Experiment No – 12

Random Forest Classifier

1. Aim of the Experiment:

Implement and demonstrate the working of Random Forest classifier using sample data sets. Build the model to classify a test sample.

Listing 1:

Sample Dataset Used: **Table 12.1**

Table 12.1 Training Dataset

S.No	CGPA	Interactiveness	Communication Skills	Practical Knowledge	Job Offer
1	≥9	Yes	Good	Good	Yes
2	<9	No	Moderate	Good	Yes
3	≥9	No	Moderate	Average	No
4	<9	No	Moderate	Average	No
5	≥9	Yes	Moderate	Good	Yes

3**. Python Program with Explanation: Random Forest Classifier**

1. Import the library ‘pandas’ to create a Data frame which is a two-dimensional data structure.

```
import pandas
```

2. Import LabelEncoder to normalize labels.

```
from sklearn.preprocessing import LabelEncoder
```

3. Import train_test_split function.

```
from sklearn.model_selection import train_test_split
```

4. Import RandomForestClassifier from sklearn.ensemble

```
from sklearn.ensemble import RandomForestClassifier
```

5. Create a list ‘data’ with the sample dataset.

```
data = {'CGPA':['g9','l9','g9','l9','g9'],
        'Inter':['Y','N','N','N','Y'],
        'PK':['++','==','==','==','=='],
        'CS':['G','G','A','A','G'],
        'Job':['Y','Y','N','N','Y']}
```

6. Create pandas dataframe “table” using the structure DataFrame with the given dataset ‘data'.

```
table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
```

7. Use a value ["CGPA"]=="g9" in the table to select matching row and count the number of columns.

```
table.where(table["CGPA"]=="g9").count()
```

8. Use LabelEncoder() to encode target labels with value between 0 and no_of_classes-1.

```
encoder=LabelEncoder()
```

9. Then transform non-numerical labels to numerical labels.

```
for i in table:
   table[i]=encoder.fit_transform(table[i])
```

10. Use iloc property to select by position.

Select the columns until (excluding) the fifth column.

```
X=table.iloc[:,0:4].values
```

Select the fifth column

```
y=table.iloc[:,4].values
```

11. Split the dataset into training dataset and test dataset by using the function train_test_split().

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=2)
```

12. Use RandomForestClassifier class. The most important parameter used is n_estimators.
    - n_estimators is used to denote the number of trees in the forest.

```
model = RandomForestClassifier(n_estimators=3)
```

13. Then train the classifier using the function fit().

```
model.fit(X_train,y_train)
```

14. After training, the fitted model can be used to predict a new instance.

The non-numerical equivalent of the new instance [0, 1, 1, 1] given is [‘g9’, ‘Y’, ‘==’, ‘G’]

```
print([0,1,1,1])
if model.predict([[0,1,1,1]])==1:
    print("Got JOB")
else:
    print("Didn't get JOB")
```

The non-numerical equivalent of the new instance [0, 0, 1, 0] given is [g9’, ‘N’, ‘==’, ‘A’]

```
print([0,0,1,0])
if model.predict([[0,0,1,0]])==1:
    print("Got JOB")
else:
    print("Didn't get JOB")
```

Complete Program: Random Forest Classifier

```
import pandas
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestClassifier

data = {'CGPA':['g9','l9','g9','l9','g9'],
        'Inter':['Y','N','N','N','Y'],
        'PK':['++','==','==','==','=='],
        'CS':['G','G','A','A','G'],
        'Job':['Y','Y','N','N','Y']}

table=pandas.DataFrame(data,columns=["CGPA","Inter","PK","CS","Job"])
table.where(table["CGPA"]=="g9").count()
encoder=LabelEncoder()

for i in table:
    table[i]=encoder.fit_transform(table[i])

