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

```
C:\Users\user\AppData\Roaming\Python\Python311\site-packages\pandas\
core\arrays\masked.py:61: UserWarning: Pandas requires version '1.3.6'
or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

```python
df =
pd.read_excel("C:\Users\user\Downloads\random_forest_dataset.xlsx")
df.head()
```

```
  Unnamed: 0  Unnamed: 1                  Unnamed: 2  \
0     Sl.No.         USN                        Name
1          1  1RV21MC001                 [illegible]
2          2  1RV21MC002                 [illegible]
3          3  1RV21MC003                 [illegible]
4          4  1RV21MC004                 [illegible]

                                           Unnamed: 3 Unnamed: 4
Unnamed: 5  \
0                                               Title         P1
C1
1                      Generative AI Prompt Pipeline          72
16.6
2  Android based Smart Vehicle Parking System usi...          84
16.8
3        Sentimental analysis for product ratings          84
16.8
4  Analysis and deployment of an efficient deep l...          88
17.6

  Unnamed: 6 Unnamed: 7 Unnamed: 8 Unnamed: 9 Unnamed: 10 Unnamed: 11
\
0         P2         C2         P3         C3          P4          T1

1         88         24         90       18.0          13           4

2         82       22.3         77       15.4          15           5

3         86       22.3         88       17.6          13           3

4         88         24         96       19.2          13           4


  Unnamed: 12 Unnamed: 13 Unnamed: 14
```

```
0     AST    Total    Grade
1    35.8     81.2        4
2    26.4       36        1
3    31.8     81.2        4
4    26.2     57.8        4
```

```
!pip install --upgrade openpyxl
```

```
df =
pd.read_excel("C:/Users/user/Downloads/random_forest_dataset.xlsx",
header=1)
df.head()
```

```
   Sl. No           USN                          Name  \
0       1  [illegible]                    [illegible]
1       2  [illegible]                    [illegible]
2       3  [illegible]                    [illegible]
3       4  [illegible]                    [illegible]
4       5  [illegible]                    [illegible]

                                               Title  P1    C1    P2
C2  \
0                        [illegible]   78  16.8  59.8
24.0
1  Android based Smart Vehicle Parking System and ...   86  16.9  82.8
22.8
2           Sentimental Analysis for product ratings   86  16.8  82.8
42.8
3  Analysis and Deployment of an efficient Deep L...   88  17.6  86.6
34.8
4  Development of Deep Learning Model for Varied ...   84  16.5  82.8
33.8

   P3    C3    R1   T1   AST  Total  Grade
0  88  14.8  13.8  4.8  80.8   88.3      4
1  77  18.4  15.8  9.8  26.4   66.8      4
2  88  17.8  15.8  1.8  31.8   81.3      4
3  58  18.3  13.8  4.8  30.2   57.8      4
4  77  16.9  16.8  8.8  88.4   88.8      4
```

```
df.isnull().sum()
```

```
Sl. No    0
USN       0
Name      0
Title     0
P1        0
C1        0
P2        0
C2        0
P3        0
```

```
C3       0
E1       0
T1       0
PST      0
Total    0
Grade    0
dtype: int64
```

```
# Handle outliers
df.dropna(inplace=True)
# there is this extra space after the column names
x = df.drop(['S.No ', 'USN ', 'Name ', 'Title ', 'Grade'], axis=1)
y = df.Grade
```

```
x.head()
```

	F1	C1	F2	C2	F3	C3	E1	T1	PST	Total
0	79	15.8	84.0	24.0	82	16.4	22.0	4.0	26.5	58.2
1	84	16.8	91.0	30.0	77	15.4	33.0	5.0	35.4	55.0
2	84	16.8	82.0	30.0	88	17.6	22.0	1.0	32.6	53.2
3	86	17.4	94.0	32.0	96	19.2	22.0	4.0	40.2	67.0
4	84	16.8	82.0	32.0	77	15.4	22.0	5.0	32.4	55.0

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

```
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['Grade'])
```

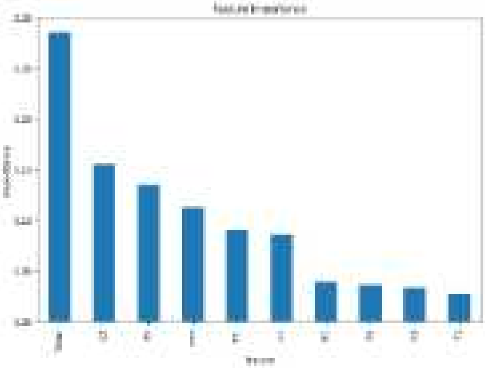
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
rf = RandomForestClassifier(random_state=[illegible])
rf.fit(x_train,y_train)
```

```
RandomForestClassifier(random_state=[illegible])
```

```
feature_importances = pd.Series(rf.feature_importances_,
index=x.columns).sort_values(ascending=False)
print(feature_importances)
plt.figure(figsize=(10, 6))
feature_importances.plot(kind='bar')
plt.xlabel('Feature')
plt.ylabel('Importance')
plt.title('Feature Importance')
plt.show()
```

```
Total    0.268488
C2       0.125264
F2       0.124661
PST      0.112657
F1       0.080618
C1       0.080622
E1       0.030038
```

```
M3        0.058238
C2        0.055946
T1        0.037876
dtype: float64
```



