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.

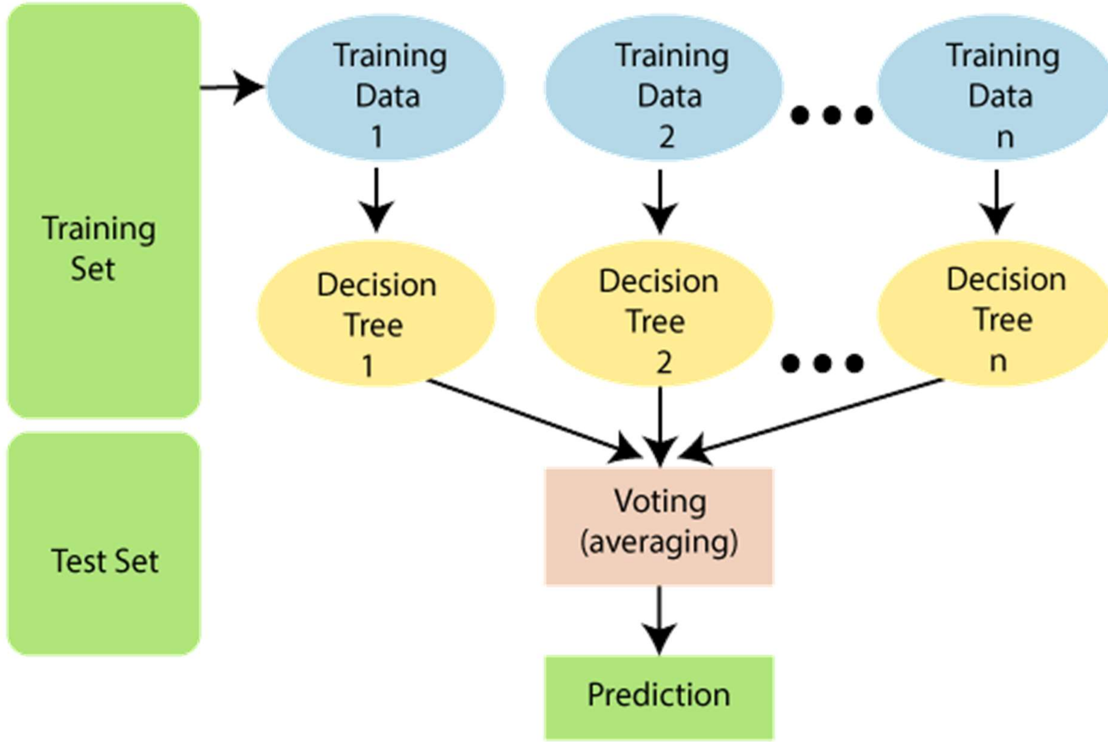
Theory:

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.
    - Age vs. Workclass: There is mild positive correlation (around 0.08) between “Age” and "Workclass". This suggests that, on average, older individuals tend to have higher Workclass.

- Age vs. Education Number: There is a mild positive correlation (around 0.03) between "age" and "education-num." This indicates that, on average, older individuals tend to have slightly higher levels of education.
- Education vs. Education-num: There is a positive correlation (around 0.38) between " Education " and " Education-num." This suggests that higher the education the more higher are the levels of education for the individuals.
- Age vs. Hours per Week: There is a very weak positive correlation (around 0.06) between "age" and "hours.per.week," indicating that older individuals may work slightly more hours per week.
- Education Number vs. Hours per Week: There is a very weak positive correlation (around 0.13) between "education.num" and "hours.per.week," suggesting that individuals with higher education levels might work slightly longer hours.

2. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.
    - Accuracy : 0.78.
    - Confusion matrix : [[6039 907]
      [1134 1123]]
    - Precision : 0.84
    - Recall : 0.87
    - F1 Score : 0.86
3. Compare the results obtained by applying random forest and decision tree algorithm on the Adult Census Income Dataset.
    - The results obtained by applying Decision Tree algorithm and Random Forest algorithm shows some variation in each case. The Random Forest algorithm generally outperforms the Decision Tree algorithm in terms of accuracy, precision, recall, and F1-score on the Adult Census Income Dataset.

```
import pandas as pd
import numpy as np
import seaborn  as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style='white', context='notebook', palette='deep')

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold, learning_curve, train_test_split, KFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

```
df = pd.read_csv('adult.csv')
df.head()
```

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gen
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	M
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	M

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              48842 non-null  int64
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64
 11  capital-loss     48842 non-null  int64
 12  hours-per-week   48842 non-null  int64
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

```
df['capital-gain'].value_counts()
```

```
0        44807
15024      513
7688       410
7298       364
99999      244
         ...
1111         1
7262         1
22040        1
1639         1
2387         1
Name: capital-gain, Length: 123, dtype: int64
```

```
df['capital-loss'].value_counts()
```

```
0       46560
1902      304
1977      253
1887      233
2415       72
        ...
2465        1
2080        1
155         1
1911        1
2201        1
Name: capital-loss, Length: 99, dtype: int64
```

```
df.drop(['fnlwgt','race','capital-gain','capital-loss'],axis=1,inplace=True)
```

```
df.replace('?',np.nan,inplace = True)
df.dropna(inplace=True)
```

```
df.duplicated().sum()
```

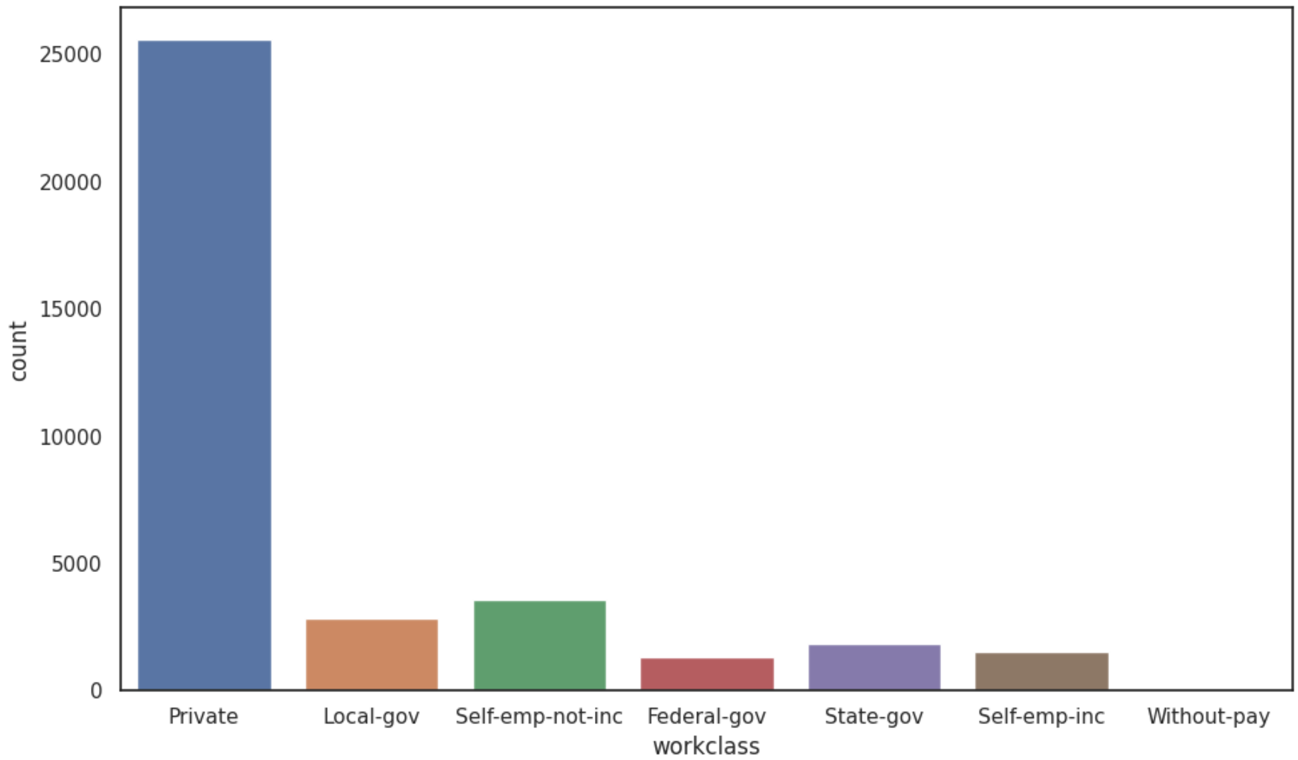
```
8411
```