X=table.iloc[:,0:4].values
y=table.iloc[:,4].values

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=2)
model = RandomForestClassifier(n_estimators=3)
model.fit(X_train,y_train)

print("\n")
print([0,1,1,1])
if model.predict([[0,1,1,1]])==1:
    print("Got JOB")
else:
    print("Didn't get JOB")

print("\n")
print([0,0,1,0])
if model.predict([[0,0,1,0]])==1:
    print("Got JOB")
else:
    print("Didn't get JOB")
```

Output:

Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:37:02) [MSC v.1924 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

=== RESTART: C:\Users\ADMIN\pythonpgms\final\jnf randomforest classifier.py ===

[0, 1, 1, 1]

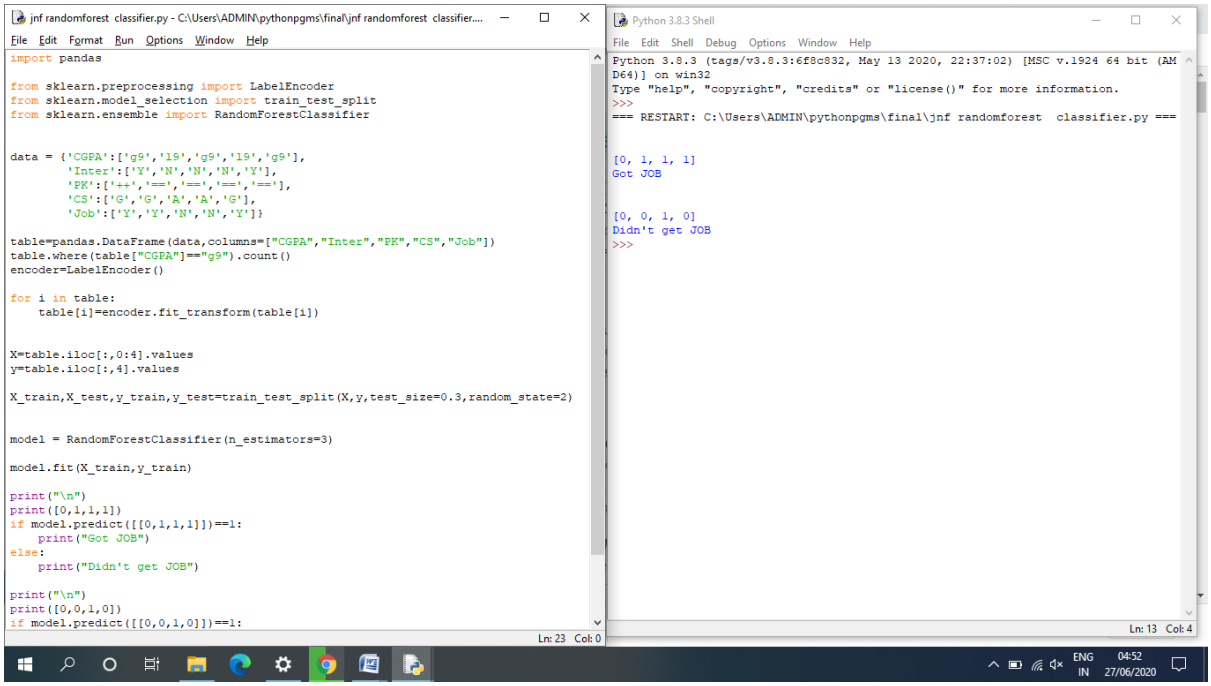
Got JOB

[0, 0, 1, 0]

Didn't get JOB

>>>

Screen Shot of the Output:



Listing 2:

Program Code:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

#Load the Iris data set.
dataset = pd.read_csv("Iris.csv")

#  Split the Iris features into input and output columns
X = dataset.iloc[:, 1:5].values
y = dataset.iloc[:, 5].values
print(" Training Dataset\n")
print("\n", X)
print("\n", y)

# Split the data matrix into train and test dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

#Train the model using RandomForestClassifier
model= RandomForestClassifier(n_estimators=20)

#Fit the model
model.fit(X_train,y_train)

#Predict the Output
y_pred = model.predict(X_test)
#Print the model accuracy

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Screen Shot of the program

jnf randomforest scalar iris.py - C:/Users/ADMIN/pythonpgms/Review/jnf randomforest scalar iris.py (3.8.3)

```python
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn import metrics

dataset = pd.read_csv("Iris.csv")


X = dataset.iloc[:, 1:5].values
y = dataset.iloc[:, 5].values
print(" Training Dataset\n")
print("\n", X)
print("\n", y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

model= RandomForestClassifier(n_estimators=20)

model.fit(X_train,y_train)

y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Python 3.8.3 Shell

```
Squeezed text (150 lines).

['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica']
Accuracy: 1.0
>>>
```