PROJECT ON TEACHING ASSISTANT EVALUATION

Importing Required libraries

In [624]:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as plt
import pandas as pd
```

Getting the DataSet

In [625]:

```
#importing datasets
data_set= pd.read_csv('class.csv')
```

Shape of dataSet

In [626]:

```
data_set.shape
```

Out[626]:

```
(151, 6)
```

In [627]:

```
data_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 151 entries, 0 to 150
Data columns (total 6 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Native English Speaker  151 non-null    int64
 1   Course Instructor      151 non-null    int64
 2   Course                 151 non-null    int64
 3   Summer or regular      151 non-null    int64
 4   Class size             151 non-null    int64
 5   Class attribute        151 non-null    int64
dtypes: int64(6)
memory usage: 7.2 KB
```

Printing the Head of dataset

In [628]:

```
data_set.head()
```

Out[628]:

	Native English Speaker	Course Instructor	Course	Summer or regular	Class size	Class attribute
0	1	23	3	1	19	3
1	2	15	3	1	17	3
2	1	23	3	2	49	3
3	1	5	2	2	33	3
4	2	7	11	2	55	3

Extracting dependent and independent Variables

In [629]:

```
#Extracting Independent and dependent Variable
y = data_set['Class attribute']
x=data_set.drop('Class attribute', axis=1, inplace=True)
```

Shape of dependent variable

y.shape

Printing the dependent Variable

In [630]:

```
y
```

Out[630]:

```
0      3
1      3
2      3
3      3
4      3
      ..
146    1
147    1
148    1
149    1
150    1
Name: Class attribute, Length: 151, dtype: int64
```

In [631]:

```
#New Shape After Splitting
data_set.shape
```

Out[631]:

```
(151, 5)
```

In [632]:

```
data_set.head()
```

Out[632]:

	Native English Speaker	Course Instructor	Course	Summer or regular	Class size
0	1	23	3	1	19
1	2	15	3	1	17
2	1	23	3	2	49
3	1	5	2	2	33
4	2	7	11	2	55

In [633]:

```
x_train.head()
```

Out[633]:

	Native English Speaker	Course Instructor	Course	Summer or regular	Class size
75	2	4	16	2	21
88	1	23	3	2	38
124	2	14	15	2	36
95	2	1	8	2	18
79	1	13	3	1	17

In [634]:

```
x_test.head()
```

Out[634]:

	Native English Speaker	Course Instructor	Course	Summer or regular	Class size
119	2	15	1	2	19
43	2	7	11	2	55
44	2	23	3	1	20
31	2	18	5	2	19
84	1	22	3	2	45

In [635]:

```
y_train.head()
```

Out[635]:

```
75     1
88     3
124    3
95     2
79     3
Name: Class attribute, dtype: int64
```

In [636]:

```
y_test.head()
```

Out[636]:

```
119    1
43     3
44     3
31     1
84     3
Name: Class attribute, dtype: int64
```

Splitting the dataset into training and test set.

In [637]:

```
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data_set, y, test_size=0.25, shuffle=Tr
```

Random Forest Classifier

Fitting Decision Tree classifier to the training set

In [638]:

```
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 100, criterion='entropy',random_state=42)
```

In [639]:

```
classifier = RandomForestClassifier(n_estimators=100, max_features=0.7, bootstrap=True, max
```

Fitting

In [640]:

```
#fitting
classifier.fit(x_train, y_train)
```

Out[640]:

```
RandomForestClassifier(max_depth=10, max_features=0.7, min_samples_leaf=2,
                       random_state=42)
```

Prediction

In [641]:

```
# Prediction
y_pred = classifier.predict(x_test)
```

Metrices Confused matrix and Accuracy

In [642]:

```
#Metrics
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
accuracy = metrics.accuracy_score(y_test, y_pred)
#precision = metrics.precision_score(y_test, y_pred)
#recall = metrics.recall_score(y_test, y_pred)
```

Evaluating the Model Perform

In [648]:

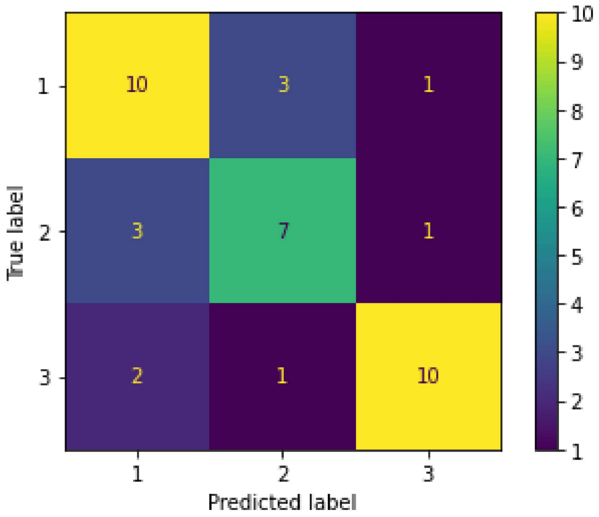
```
#Evaluating the model performance
from sklearn import metrics
cm = metrics.confusion_matrix(y_test, y_pred)
print("Confussion Matrix:\n",cm)
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy score:",accuracy)
```

