Practical 3:-

- **Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.**

Plant Flower Iris Detection DECISIONTREE

Leaf Species Detection | DECISION TREE

ENTROPY:

Entropy measures the impurity of a collection of examples.

$$Entropy\ (S) \equiv -p_{\oplus} log_2 p_{\oplus} - p_{\ominus} log_2 p_{\ominus}$$

Where, p_+ is the proportion of positive examples in S
p_- is the proportion of negative examples in S.

INFORMATION GAIN:

- ***Information gain,*** is the expected reduction in entropy caused by partitioning the examples according to this attribute.
- The information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Import basic Libraries

```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
```

Load Dataset

```
dataset = load_iris()
```

Summarize Dataset

```
print(dataset.data)
print(dataset.target)

print(dataset.data.shape)
```

```
       [5.1 2.5 3.  1.1]
       [5.7 2.8 4.1 1.3]
       [6.3 3.3 6.  2.5]
       [5.8 2.7 5.1 1.9]
       [7.1 3.  5.9 2.1]
       [6.3 2.9 5.6 1.8]
       [6.5 3.  5.8 2.2]
       [7.6 3.  6.6 2.1]
       [4.9 2.5 4.5 1.7]
       [7.3 2.9 6.3 1.8]
       [6.7 2.5 5.8 1.8]
       [7.2 3.6 6.1 2.5]
       [6.5 3.2 5.1 2. ]
       [6.4 2.7 5.3 1.9]
       [6.8 3.  5.5 2.1]
       [5.7 2.5 5.  2. ]
       [5.8 2.8 5.1 2.4]
       [6.4 3.2 5.3 2.3]
       [6.5 3.  5.5 1.8]
       [7.7 3.8 6.7 2.2]
       [7.7 2.6 6.9 2.3]
       [6.  2.2 5.  1.5]
       [6.9 3.2 5.7 2.3]
       [5.6 2.8 4.9 2. ]
       [7.7 2.8 6.7 2. ]
       [6.3 2.7 4.9 1.8]
       [6.7 3.3 5.7 2.1]
       [7.2 3.2 6.  1.8]
       [6.2 2.8 4.8 1.8]
       [6.1 3.  4.9 1.8]
       [6.4 2.8 5.6 2.1]
       [7.2 3.  5.8 1.6]
       [7.4 2.8 6.1 1.9]
       [7.9 3.8 6.4 2. ]
       [6.4 2.8 5.6 2.2]
       [6.3 2.8 5.1 1.5]
       [6.1 2.6 5.6 1.4]
       [7.7 3.  6.1 2.3]
       [6.3 3.4 5.6 2.4]
       [6.4 3.1 5.5 1.8]
       [6.  3.  4.8 1.8]
       [6.9 3.1 5.4 2.1]
       [6.7 3.1 5.6 2.4]
       [6.9 3.1 5.1 2.3]
       [5.8 2.7 5.1 1.9]
       [6.8 3.2 5.9 2.3]
       [6.7 3.3 5.7 2.5]
       [6.7 3.  5.2 2.3]
       [6.3 2.5 5.  1.9]
       [6.5 3.  5.2 2. ]
       [6.2 3.4 5.4 2.3]
       [5.9 3.  5.1 1.8]]
      [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
       0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
       1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
       2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
       2 2]
      (150, 4)
```

Segregate Dataset into X(Input/IndependentVariable) & Y(Output/DependentVariable)

```
Y = pd.DataFrame(dataset.target)
Y
```

	0
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

150 rows × 1 columns

Next steps: Generate code with Y View recommended plots

```
X = pd.DataFrame(dataset.data, columns=dataset.feature_names)
X
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

Next steps: Generate code with X View recommended plots

Splitting Dataset into Train & Test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state = 0)
print(X_train.shape)
print(X_test.shape)
```

```
(112, 4)
(38, 4)
```

