Advance Software Engineering (SE-406)

LAB A1-G3

Laboratory Manual

Department of Software Engineering

DELHI TECHNOLOGICAL UNIVERSITY(DTU)

Shahbad Daulatpur, Bawana Road, Delhi-110042

Submitted to: -	**Submitted by:-**
Ms. Parul Sharma	Name: ASHISH KUMAR
	Roll number: 2K18/SE/041

EXPERIMENT 10

- ASHISH KUMAR
- 2K18/SE/041

Aim:- To train and test fault prediction model using Random Forest.

Introduction:- Technically it is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator. Each tree depends on an independent random sample. It is simpler and more powerful compared to the other non-linear classification algorithms.

Advantages:

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems.
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

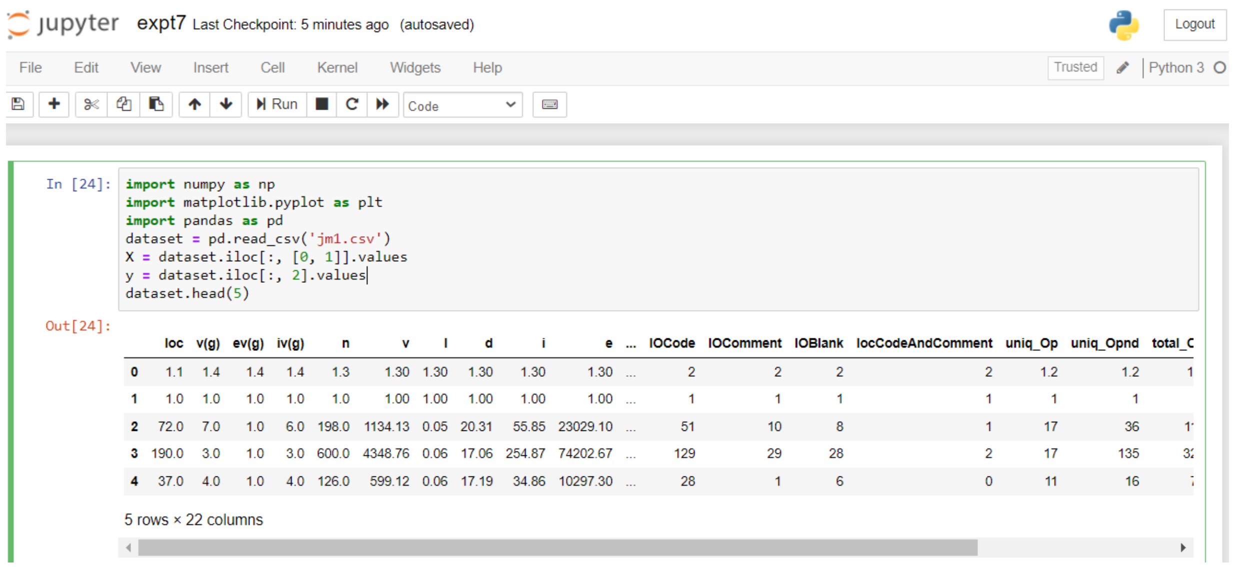
Disadvantages:

- Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

Code & Output:-

Link to dataset: http://promise.site.uottawa.ca/SERepository/datasets/cm1.arff
This is a PROMISE data set made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering.

```
# Importing the dataset
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset = pd.read_csv('jm1.csv')
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, 2].values
dataset.head(5)
```



```
In [24]: import numpy as np
         import matplotlib.pyplot as plt
         import pandas as pd
         dataset = pd.read_csv('jm1.csv')
         X = dataset.iloc[:, [0, 1]].values
         y = dataset.iloc[:, 2].values
         dataset.head(5)
```

Out[24]:

	loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	...	lOCode	lOComment	lOBlank	locCodeAndComment	uniq_Op	uniq_Opnd	total_C
0	1.1	1.4	1.4	1.4	1.3	1.30	1.30	1.30	1.30	1.30	...	2	2	2	2	1.2	1.2	1
1	1.0	1.0	1.0	1.0	1.0	1.00	1.00	1.00	1.00	1.00	...	1	1	1	1	1	1	
2	72.0	7.0	1.0	6.0	198.0	1134.13	0.05	20.31	55.85	23029.10	...	51	10	8	1	17	36	1
3	190.0	3.0	1.0	3.0	600.0	4348.76	0.06	17.06	254.87	74202.67	...	129	29	28	2	17	135	3
4	37.0	4.0	1.0	4.0	126.0	599.12	0.06	17.19	34.86	10297.30	...	28	1	6	0	11	16	7

