Random Forest Classifier and Random Forest Regressor

Aim of the Exercise:

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

****

Exercise- 1:

Sample Dataset: IRIS or any general dataset.

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 RandomForestRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
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
```

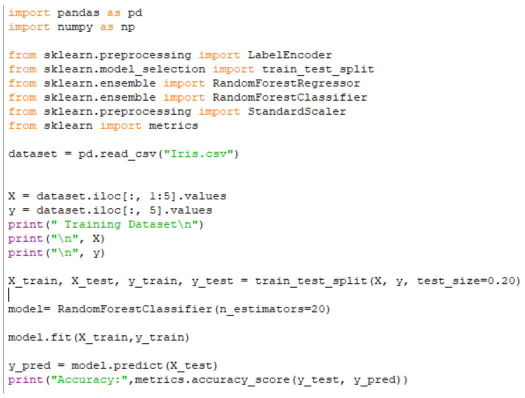
```
model= RandomForestClassifier(n_estimators=20)

#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



```
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))
```


```
['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
>>>
```

Exercise 2:

Random Forest Regression Model

II. Regression problems can also be solved by RandomForestRegressor class of the sklearn.ensemble library.

Training Dataset

CGPA	Assessment	Project	Result
9.2	85	8	95
8	80	7	82
8.5	81	8	86
6	45	5	60
6.5	50	4	55
8.2	72	7	78

5.8	38	5	48
8.9	91	9	80

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

1. Import the library 'pandas' to read data from a CSV file.

```
import pandas as pd
```

2. Import numpy

```
import numpy as np
```

3. Import LabelEncoder to normalize labels.

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

4. Import train_test_split function.

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

5. Import RandomForestRegressor from sklearn.ensemble

```
from sklearn.ensemble import RandomForestRegressor
```

6. Import StandardScaler to apply scaling transformations.

```
from sklearn.preprocessing import StandardScaler
```

7. Import metrics to measure the prediction error.

```
from sklearn import metrics
```

8. Load data from a CSV file into a Pandas DataFrame.

```
dataset = pd.read_csv("randomforest csv.csv")
```

9. Use iloc property to select by position.

Select the columns until (excluding) the last column. Then print it.

```
X = dataset.iloc[:, :-1].values
```

Select the fourth column and then print it.

```
y = dataset.iloc[:, 3].values
```

```
print(" Training Dataset\n")
print("\n", X)
print("\n", y)
```

10. 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=0.20)
```

11. Create a new instance of StandardScaler and then fit and transform the scaler to X_train and X_test. Standardize X_train and X_test by computing the mean and standard deviation by the transform() function on a training set so as to later reapply the same transformation on the testing set.

```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

12. Use RandomForestRegressor model. The most important parameter used is n_estimators.

- n_estimators is used to denote the number of trees in the forest.

```
model= RandomForestRegressor(n_estimators=20, random_state=0)
```

13. The model takes as input two arrays: an array X_train, holding the training instances, and an array y_train holding the class labels for the training instances. Then train the classifier using the function fit().

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

14. To make predictions, the predict method of the RandomForestRegressor class is used.

```
y_pred = model.predict(X_test)
```

15. Print Mean Absolute Error and Mean Squared Error.

```
print('Mean Absolute Error:',
metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:',
metrics.mean_squared_error(y_test, y_pred))
```

16. Predict the output for the test sample. [CGPA: 9.1, Assessment: 85, Project: 8]

```
print([9.1,85,8])
predicted = model.predict([[9.1,85,8]])
print(predicted)
```

Complete Program: Random Forest Regressor

```
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.preprocessing import StandardScaler
from sklearn import metrics

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

X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 3].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)
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
```

```
X_test = scaler.transform(X_test)

model= RandomForestRegressor(n_estimators=20, random_state=0)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))
#print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

print([9.1,85,8])
predicted = model.predict([[9.1,85,8]])
print(predicted)
```

Output: Random Forest Regressor

```
====== RESTART: randomforest_regressor.py =====
 Training Dataset

 [[ 9.2 85.   8. ]
 [ 8.  80.   7. ]
 [ 8.5 81.   8. ]
 [ 6.  45.   5. ]
 [ 6.5 50.   4. ]
 [ 8.2 72.   7. ]
 [ 5.8 38.   5. ]
 [ 8.9 91.   9. ]]

 [95 82 86 60 55 78 48 80]
Mean Absolute Error: 6.175000000000001
Mean Squared Error: 62.88124999999999
[9.1, 85, 8]
[90.95]
>>>
```

Screen Shot of the Output: Random Forest Regressor

```
File  Edit  Format  Run  Options  Window  Help
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.preprocessing import StandardScaler
from sklearn import metrics

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


X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 3].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)
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

model= RandomForestRegressor(n_estimators=20, random_state=0)

model.fit(X_train,y_train)

y_pred = model.predict(X_test)

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))

print([9.1,85,8])
predicted = model.predict([[9.1,85,8]])
print(predicted)
```

```
 Training Dataset


 [[ 9.2 85.   8. ]
 [ 8.  80.   7. ]
 [ 8.5 81.   8. ]
 [ 6.  45.   5. ]
 [ 6.5 50.   4. ]
 [ 8.2 72.   7. ]
 [ 5.8 38.   5. ]
 [ 8.9 91.   9. ]]

 [95 82 86 60 55 78 48 80]
Mean Absolute Error: 6.175000000000001
Mean Squared Error: 62.88124999999999
[9.1, 85, 8]
[90.95]
>>>
```