Data Mining and machine Learning

Experiment 8

26 February

Name: Tufan Kundu

Reg no.: 24MDT0184

Decision Tree: Gradient boosting

Q1. Today we will try to see how gradient boosting can be implemented both manually and using the inbuilt classes.

In [10]:

```
## Importing the necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error,r2_score
from sklearn.tree import DecisionTreeRegressor
```

In [11]:

```
## Loading the data
df = pd.read_csv(r"C:\Users\Batch1\Documents\Downloads\Book1.csv")
df
```

Out[11]:

	price	area	bedrooms	bathrooms	stories	parking	furnishingstatus
0	13300000	7420	4	2	3	2	furnished
1	12250000	8960	4	4	4	3	furnished
2	12250000	9960	3	2	2	2	semi-furnished
3	12215000	7500	4	2	2	3	furnished
4	11410000	7420	4	1	2	2	furnished
...	...	...	...	...	...	...	...
244	4550000	5320	3	1	2	0	semi-furnished
245	4550000	5360	3	1	2	2	unfurnished
246	4550000	3520	3	1	1	0	semi-furnished
247	4550000	8400	4	1	4	3	unfurnished
248	4543000	4100	2	2	1	0	semi-furnished

249 rows × 7 columns

In [12]:

```
df.drop('furnishingstatus',axis=1,inplace = True)
```

In [26]:

```
df
```

Out[26]:

	price	area	bedrooms	bathrooms	stories	parking
0	13300000	7420	4	2	3	2
1	12250000	8960	4	4	4	3
2	12250000	9960	3	2	2	2
3	12215000	7500	4	2	2	3
4	11410000	7420	4	1	2	2
...	...	...	...	...	...	...
244	4550000	5320	3	1	2	0
245	4550000	5360	3	1	2	2
246	4550000	3520	3	1	1	0
247	4550000	8400	4	1	4	3
248	4543000	4100	2	2	1	0

249 rows × 6 columns

In [13]:

```
# min max scaling
scaler = MinMaxScaler()
X = scaler.fit_transform(df)
x = X[:,1:]
y = X[:,0]
```

In [23]:

```
# Train test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [24]:

```
## Fitting a decision tree model

model_dtr = DecisionTreeRegressor()
model_dtr.fit(x_train,y_train)
y_pred_dtr = model_dtr.predict(x_test)
```

In [27]:

```
print("MSE(Decision tree regressor):",mean_squared_error(y_test,y_pred_dtr))
print("r2 score:",r2_score(y_test,y_pred_dtr))
```

```
MSE: 0.03492630285149268
r2 score: -0.4198076700398907
```

Now we will see how we can implement the gradient boosting technique with inbuilt class

In [29]:

```
from sklearn.ensemble import GradientBoostingRegressor
gbr = GradientBoostingRegressor(n_estimators=100,learning_rate=0.01,max_depth=3,random_state=0)
gbr.fit(x_train,y_train)
y_pred_gbr = gbr.predict(x_test)

print("MSE(gradient boosting):",mean_squared_error(y_test,y_pred_gbr))
print("r2 score:",r2_score(y_test,y_pred_gbr))
```

```
MSE(gradient boosting): 0.017811289322634986
r2 score: 0.27594382660242045
```

Error reduced significantly by using gradient boosting technique

Perform hyperparameter tuning using GridsearchCV by giving an parameter grid with the hyperparameters n estimators, learning rate and max depth. Each parameter should be having ateast 10 values in the parameter grid.

In [38]:

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators' : [25,50,100,200,250,300,400,450,500,700],
    'learning_rate' : [0.1,0.5,0.01,0.05,0.08,0.001,0.005,0.008,0.0001,0.0005],
    'max_depth' : [1,2,3,4,5,6,7,8,9,10],
}
gbr_model = GradientBoostingRegressor()
grid_search = GridSearchCV(estimator=gbr_model, param_grid=param_grid, cv=5,scoring='neg_mean_squared_error', n_jobs=-1)
grid_search.fit(x_train, y_train)
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_
y_pred = best_model.predict(x_test)
```

In [39]:

```
best_params
```

Out[39]:

```
{'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 700}
```

In [40]:

```
best_model
```

Out[40]:

```
▾ GradientBoostingRegressor
GradientBoostingRegressor(max_depth=1, n_estimators=700)
```

In [41]:

```
print("MSE(optimised after hyperparameter tuning):",mean_squared_error(y_test,y_pred))
print("r2 score:",r2_score(y_test,y_pred))
```

```
MSE(optimised after hyperparameter tuning): 0.018494705889616967
r2 score: 0.24816189709905545
```

Similarly you can import GradientBoostingClassifier from sklearn.ensemble. Use the liver patient dataset and fit a Decision tree and GradientBoostingClassfier.

