SMAI ASSIGNMENT 1

NAME: ANIMI REDDY
ROLL NUMBER: 20161191

PART1:
Attributes used as Categorichal data: ['Work_accident','promotion_last_5years','sales','salary','left']
```
Accuracy: 0.7624555160142349
Recall: 0.001869158878504673
Precision: 0.9999999999999998
F1_score: 0.0037313432835820895
```

PART2:
Attributes used are both Categorichal and numeric.
```
Accuracy: 0.7929181494661922
Recall: 0.4638655462184874
Precision: 0.6202247191011236
F1_score: 0.5307692307692308
```

PART3:
Effectiveness of using Misclassification rate, Gini, Entropy as impurity measures over given dataframe:

Entropy as impurity measure:
```
Accuracy: 0.7624555160142349
Recall: 0.001869158878504673
Precision: 0.9999999999999998
F1_score: 0.0037313432835820895
```

Gini as impurity measure:
```
Accuracy: 0.7614597240765465
Recall: 0.0037174721189591076
Precision: 1.0
F1_score: 0.007407407407407408
```

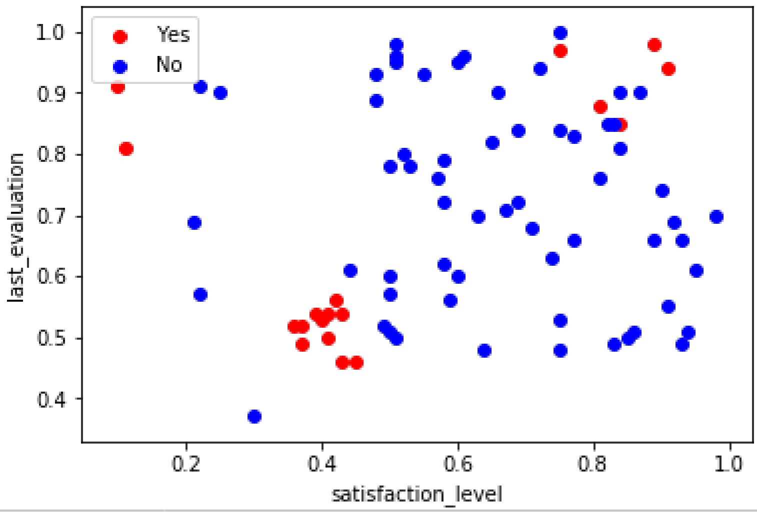
Misclassification rate as impurity measure:
```
Accuracy: 0.7491103202846975
Recall: 0.0035335689045936395
Precision: 1.0
F1_score: 0.007042253521126761
```

PART4:

Visualization of Training data:

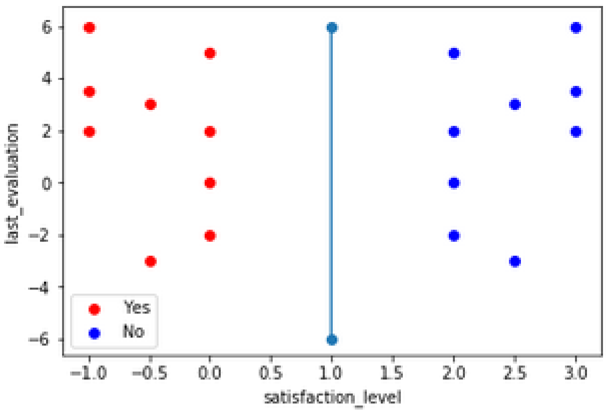
```
In [76]: import matplotlib.pyplot as plt
         df = pd.read_csv('decision_Tree/train.csv')
         df = df.sample(n=80)
         f1 = df[df['left']==1]['satisfaction_level']
         f2 = df[df['left']==1]['last_evaluation']
         plt.scatter(f1,f2,color="red",label="Yes")

         g1 = df[df['left']==0]['satisfaction_level']
         g2 = df[df['left']==0]['last_evaluation']
         plt.scatter(g1,g2,color="blue",label="No")
         plt.xlabel ('satisfaction_level')
         plt.ylabel ('last_evaluation')
         plt.legend()
         plt.show()
```



Visualization of Decision Boundary:

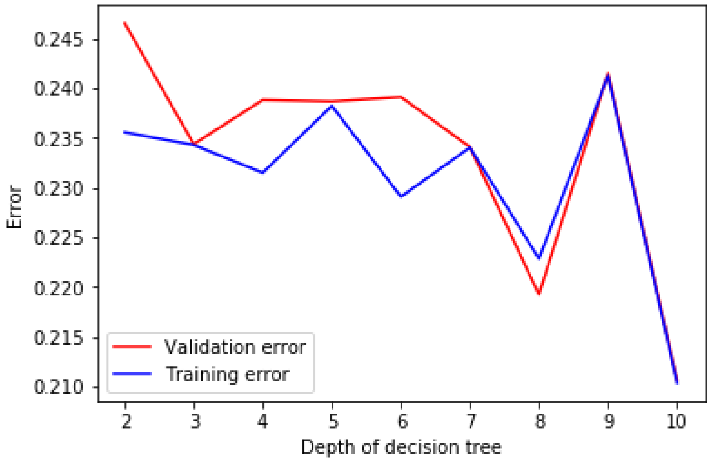
In [3]:
```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log
import matplotlib.pyplot as plt
df = pd.read_csv('inp.csv')
# df = df.sample(n=80)
f1 = df[df['left']==1]['satisfaction_level']
f2 = df[df['left']==1]['last_evaluation']
plt.scatter(f1,f2,color="red",label="Yes")
g1 = df[df['left']==0]['satisfaction_level']
g2 = df[df['left']==0]['last_evaluation']
plt.scatter(g1,g2,color="blue",label="No")
x1, y1 = [1, 1], [-6, 6]
plt.plot(x1,y1, marker = 'o')
plt.xlabel ('satisfaction_level')
plt.ylabel ('last_evaluation')
plt.legend()
plt.show()
```