```
Confussion Matrix:
 [[10  3  1]
 [ 3  7  1]
 [ 2  1 10]]
Accuracy score: 0.7105263157894737
```

Confusion Matrix

In [649]:

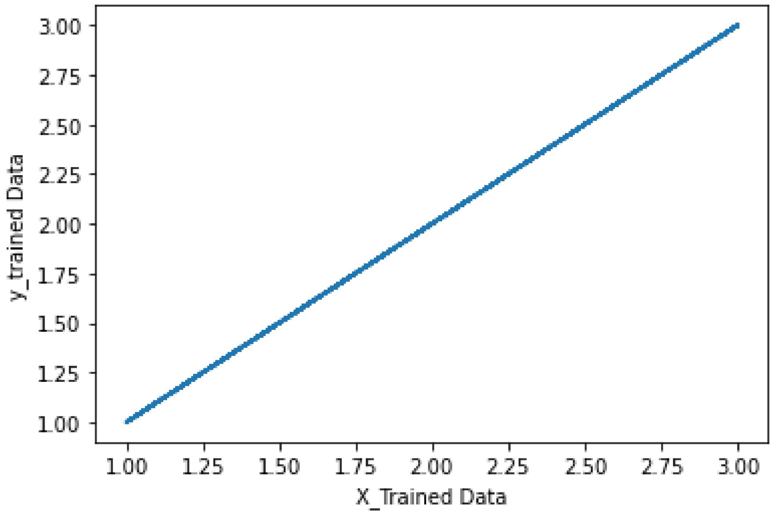
```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier,x_test,y_test)
plt.show()
```



Plotting X_trained data VS Y_trained data

In [650]:

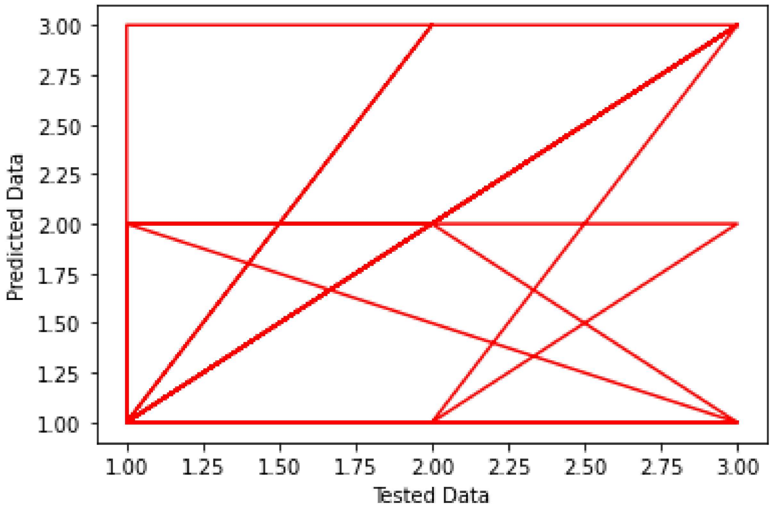
```
plt.plot(y_train,y_train)
plt.xlabel('X_Trained Data')
plt.ylabel('y_trained Data')
plt.show()
```



Plotting Y_tested data VS Y_predicted data

In [661]:

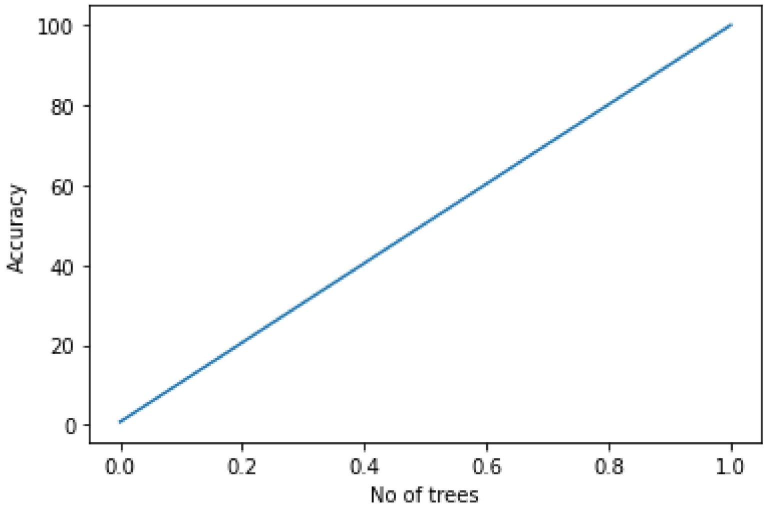
```
plt.plot(y_test,y_pred,color='red')
plt.xlabel('Tested Data')
plt.ylabel('Predicted Data')
plt.show()
```



Plotting Accuracy VS N_estimators

In [659]:

```
plt.plot((accuracy,100))
plt.xlabel('No of trees')
plt.ylabel('Accuracy')
plt.show()
```


Accuracy
No of trees