```
df=df.drop_duplicates()
```

```
df.shape
```

```
(36811, 11)
```

```
plt.figure(figsize=(10, 6))
sns.countplot(df , x='workclass' )
plt.tight_layout()
```



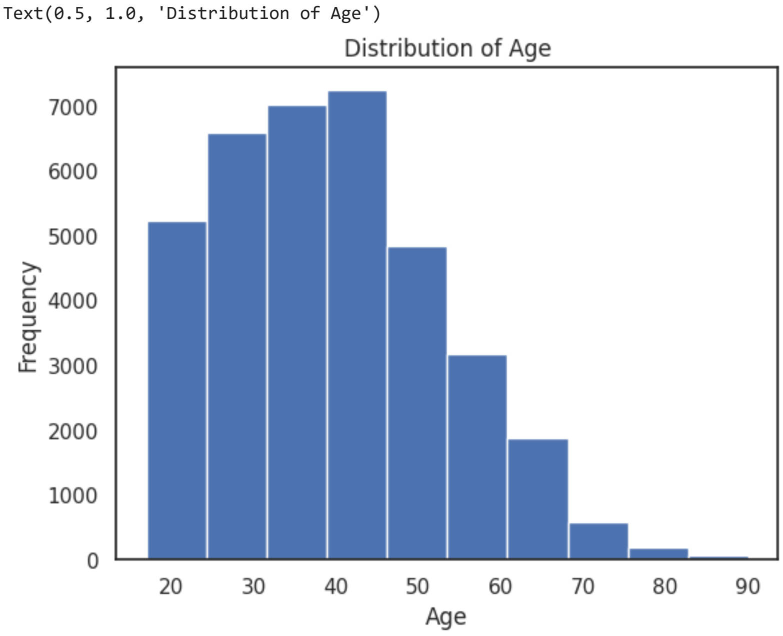

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

df['workclass'] = label_encoder.fit_transform(df['workclass'])
df['marital-status'] = label_encoder.fit_transform(df['marital-status'])
df['occupation'] = label_encoder.fit_transform(df['occupation'])
df['relationship'] = label_encoder.fit_transform(df['relationship'])
df['gender'] = label_encoder.fit_transform(df['gender'])
df['native-country'] = label_encoder.fit_transform(df['native-country'])
df['income'] = label_encoder.fit_transform(df['income'])
df['educational-num']=label_encoder.fit_transform(df['educational-num'])
df['education'] = label_encoder.fit_transform(df['education'])
```

```
df.head()
```

	age	workclass	education	educational-num	marital-status	occupation	relationship	gender	hours-per-week	native-country	income
0	25	2	1	6	4	6	3	1	40	38	0
1	38	2	11	8	2	4	0	1	50	38	0
2	28	1	7	11	2	10	0	1	40	38	1
3	44	2	15	9	2	6	0	1	40	38	1
5	34	2	0	5	4	7	1	1	30	38	0

```
plt.hist(df['age'])
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Age')
```

```
Text(0.5, 1.0, 'Distribution of Age')
```




```
df['income'] = df['income'].astype('category')
```