5 rows × 22 columns

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
x_train, x_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=1)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

#Applying Random Forest Classifier on the Training Set

from sklearn.ensemble import RandomForestClassifier

creating a RF classifier

clf = RandomForestClassifier(n_estimators = 100)

Training the model on the training dataset

clf.fit(x_train, y_train)

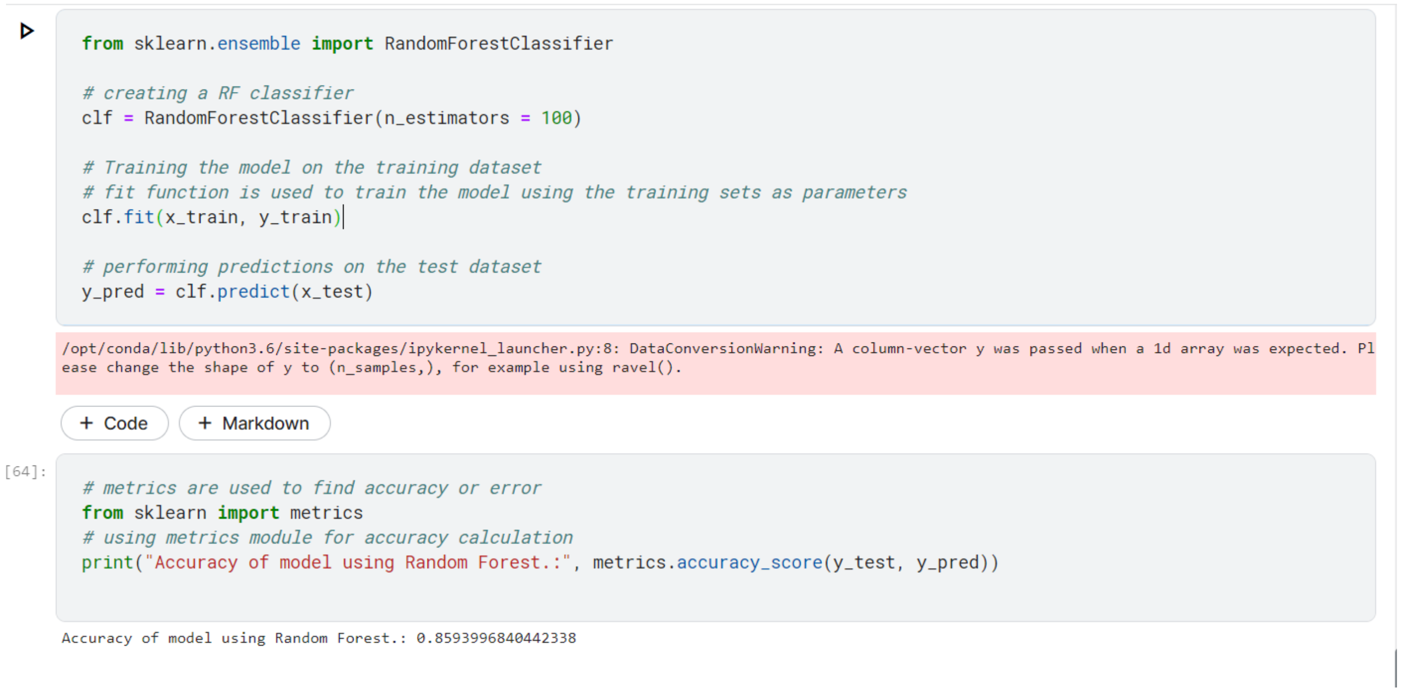
y_pred = clf.predict(x_test)

# Model Accuracy: how often is the classifier correct?

from sklearn import metrics

using metrics module for accuracy calculation

print("Accuracy of model using Random Forest:", metrics.accuracy_score(y_test, y_pred))



```
from sklearn.ensemble import RandomForestClassifier

# creating a RF classifier
clf = RandomForestClassifier(n_estimators = 100)

# Training the model on the training dataset
# fit function is used to train the model using the training sets as parameters
clf.fit(x_train, y_train)

# performing predictions on the test dataset
y_pred = clf.predict(x_test)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:8: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Pl
ease change the shape of y to (n_samples,), for example using ravel().
```

```
# metrics are used to find accuracy or error
from sklearn import metrics
# using metrics module for accuracy calculation
print("Accuracy of model using Random Forest.:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy of model using Random Forest.: 0.8593996840442338
```

<u>Result:</u>- The accuracy of the Random Forest classifier comes out to be 85.93%.

<u>Learning from experiment:</u>- We have successfully been able to train and test fault prediction model using Random Forest and learnt its advantages as well as its disadvantages.