

Div: B

Roll No.: 422004

Gr. No.: 21810939

Name of Student: Aakanksha Bhondve

subject: Advanced Machine Learning

Assignment No. 1

Title: Write a program to do : A dataset collected in a cosmetics shop showing details of customers and whether or not they responded to a special offer to buy a new lip-stick is shown in table below. Use this dataset to build a decision tree, with Buys as the target variable, to help in buying lip-sticks in the future. Find the root node of decision tree. According to the decision tree you have made from previous training data set, what is the decision for the test data: [Age < 21, Income = Low, Gender = Female, Marital Status = Married]?

ID	Age	Income	Gender	Marital Status	Buys
1	< 21	High	Male	Single	No
2	< 21	High	Male	Married	No
3	21-35	High	Male	Single	Yes
4	>35	Medium	Male	Single	Yes
5	>35	Low	Female	Single	Yes
6	>35	Low	Female	Married	No
7	21-35	Low	Female	Married	Yes
8	< 21	Medium	Male	Single	No
9	<21	Low	Female	Married	Yes
10	> 35	Medium	Female	Single	Yes
11	< 21	Medium	Female	Married	Yes
12	21-35	Medium	Male	Married	Yes
13	21-35	High	Female	Single	Yes
14	> 35	Medium	Male	Married	No

Theory:

Machine Learning:

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Types of Machine Learning Algorithms:

1. Supervised Learning

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output. The system creates a model using labeled data to understand the datasets and learn about each data, once the training and processing are done then we test the model by providing a sample data to check whether it is predicting the exact output or not. The goal of supervised learning is to map input data with the output data.

Supervised learning can be grouped further in two categories of algorithms:

- Classification
- Regression

2. Unsupervised Learning

Unsupervised learning is a learning method in which a machine learns without any supervision. The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns. In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data.

It can be further classified into two categories of algorithms:

- Clustering
- Association

3. Reinforcement Learning

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

Supervised Machine Learning: Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y). Supervised learning can be further divided into two types of problems: 1. Regression Regression algorithms are used if there is a relationship between the input variable and the output variable. It is

used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc.

Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

2. Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Decision Tree Working

- **Step 1:**

Begin the tree with the root node, says S, which contains the complete dataset.

- **Step 2:**

Find the best attribute in the dataset using Attribute Selection Measure (ASM).

- **Step 3:**

Divide the S into subsets that contain possible values for the best attributes.

- **Step 4:**

Generate the decision tree node, which contains the best attribute.

- **Step 5:**

Recursively make new decision trees using the subsets of the dataset created in step 3. Continue this process until a stage is reached where you cannot further classify the nodes and call the final node as a leaf node.

Code & Output:

Untitled3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM 8 Disk

Editing

[1] import numpy as np
import pandas as pd

[2] dataset = {
 'Id':[0,1,2,3,4,5,6,7,8,9,10,11,12,13],
 'Age':['<21','<21','21-35','>35','>35','21-35','<21','<21','>35','<21','21-35','21-35','>35'],
 'Income':['High','High','High','Medium','Low','Low','Low','Medium','Low','Medium','Medium','Medium','High','Medium'],
 'Gender':['Male','Male','Male','Female','Female','Female','Female','Male','Female','Female','Male','Female','Male'],
 'MaritalStatus':['Single','Married','Single','Single','Single','Married','Married','Single','Married','Single','Married','Married','Single','Married'],
 'Buys':['No','No','Yes','Yes','No','Yes','No','Yes','No','Yes','Yes','Yes','Yes','No']
}

[4] dataset=pd.DataFrame.from_dict(dataset)
dataset.head()

	Id	Age	Income	Gender	MaritalStatus	Buys
0	0	<21	High	Male	Single	No
1	1	<21	High	Male	Married	No
2	2	21-35	High	Male	Single	Yes
3	3	>35	Medium	Male	Single	Yes
4	4	>35	Low	Female	Single	Yes

[5] dataset.info()