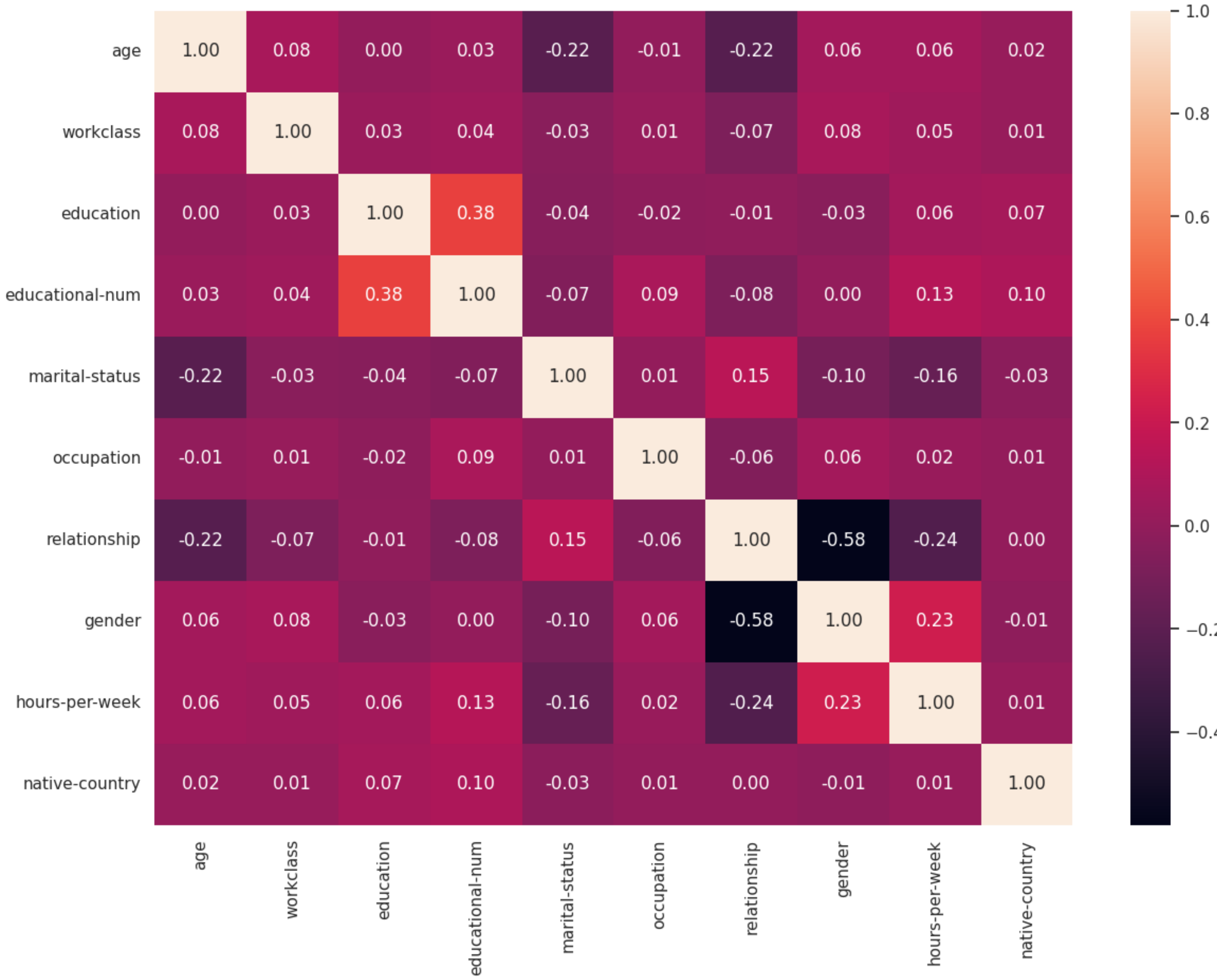
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 36811 entries, 0 to 48841
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              36811 non-null  int64
```

```
 1   workclass        36811 non-null  int64
 2   education        36811 non-null  int64
 3   educational-num  36811 non-null  int64
 4   marital-status   36811 non-null  int64
 5   occupation       36811 non-null  int64
 6   relationship     36811 non-null  int64
 7   gender           36811 non-null  int64
 8   hours-per-week   36811 non-null  int64
 9   native-country   36811 non-null  int64
 10  income           36811 non-null  category
dtypes: category(1), int64(10)
memory usage: 3.1 MB
```

