Experiment No. 4
Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

Aim: Apply Random Forest Algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: Able to perform various feature engineering tasks, apply Random Forest Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score.

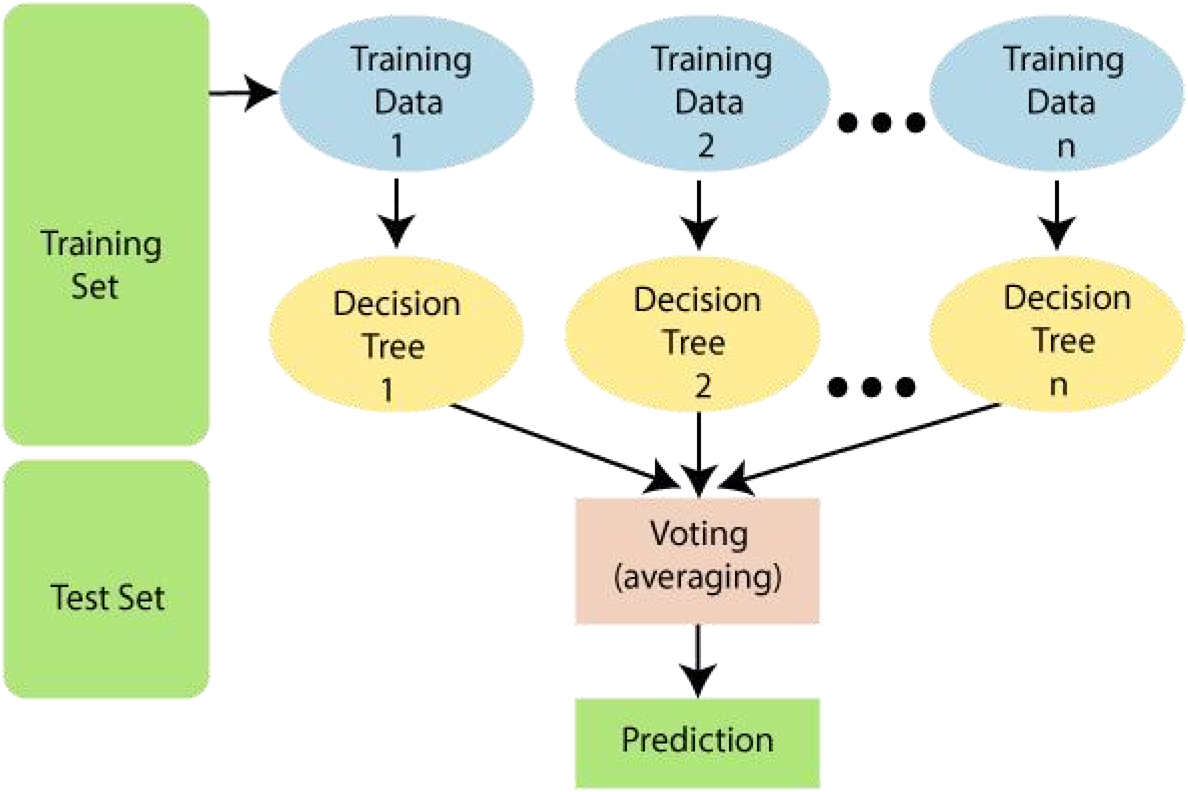
Theory:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



Dataset:

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad &Tobago, Peru, Hong, Holand-Netherlands.

Code:

Conclusion:

1. State the observations about the data set from the correlation heat map.

The correlation heatmap offers valuable insights into the interplay among different dataset features. These observations provide us with an understanding of how various attributes may or may not be interconnected. However, it's noteworthy that the majority of the observed correlations are relatively weak, suggesting that these connections may have limited influence on the associated variables.

The correlation coefficient between age and education.num is approximately 0.0365. This suggests a very weak positive correlation

Age and fnlwgt exhibit a weak negative correlation of approximately -0.0766. This implies that, on average, younger individuals may have slightly higher final weight values.

2. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.
   - **Accuracy**: The model is 85.33% accurate in predicting income levels.
   - **Confusion Matrix**: It correctly identifies 1435 '>50K' income instances and 6287 '<=50K' income instances.
   - **Precision**: For '>50K', it's 74%, and for '<=50K', it's 88%.
   - **Recall**: For '>50K', it's 63%, and for '<=50K', it's 93%.
   - **F1 Score**: The weighted F1 score is 0.85, indicating a good balance of precision and recall.

   In summary, the model is reasonably accurate, with a focus on correctly classifying '<=50K' income. It can be fine-tuned for better performance on '>50K' income predictions if needed.

3. Compare the results obtained by applying random forest and decision tree algorithm on the Adult Census Income Dataset

In comparison, If you are looking for a simple, interpretable model and have a relatively small dataset, a Decision Tree may be suitable. However, if you prioritize predictive accuracy, want to reduce overfitting, or have a larger dataset, Random Forest is often a better choice. Random Forest tends to yield more robust and accurate results, making it a popular choice in many machine learning applications Random Forest tends to provide better results than a Decision Tree.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

data = pd.read_csv('/adult.csv')
```

```
print(data)
```

```
       age workclass  fnlwgt     education  education.num      marital.status  \
0       90         ?   77053       HS-grad              9             Widowed
1       82   Private  132870       HS-grad              9             Widowed
2       66         ?  186061  Some-college             10             Widowed
3       54   Private  140359       7th-8th              4            Divorced
4       41   Private  264663  Some-college             10           Separated
...    ...       ...     ...           ...            ...                 ...
32556   22   Private  310152  Some-college             10       Never-married
32557   27   Private  257302    Assoc-acdm             12  Married-civ-spouse
32558   40   Private  154374       HS-grad              9  Married-civ-spouse
32559   58   Private  151910       HS-grad              9             Widowed
32560   22   Private  201490       HS-grad              9       Never-married

              occupation   relationship   race     sex  capital.gain  \
0                      ?  Not-in-family  White  Female             0
1        Exec-managerial  Not-in-family  White  Female             0
2                      ?      Unmarried  Black  Female             0
3      Machine-op-inspct      Unmarried  White  Female             0
4         Prof-specialty      Own-child  White  Female             0
...                  ...            ...    ...     ...           ...
32556    Protective-serv  Not-in-family  White    Male             0
32557       Tech-support           Wife  White  Female             0
32558  Machine-op-inspct        Husband  White    Male             0
32559       Adm-clerical      Unmarried  White  Female             0
32560       Adm-clerical      Own-child  White    Male             0

       capital.loss  hours.per.week native.country income
0              4356              40  United-States  <=50K
1              4356              18  United-States  <=50K
2              4356              40  United-States  <=50K
3              3900              40  United-States  <=50K
4              3900              40  United-States  <=50K
...             ...             ...            ...    ...
32556             0              40  United-States  <=50K
32557             0              38  United-States  <=50K
32558             0              40  United-States   >50K
32559             0              40  United-States  <=50K
32560             0              20  United-States  <=50K

[32561 rows x 15 columns]
```

```
data.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	30162.000000	3.016200e+04	30162.000000	30162.000000	30162.000000	30162.000000
mean	38.437902	1.897938e+05	10.121312	1092.007858	88.372489	40.931238
std	13.134665	1.056530e+05	2.549995	7406.346497	404.298370	11.979984
min	17.000000	1.376900e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.176272e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.784250e+05	10.000000	0.000000	0.000000	40.000000
75%	47.000000	2.376285e+05	13.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

```
data.isnull().sum()
```

