YES":1 , "NO" : 0})

fraud["Urban"] = fraud["Urban"].map({"YES" : 1 , "NO" : 0})

#descrotizing Taxable.Income as per in bussiness problem

fraud["Taxable.Income"]
=np.where(fraud["Taxable.Income"]<=30000,"Risky",np.where(fraud["Taxable.Income"]>30000,"Good",fraud["Taxable.Income"]))

labeling data

from sklearn.preprocessing import LabelEncoder

lb=LabelEncoder()

fraud["Marital.Status"] = lb.fit_transform(fraud["Marital.Status"])

creating dummies

fraud["Taxable.Income"]=pd.get_dummies(fraud,columns=(["Taxable.Income"]), drop_first=True)

splitting data into target and predictors

fraud.columns

target=fraud['Taxable.Income']

```
predictors=fraud.drop('Taxable.Income',axis=1)
```

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 training and testing

```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(predictors,target)
```

Decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier
clf= DecisionTreeClassifier(random_state=125)
clf.fit(x_train,y_train)

from sklearn.metrics import accuracy_score

accuracy_score(y_train,clf.predict(x_train))
accuracy_score(y_test, clf.predict(x_test))
```

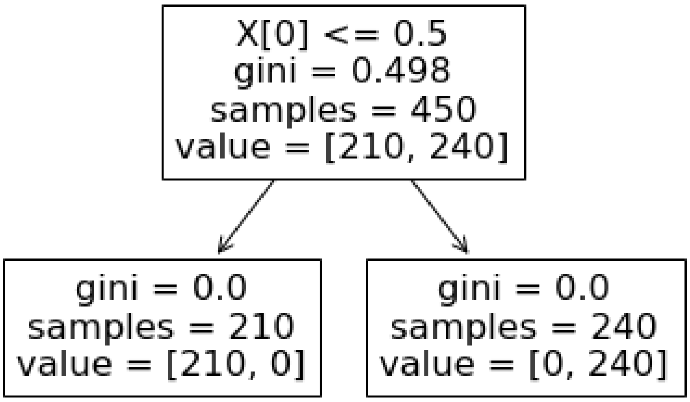
```
accuracy_score(y_train,clf.predict(x_train))
1.0

accuracy_score(y_test, clf.predict(x_test))
1.0
```

Ashpak Sheikh
DECISION TREE CLASSIFIER
BATCH : DSWDMOD 020421

plotting tree

from sklearn import tree

tree.plot_tree(clf)



Grid search Cross validation

from sklearn.model_selection import GridSearchCV

dt=DecisionTreeClassifier(random_state=125)

param={"max_depth" : [2,3,4,5,6],"max_features":[2,3,4,5],"ccp_alpha" : [0.001,0.002,0.05,0.01,0.02,0.04]}

search=GridSearchCV(dt,param,scoring="accuracy",n_jobs=-1,cv=5)

search.fit(x_train,y_train)

search.best_params_

```
search.best_params_
{'ccp_alpha': 0.001, 'max_depth': 2, 'max_features': 3}
```

final decision tree on best parameters

dt_1=DecisionTreeClassifier(max_depth=2,max_features=3,ccp_alpha=0.004 ,random_state=245)

dt_1.fit(x_train,y_train)

accuracy_score(y_train,dt_1.predict(x_train))

accuracy_score(y_test, dt_1.predict(x_test))

```
accuracy_score(y_train,dt_1.predict(x_train))
1.0

accuracy_score(y_test, dt_1.predict(x_test))
1.0
```

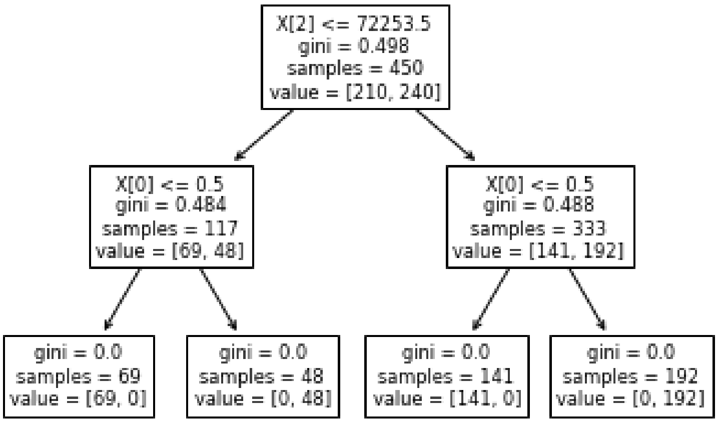
pd.crosstab(y_train,dt_1.predict(x_train))

pd.crosstab(y_test, dt_1.predict(x_test))

```
In [628]: pd.crosstab(y_train,dt_1.predict(x_train))
Out[628]:
col_0            0    1
Taxable.Income
0              210    0
1                0  240

In [629]: pd.crosstab(y_test, dt_1.predict(x_test))
Out[629]:
col_0           0   1
Taxable.Income
0              78   0
1               0  72
```

plotting tree

tree.plot_tree(dt_1)



Random forest

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=450,n_jobs=-1,random_state=145,ccp_alpha=0.003)

rf.fit(x_train,y_train)

accuracy_score(y_train,rf.predict(x_train))

accuracy_score(y_test, rf.predict(x_test))

```
accuracy_score(y_train,rf.predict(x_train))
1.0

accuracy_score(y_test, rf.predict(x_test))
1.0
```

pd.crosstab(y_train,rf.predict(x_train))

pd.crosstab(y_test, rf.predict(x_test))

```
In [636]: pd.crosstab(y_train,rf.predict(x_train))
Out[636]:
col_0            0    1
Taxable.Income
0              210    0
1                0  240

In [637]: pd.crosstab(y_test, rf.predict(x_test))
Out[637]:
col_0           0   1
Taxable.Income
0              78   0
1               0  72
```

Summary and inference: -

- Random forest and Decision Tree are performing exactly same on this data set

Note:- Please don’t forget to mention in comments why I am getting 100 percent accuracy in both cross validation techniques