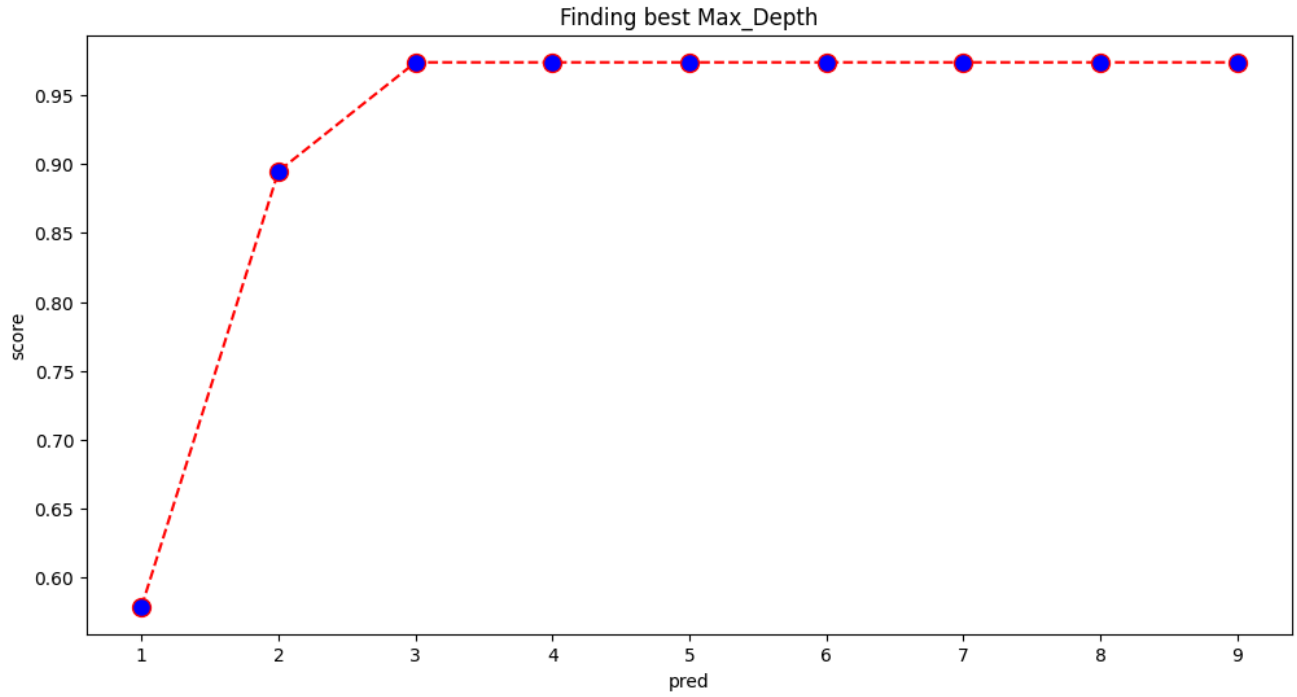
Finding best max_depth Value

```
accuracy = []
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

for i in range(1, 10):
    model = DecisionTreeClassifier(max_depth = i, random_state = 0)
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    score = accuracy_score(y_test, pred)
    accuracy.append(score)

plt.figure(figsize=(12, 6))
plt.plot(range(1, 10), accuracy, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Finding best Max_Depth')
plt.xlabel('pred')
plt.ylabel('score')
```

```
Text(0, 0.5, 'score')
```



Training

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion = 'entropy',max_depth = 3, random_state = 0)
model.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

Prediction

```
y_pred = model.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[2 2]
 [1 1]
 [0 0]
 [2 2]
 [0 0]
 [2 2]
 [0 0]
 [1 1]
 [1 1]
 [1 1]
 [2 2]
 [1 1]
```

```
[1 1]
[1 1]
[1 1]
[0 0]
[1 1]
[1 1]
[0 0]
[0 0]
[2 2]
[1 1]
[0 0]
[0 0]
[2 2]
[0 0]
[0 0]
[1 1]
[1 1]
[0 0]
[2 2]
[1 1]
[0 0]
[2 2]
[2 2]
[1 1]
[0 0]
[2 1]]
```

Accuracy Score

```
from sklearn.metrics import accuracy_score
print("Accuracy of the Model: {0}%".format(accuracy_score(y_test, y_pred)*100))
```

```
Accuracy of the Model: 97.36842105263158%
```