```
plt.figure(figsize=(14,10))
sns.heatmap(df.corr(),annot=True,fmt='.2f')
plt.show()
```

```
<ipython-input-107-ce7fadf8682c>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will defa
  sns.heatmap(df.corr(),annot=True,fmt='.2f')
```




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

```
X = df.drop('income',axis=1)
```

```
y = df['income']
```

```
X.head(3)
```

	age	workclass	education	educational-num	marital-status	occupation	relationship	gender	hours-per-week	native-country
0	25	2	1	6	4	6	3	1	40	38
1	38	2	11	8	2	4	0	1	50	38
2	28	1	7	11	2	10	0	1	40	38

```
y.head(3)
```

```
0    0
1    0
2    1
Name: income, dtype: category
Categories (2, int64): [0, 1]
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y)
```

```
X_train.head()
```

	age	workclass	education	educational-num	marital-status	occupation	relationship	gender	hours-per-week	native-country
30376	19	2	15	9	4	5	1	1	45	38
2515	45	4	15	9	2	5	0	1	40	38
27548	45	2	11	8	4	3	1	0	40	38
39539	31	3	11	8	2	3	0	1	50	38

```
test_size = 0.20
seed = 7
num_folds = 10
scoring = 'accuracy'
```

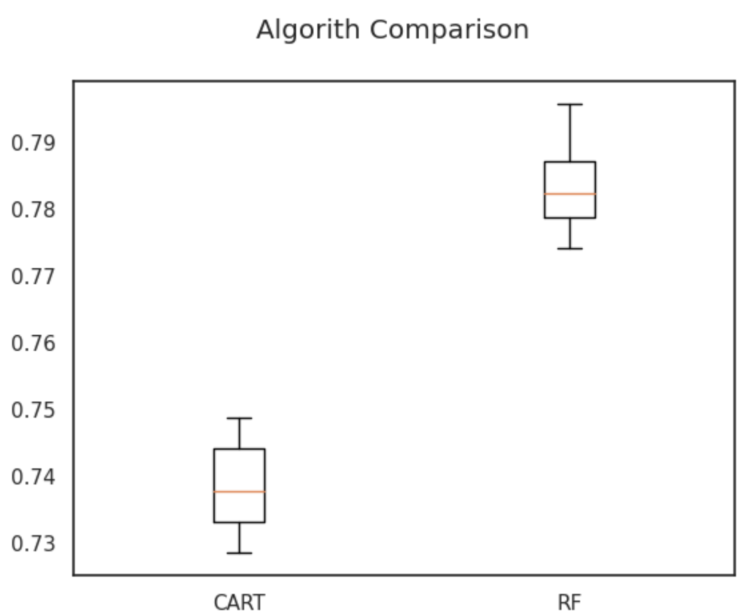
```
num_trees = 100
max_features = 3
```

```
models = []
models.append(('CART', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier(n_estimators=num_trees, max_features=max_features)))
```

```
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, shuffle=True, random_state=seed)
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
CART: 0.738409 (0.006618)
RF: 0.783469 (0.006845)
```

```
fig = plt.figure()
fig.suptitle('Algorith Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```




```
random_forest = RandomForestClassifier(n_estimators=250,max_features=5)
random_forest.fit(X_train, y_train)
predictions = random_forest.predict(X_test)
print("Accuracy: %s%%" % (100*accuracy_score(y_test, predictions)))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
Accuracy: 77.82244920134738%
[[6039  907]
 [1134 1123]]
              precision    recall  f1-score   support

           0       0.84      0.87      0.86      6946
           1       0.55      0.50      0.52      2257

    accuracy                           0.78      9203
   macro avg       0.70      0.68      0.69      9203
weighted avg       0.77      0.78      0.77      9203
```