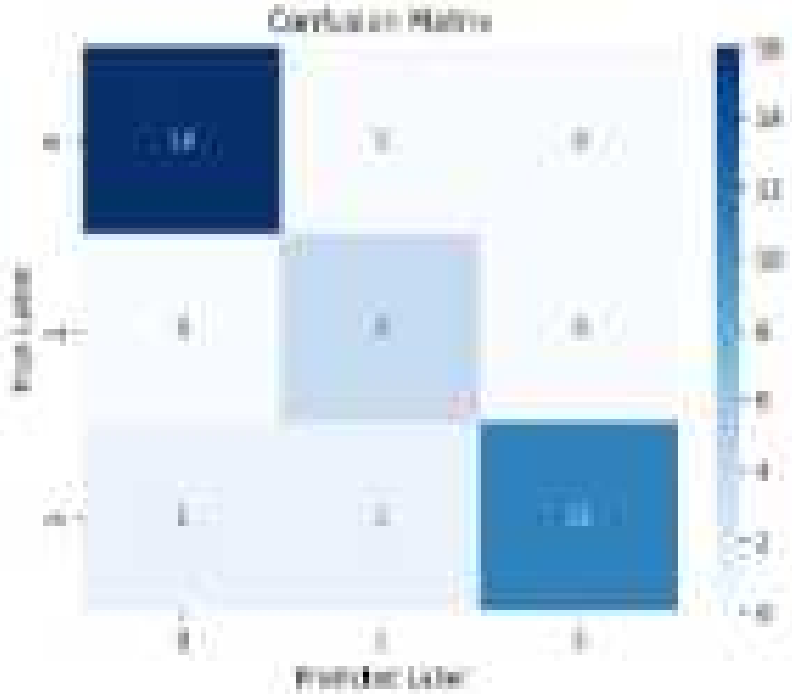
```python
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier

# Define the parameter grid correctly
param_grid = {
    'n_estimators': [100, 200, 300],  # Fixed key name
    'max_depth': [10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]  # Fixed 'min_samples_leaf' key
}

grid_search = GridSearchCV(estimator=rf, param_grid = param_grid,
cv=5, n_jobs=-1, verbose=2)

grid_search.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 81 candidates, totalling 405 fits
```