<class 'pandas.core.frame.DataFrame'>

[5] dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
Column Non-Null Count Dtype
--- --
0 Id 14 non-null int64
1 Age 14 non-null object
2 Income 14 non-null object
3 Gender 14 non-null object
4 MaritalStatus 14 non-null object
5 Buys 14 non-null object
dtypes: int64(1), object(5)
memory usage: 800.0+ bytes

x=dataset.iloc[:,1:5]
x

	Age	Income	Gender	MaritalStatus
0	<21	High	Male	Single
1	<21	High	Male	Married
2	21-35	High	Male	Single
3	>35	Medium	Male	Single
4	>35	Low	Female	Single
5	>35	Low	Female	Married

7 <21 Medium Male Single
8 <21 Low Female Married
9 >35 Medium Female Single
10 <21 Medium Female Married
11 21-35 Medium Male Married
12 21-35 High Female Single
13 >35 Medium Male Married

y=dataset.iloc[:,5]
y

0	No
1	No
2	Yes
3	Yes
4	Yes
5	No
6	Yes
7	No
8	Yes
9	Yes
10	Yes
11	Yes
12	Yes
13	No

Name: Buys, dtype: object

```
10 x
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

	Age	Income	Gender	MaritalStatus
0	<21	High	Male	Single
1	<21	High	Male	Married
2	21-35	High	Male	Single
3	>35	Medium	Male	Single
4	>35	Low	Female	Single
5	>35	Low	Female	Married
6	21-35	Low	Female	Married
7	<21	Medium	Male	Single
8	<21	Low	Female	Married
9	>35	Medium	Female	Single
10	<21	Medium	Female	Married
11	21-35	Medium	Male	Married
12	21-35	High	Female	Single
13	>35	Medium	Male	Married

```
[11] from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
```

```
[11] from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
encoded_x=x
for i in x.columns:
    encoded_x[i]=enc.fit_transform(x[i])
print(encoded_x)
```

	Age	Income	Gender	MaritalStatus
0	1	0	1	1
1	1	0	1	0
2	0	0	1	1
3	2	2	1	1
4	2	1	0	1
5	2	1	0	0
6	0	1	0	0
7	1	2	1	1
8	1	1	0	0
9	2	2	0	1
10	1	2	0	0
11	0	2	1	0
12	0	0	0	1
13	2	2	1	0

```
[12] from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(x,y)
y_pred=model.predict(x)
print('Predicted values: ',y_pred)
print('Actual Dependant valyes: ',y)
```

Predicted values: ['No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No']

```
[12] Predicted values: ['No' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No']
Actual Dependant valyes: 0 No
1 No
2 Yes
3 Yes
4 Yes
5 No
6 Yes
7 No
8 Yes
9 Yes
10 Yes
11 Yes
12 Yes
13 No
Name: Buys, dtype: object
```

```
[13] from sklearn.tree import plot_tree
plot_tree(model)
```

[Text(111.60000000000001, 195.696, 'X[0] <= 0.5\\ngini = 0.459\\nsamples = 14\\nvalue = [5, 9]'),
Text(74.4, 152.208, 'gini = 0.0\\nsamples = 4\\nvalue = [0, 4]'),
Text(148.8, 152.208, 'X[2] <= 0.5\\ngini = 0.5\\nsamples = 10\\nvalue = [5, 5]'),
Text(74.4, 108.72, 'X[0] <= 1.5\\ngini = 0.32\\nsamples = 5\\nvalue = [1, 4]'),
Text(37.2, 65.232, 'gini = 0.0\\nsamples = 2\\nvalue = [0, 2]'),
Text(111.60000000000001, 65.232, 'X[3] <= 0.5\\ngini = 0.444\\nsamples = 3\\nvalue = [1, 2]'),
Text(74.4, 21.744, 'gini = 0.0\\nsamples = 1\\nvalue = [1, 0]'),
Text(148.8, 21.744, 'gini = 0.0\\nsamples = 2\\nvalue = [0, 2]'),
Text(223.20000000000002, 108.72, 'X[0] <= 1.5\\ngini = 0.32\\nsamples = 5\\nvalue = [4, 1]'),
Text(186.0, 65.232, 'gini = 0.0\\nsamples = 3\\nvalue = [3, 0]'),
Text(260.40000000000003, 65.232, 'X[3] <= 0.5\\ngini = 0.5\\nsamples = 2\\nvalue = [1, 1]'),
Text(223.20000000000002, 21.744, 'gini = 0.0\\nsamples = 1\\nvalue = [1, 0]'),