PART5:

Depth of Decision Tree	Training error	Validation error
10	0.2103724418456845	0.21061999406704246
9	0.24126096351849502	0.24147137347967962
8	0.22282954112113895	0.2192227825571047
7	0.23401550781746538	0.23405517650548802
6	0.22905809075886618	0.23909819044793834
5	0.2382102453285878	0.23864648263579702
4	0.23147324265920932	0.23880154256897068
3	0.234269734333291	0.23435182438445568
2	0.23554086691241893	0.24651438742212994

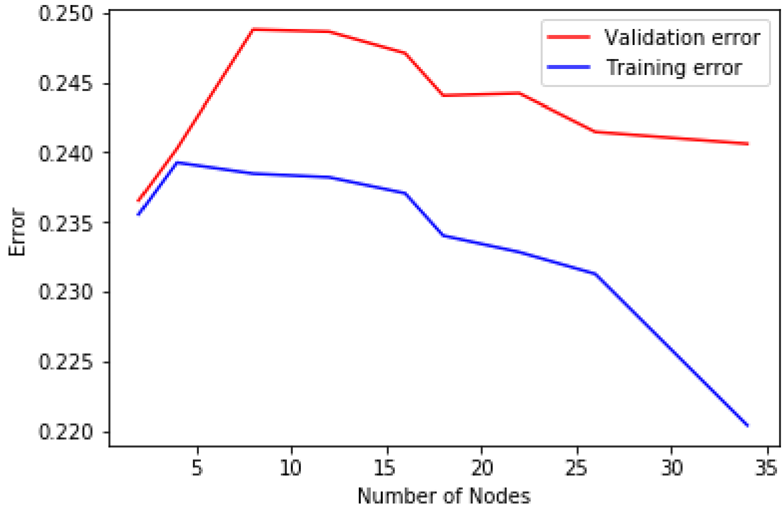
```
In [320]: import matplotlib.pyplot as plt
df = pd.read_csv('tab.csv')
plt.plot(df['d'],df['ve'],color="red",label='Validation error')
plt.plot(df['d'],df['te'],color="blue",label='Training error')
plt.xlabel ('Depth of decision tree')
plt.ylabel ('Error')
plt.legend()
plt.show()
```



Number of Nodes in DT	Training error	Validation error
34	0.2203724418456845	0.24061999406704246
26	0.23126096351849502	0.24147137347967962
22	0.23282954112113895	0.2442227825571047
18	0.23401550781746538	0.24405517650548802
16	0.23705809075886618	0.24709819044793834
12	0.2382102453285878	0.24864648263579702
8	0.23847324265920932	0.24880154256897068
4	0.239269734333291	0.240269734333291
2	0.23554086691241893	0.23654086691241893

In [86]:

```python
import matplotlib.pyplot as plt
df = pd.read_csv('tab1.csv')
plt.plot(df['d'],df['ve'],color="red",label='Validation error')
plt.plot(df['d'],df['te'],color="blue",label='Training error')
plt.xlabel ('Number of Nodes')
plt.ylabel ('Error')
plt.legend()
plt.show()
```



PART6:

Say we built a decision tree of whether one should play or not based on the weather conditions. We may have a training dataset like this:

OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

And use it to build a decision tree that may look something like this:

```
              Outlook
            /   |       \
  overcast /    |sunny   \rain
          /     |          \
     Play    Humidity    Windy
            /   |          |  \
          /     |          |   \
    <=75 /   >75|      true|    \false
```

```
    /       |          |      \
  Play  Don'tPlay Don'tPlay  Play
```

1. Decision Tree Algorithm deals with missing values by returning the probability distribution of the labels under the attribute branch for which the value is missing. Suppose that we had an instance in our test data that showed the outlook to be Sunny but did not have a value for the attribute Humidity. Also, suppose that our training data had **2** instances for which the outlook was Sunny, Humidity was below 75, and a label of Play. Furthermore, suppose the training data had **3** instances where the outlook was Sunny, Humidity was above 75, and had a label of Don't Play. So for the test instance with the missing Humidity attribute, the C4.5 algorithm would return a probability distribution of [0.4, 0.6] corresponding to [Play, Don't Play].