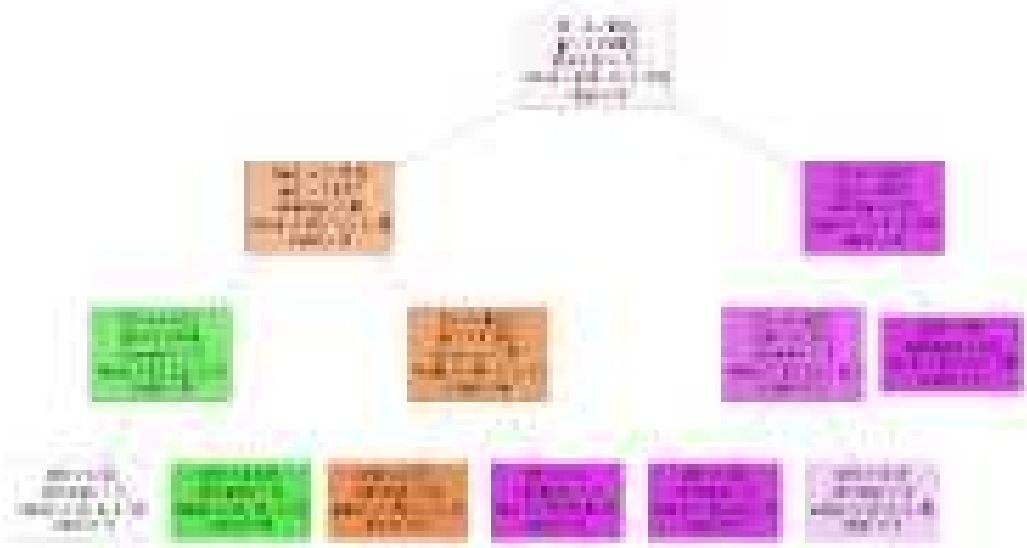
```
C:\Programdata\anaconda3\lib\site-packages\sklearn\model_selection\
_split.py:700: UserWarning: The least populated class in y has only 4
members, which is less than n_splits=5.
  warnings.warn(

GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=42),
n_jobs=-1,
             param_grid={'max_depth': [10, 20, 30],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [100, 200, 300]},
             verbose=2)

best_params = grid_search.best_params_
best_params

{'max_depth': 10,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 100}

best_rf = RandomForestClassifier(random_state=42,**best_params)
best_rf.fit(x_train,y_train)
y_pred = best_rf.predict(x_test)

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix,
    classification_report
)
# Print metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

# Print classification report
print("\nClassification Report:\n", classification_report(y_test,
y_pred))


Accuracy: 0.9384
Precision: 0.9473
Recall: 0.9384
```

```
F1 Score: 0.9394

Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        16
           1       0.88      1.00      0.88         8
           2       1.00      0.82      0.90        11

    accuracy                           0.94        35
   macro avg       0.91      0.94      0.92        35
weighted avg       0.95      0.94      0.94        35
```

```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=encoder.inverse_transform(),yticklabels=encoder.inverse_transform())
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

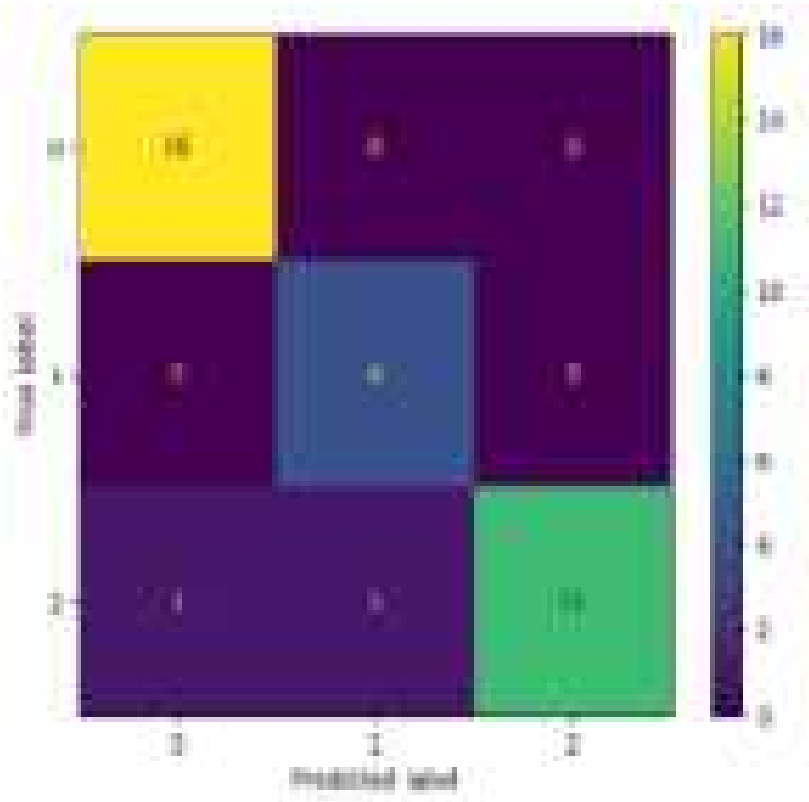


```
from sklearn.tree import plot_tree
plt.figure(figsize=(30,40))
plot_tree(best_rf.estimators_[5], feature_names =
x.columns,class_names=['a', 'b', 'c', 'd'],filled=True);
```



```
from sklearn.tree import plot_tree
plt.figure(figsize=(30,40))
plot_tree(best_rf.estimators_[7], feature_names =
x.columns,class_names=['a', 'b', 'c', 'd'],filled=True);
```

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import RandomizedSearchCV,
train_test_split
from scipy.stats import randint
param_dist = {
    'n_estimators': randint(50, 500),
    'max_depth': randint(1, 20),
    'min_samples_split': randint(2, 10),
    'min_samples_leaf': randint(1, 5)
}

# Create a random forest classifier
rf = RandomForestClassifier(random_state=0, n_jobs=-1)

# Use random search to find the best hyperparameters
rand_search = RandomizedSearchCV(
    rf, param_distributions=param_dist,
    n_iter=5, cv=5, scoring='accuracy',
    n_jobs=-1, random_state=42)
rand_search.fit(x, y)

# Create a variable for the best model
best_rf = rand_search.best_estimator_

# Print the best hyperparameters
print('Best hyperparameters:', rand_search.best_params_)
    # Improve predictions with the best model
y_pred = best_rf.predict(x_test)

# Create the confusion matrix
cm = confusion_matrix(y_true, y_pred)

ConfusionMatrixDisplay(confusion_matrix=cm).plot();

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\model_selection
_split.py:700: UserWarning: The least populated class in y has only 4
members, which is less than n_splits=5
  warnings.warn(

Best hyperparameters: {'max_depth': 3, 'min_samples_leaf': 3,
'min_samples_split': 4, 'n_estimators': 364}
```


True label
Predicted label