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
```

```
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

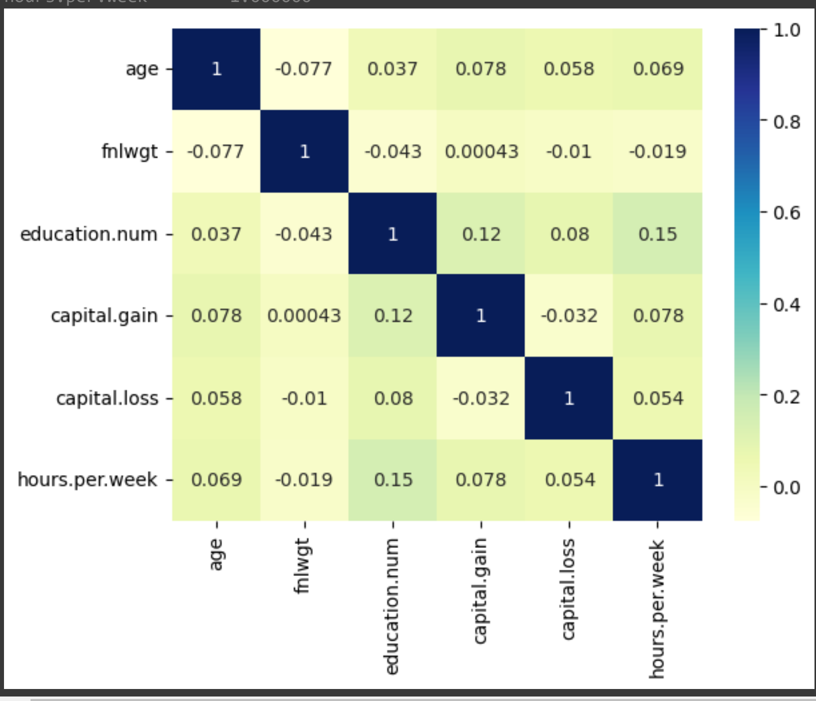
```
import matplotlib.pyplot as mp
import pandas as pd
import seaborn as sb
print(data.corr())

# plotting correlation heatmap
dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)

# displaying heatmap
mp.show()
```

```
<ipython-input-6-b698e0a536da>:4: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  print(data.corr())
<ipython-input-6-b698e0a536da>:7: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ver
  dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)
                     age    fnlwgt  education.num  capital.gain  capital.loss  \
age             1.000000 -0.076646       0.036527      0.077674      0.057775
fnlwgt         -0.076646  1.000000      -0.043195      0.000432     -0.010252
education.num   0.036527 -0.043195       1.000000      0.122630      0.079923
capital.gain    0.077674  0.000432       0.122630      1.000000     -0.031615
capital.loss    0.057775 -0.010252       0.079923     -0.031615      1.000000
hours.per.week  0.068756 -0.018768       0.148123      0.078409      0.054256

                hours.per.week
age                   0.068756
fnlwgt               -0.018768
education.num         0.148123
capital.gain          0.078409
capital.loss          0.054256
hours.per.week        1.000000
```




```
from sklearn.preprocessing import OneHotEncoder

# Handle missing values
data.replace('?', pd.NA, inplace=True)
data.dropna(inplace=True)

# Separate features and target
x = data.drop('income', axis=1)
y = data['income']
```

```
# Separate categorical and numerical columns
categorical_columns = ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', '
numerical_columns = ['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week']

x_categorical = x[categorical_columns]
x_numerical = x[numerical_columns]

# Apply one-hot encoding to categorical features
encoder = OneHotEncoder()
x_categorical_encoded = encoder.fit_transform(x_categorical)

# Combine encoded categorical features with numerical features
import numpy as np
x_encoded = np.hstack((x_categorical_encoded.toarray(), x_numerical))
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_encoded, y, test_size=0.3, random_state=1)
```

```
from sklearn.ensemble import RandomForestClassifier

# Create Random Forest classifier object
clf = RandomForestClassifier(n_estimators=100, random_state=1)

# Train the classifier
clf.fit(x_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=1)
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# Predict on the test set
predictions = clf.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, predictions)
print("Accuracy:", accuracy)

# Generate classification report
report = classification_report(y_test, predictions)
print("Classification Report:\n", report)

# Generate confusion matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:\n", conf_matrix)
```

```
Accuracy: 0.8533539617637308
Classification Report:
               precision    recall  f1-score   support

       <=50K       0.88      0.93      0.90      6781
        >50K       0.74      0.63      0.68      2268

    accuracy                           0.85      9049
   macro avg       0.81      0.78      0.79      9049
weighted avg       0.85      0.85      0.85      9049

Confusion Matrix:
 [[6287  494]
 [ 833 1435]]
```