In [57]:

```
## Loading the dataset
df = pd.read_csv(r"C:\Users\Batch1\Documents\Downloads\liver_patient.csv")
df
```

Out[57]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	liver_disease
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...	...	...
578	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	0
579	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	38	Male	1.0	0.3	216	21	24	7.3	4.4	1.50	0

583 rows × 11 columns

In [58]:

```
## Dropping unnecessary columns
df.drop(['Age','Gender'],axis =1 , inplace = True)
```

In [59]:

```
df
```

Out[59]:

	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	liver_disease
0	0.7	0.1	187	16	18	6.8	3.3	0.90	1
1	10.9	5.5	699	64	100	7.5	3.2	0.74	1
2	7.3	4.1	490	60	68	7.0	3.3	0.89	1
3	1.0	0.4	182	14	20	6.8	3.4	1.00	1
4	3.9	2.0	195	27	59	7.3	2.4	0.40	1
...	...	...	...	...	...	...	...	...	...
578	0.5	0.1	500	20	34	5.9	1.6	0.37	0
579	0.6	0.1	98	35	31	6.0	3.2	1.10	1
580	0.8	0.2	245	48	49	6.4	3.2	1.00	1
581	1.3	0.5	184	29	32	6.8	3.4	1.00	1
582	1.0	0.3	216	21	24	7.3	4.4	1.50	0

583 rows × 9 columns

In [60]:

```
## Min max scaling
X = scaler.fit_transform(df)
x = X[:,:-1]
y = X[:,-1]
```

In [61]:

```
## Train test split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=0)
```

In [63]:

```
## Fitting the model using decision tree classifier and gradient boosting classifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

##Fitting in decision tree classifier
model_dtc = DecisionTreeClassifier()
model_dtc.fit(x_train,y_train)
y_pred_dtc = model_dtc.predict(x_test)
print(f"Accuracy score when fitting the model via decision tree classifier:{accuracy_score(y_test,y_pred_dtc)*100} %")

## Fitting using Gradient boosting classifier
model_gbc = GradientBoostingClassifier(n_estimators=100,learning_rate=0.01,max_depth=3,random_state=0)
model_gbc.fit(x_train,y_train)
y_pred_gbc = model_gbc.predict(x_test)
print(f"Accuracy score when fitting the model via gradient boosting classifier:{accuracy_score(y_test,y_pred_gbc)*100} %")
```

```
Accuracy score when fitting the model via decision tree classifier:61.53846153846154 %
Accuracy score when fitting the model via gradient boosting classifier:65.8119658119658 %
```

Q2. Perform hyperparameter tuning on with the hyperparameters n estimators, learning rate and max depth. Each parameter should be having ateast 10 values in the parameter grid. Find the best combination of the parameters to get the better accuracy.

In [65]:

```
param_grid = {
    'n_estimators' : [25,50,100,200,250,300,400,450,500,700],
    'learning_rate' : [0.1,0.5,0.01,0.05,0.08,0.001,0.005,0.008,0.0001,0.0005],
    'max_depth' : [1,2,3,4,5,6,7,8,9,10],
}
gbc_model = GradientBoostingClassifier()
grid_search = GridSearchCV(estimator=gbc_model, param_grid=param_grid, cv=5,scoring='accuracy', n_jobs=-1)
grid_search.fit(x_train, y_train)
print(f"Best parameters:\n{grid_search.best_params_}")
best_model = grid_search.best_estimator_
print(f"Best model:\n{best_model}")
y_pred = best_model.predict(x_test)
print(f"Accuracy score after hyperparameter tuning:{accuracy_score(y_test,y_pred)*100} %")
```

```
Best parameters:
{'learning_rate': 0.01, 'max_depth': 1, 'n_estimators': 400}
Best model:
GradientBoostingClassifier(learning_rate=0.01, max_depth=1, n_estimators=400)
Accuracy score after hyperparameter tuning:65.8119658119658 %
```