In [6]:

```
import pandas as pd
ad = pd.read_csv("agridata.csv")
ad
```

Out[6]:

	outlook	temp	windy	soil	results
0	sunny	NaN	no	NaN	0
1	NaN	cool	yes	NaN	0
2	rain	hot	NaN	NaN	1
3	rain	warm	NaN	NaN	1
4	sunny	NaN	no	NaN	1
5	NaN	cool	yes	NaN	1
6	overcast	NaN	no	NaN	0

In [7]:

```
# currently we dont have any data of soil.
# All 3 external factors like outlook, temperature and windy are depend upon irrigation.
# if '0' at perform column means end results of cultivation is bad. so we can take soil as infertile
# if '1' at perform column means end results of cultivation is good. so we can take soil as fertile
df = pd.read_csv("agridata2.csv")
df
```

Out[7]:

	outlook	temp	windy	soil	results
0	sunny	NaN	no	infertile	0
1	NaN	cool	yes	infertile	0
2	rain	hot	NaN	fertile	1
3	rain	warm	NaN	fertile	1
4	sunny	NaN	no	fertile	1
5	NaN	cool	yes	fertile	1
6	overcast	NaN	no	infertile	0

In [13]:

```
# In every column there is NaN which means there is no data and that feild is empty
# we can insert some value like "no data" in all empty feilds

inputs = df.drop('results',axis='columns')
target = df['results']
df["outlook"].fillna("No_data", inplace = True)
df["temp"].fillna("No_data", inplace = True)
df["windy"].fillna("No_data", inplace = True)
df
```

Out[13]:

	outlook	temp	windy	soil	results
0	sunny	No data	no	infertile	0
1	No data	cool	yes	infertile	0
2	rain	hot	No data	fertile	1
3	rain	warm	No data	fertile	1
4	sunny	No data	no	fertile	1
5	No data	cool	yes	fertile	1
6	overcast	No data	no	infertile	0

In [14]:

```
# Machine learning algorithms runs on a numbers rather than any given strings
# so we need to convert every strings into number using lab encoder algorithm

from sklearn.preprocessing import LabelEncoder
le_outlook = LabelEncoder()
le_temp = LabelEncoder()
le_windy = LabelEncoder()
le_soil = LabelEncoder()


inputs['outlook_n'] = le_outlook.fit_transform(inputs['outlook'])
inputs['temp_n'] = le_temp.fit_transform(inputs['temp'])
inputs['windy_n'] = le_windy.fit_transform(inputs['windy'])
inputs['soil_n'] = le_soil.fit_transform(inputs['soil'])
inputs
```

Out[14]:

	outlook	temp	windy	soil	outlook_n	temp_n	windy_n	soil_n
0	sunny	No data	no	infertile	3	0	1	1
1	No data	cool	yes	infertile	0	1	2	1
2	rain	hot	No data	fertile	2	2	0	0
3	rain	warm	No data	fertile	2	3	0	0
4	sunny	No data	no	fertile	3	0	1	0
5	No data	cool	yes	fertile	0	1	2	0
6	overcast	No data	no	infertile	1	0	1	1

In [15]:

```
# New encoded columns has been added which represents each string as an unique number on a column
# Now we can remove string values columns as now there is no need to store

inputs_n = inputs.drop(['outlook','temp','windy','soil'],axis='columns')
inputs_n
```

Out[15]:

	outlook_n	temp_n	windy_n	soil_n
0	3	0	1	1
1	0	1	2	1
2	2	2	0	0
3	2	3	0	0
4	3	0	1	0
5	0	1	2	0
6	1	0	1	1

In [16]:

```
# Our target data is results cloumn in which 0 represents bad results and 1 respresents good results
of cultivation.

target
```

Out[16]:

```
0    0
1    0
2    1
3    1
4    1
5    1
6    0
Name: results, dtype: int64
```

In [17]:

```
#Now we are adding Decision tree algorithm to predict from the labeled or given data
#This label data model helps machine learning to train or learn the model so that it can predict acc
urately based on the given sets

from sklearn import tree
model = tree.DecisionTreeClassifier()
model.fit(inputs_n, target)
```

Out[17]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [18]:

```
# Now we can predict our score which will be going to 1
# For complex data set the value score is always less than 1

model.score(inputs_n,target)
```

Out[18]:

```
1.0
```

In [19]:

```
#Now its time to test or predict our data model
# we have to test by assigning our 4 columns value in an array of predict function
# 4 columns are outlook, temperature, windy and soil respectively
# In outlook it is 2 which represents 'rain'
# In Temperature it is 1 which represents 'cool'
# In windy it is 2 which respresents 'yes' means it is windy
# In soil it is 1 which represents infertile
# so our target results will be 0 which means end result of cultivation is not good on that day
model.predict([[2,1,1,1]])
```

Out[19]:

```
array([0], dtype=int64)
```

In [20]:

```
# we can follow the same method by taking another data values
# For better understanding, just take a look at output row [14] to take data values.
model.predict([[2,0,0,0]])
```

Out[20]:

```
array([1], dtype=int64)
```

In [21]:

```
model.predict([[2,1,1,0]])
```

Out[21]:

```
array([1], dtype=int64)
```

In [22]:

```
model.predict([[0,0,0,0]])
```

Out[22]:

```
array([1], dtype=int64)
```

In [ ]: