In [11]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sc
from sklearn.preprocessing import LabelEncoder

# Load dataset
df = pd.read_csv("D:/finalized_dataset.csv")  # adjust the path as needed
print("First few rows of the dataset:\n", df.head())

# Encode all object (categorical) columns
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
        print(f"Encoded column: {col}")

# Define features and target
X = df.iloc[:, :-1].values  # all columns except the last
Y = df.iloc[:, -1].values   # last column as target

# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_

# Train SVM
svm_clf = svm.SVC(kernel='rbf')
svm_clf.fit(X_train, Y_train)
svm_clf_pred = svm_clf.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(Y_test, svm_clf_pred))
print("Precision:", precision_score(Y_test, svm_clf_pred, average='weighted', ze
print("Recall:", recall_score(Y_test, svm_clf_pred, average='weighted'))
print("F1 Score:", f1_score(Y_test, svm_clf_pred, average='weighted'))
print("Confusion Matrix:\n", confusion_matrix(Y_test, svm_clf_pred))
print("Classification Report:\n", classification_report(Y_test, svm_clf_pred))
```

```
First few rows of the dataset:
    Database Fundamentals  Computer Architecture  \
0                      1                      6   
1                      2                      2   
2                      4                      4   
3                      1                      0   
4                      1                      1   

   Distributed Computing Systems  Cyber Security  Networking  \
0                              1               1           4   
1                              2               2           2   
2                              4               4           4   
3                              1               1           1   
4                              1               6           1   

   Software Development  Programming Skills  Project Management  \
0                     1                   1                   1   
1                     2                   2                   2   
2                     4                   4                   4   
3                     1                   1                   1   
4                     1                   1                   1   

   Computer Forensics Fundamentals  Technical Communication  AI ML  \
0                                1                        1      1   
1                                2                        2      2   
2                                4                        4      4   
3                                1                        1      1   
4                                1                        2      1   

   Software Engineering  Business Analysis  Communication skills  \
0                     1                  1                     1   
1                     2                  2                     2   
2                     4                  4                     5   
3                     1                  1                     6   
4                     1                  1                     1   

   Data Science  Troubleshooting skills  Graphics Designing  Role  
0             1                       1                   1     9  
1             2                       6                   2    10  
2             4                       4                   6     8  
3             1                       1                   1     4  
4             1                       1                   1     5  
Accuracy: 0.955
Precision: 0.9603399144333926
Recall: 0.955
F1 Score: 0.9559363316972531
Confusion Matrix:
 [[23  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 24  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 23  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 17  0  0  0  0  0  1  0  0  1  0  0]
 [ 0  0  0  0  0  1 20  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 27  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  1  1  0  1  0 20  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 21  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0]
```

```
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0 25  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0]
 [ 0  0  0  0  0  2  0  0  0  0  0  0  0  0  1 27  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.96      0.96        24
           1       1.00      0.92      0.96        26
           2       1.00      1.00      1.00        24
           3       0.95      1.00      0.98        21
           4       0.92      0.96      0.94        24
           5       0.74      0.89      0.81        19
           6       1.00      0.95      0.98        21
           7       0.96      0.96      0.96        28
           8       1.00      0.96      0.98        24
           9       1.00      0.83      0.91        24
          10       1.00      1.00      1.00        21
          11       0.96      1.00      0.98        25
          12       1.00      0.95      0.97        20
          13       1.00      0.96      0.98        26
          14       0.80      1.00      0.89        20
          15       0.96      0.90      0.93        30
          16       1.00      1.00      1.00        23

    accuracy                           0.95       400
   macro avg       0.96      0.96      0.95       400
weighted avg       0.96      0.95      0.96       400
```

In [15]:

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\subod\anaconda3\lib\site-pa
ckages (3.10.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\subod\anaconda3\lib\s
ite-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\subod\anaconda3\lib\site-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\subod\anaconda3\lib
\site-packages (from matplotlib) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\subod\anaconda3\lib
\site-packages (from matplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in c:\users\subod\anaconda3\lib\site-p
ackages (from matplotlib) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\users\subod\appdata\roaming
\python\python312\site-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in c:\users\subod\appdata\roaming\python
\python312\site-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\subod\anaconda3\lib\s
ite-packages (from matplotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\subod\appdata\roa
ming\python\python312\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\subod\appdata\roaming\python
\python312\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
```

In [29]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sc
from sklearn.preprocessing import LabelEncoder
```

```
# Load dataset
df = pd.read_csv("D:/finalized_dataset.csv")  # adjust the path as needed
print("First few rows of the dataset:\n", df.head())

# Encode all object (categorical) columns
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
        print(f"Encoded column: {col}")

# Define features and target
X = df.iloc[:, :-1].values  # all columns except the last
Y = df.iloc[:, -1].values   # last column as target

# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_

# Train SVM
svm_clf = svm.SVC(kernel='rbf')
svm_clf.fit(X_train, Y_train)
svm_clf_pred = svm_clf.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(Y_test, svm_clf_pred))
print("Precision:", precision_score(Y_test, svm_clf_pred, average='weighted', ze
print("Recall:", recall_score(Y_test, svm_clf_pred, average='weighted'))
print("F1 Score:", f1_score(Y_test, svm_clf_pred, average='weighted'))
print("Confusion Matrix:\n", confusion_matrix(Y_test, svm_clf_pred))
print("Classification Report:\n", classification_report(Y_test, svm_clf_pred))
```

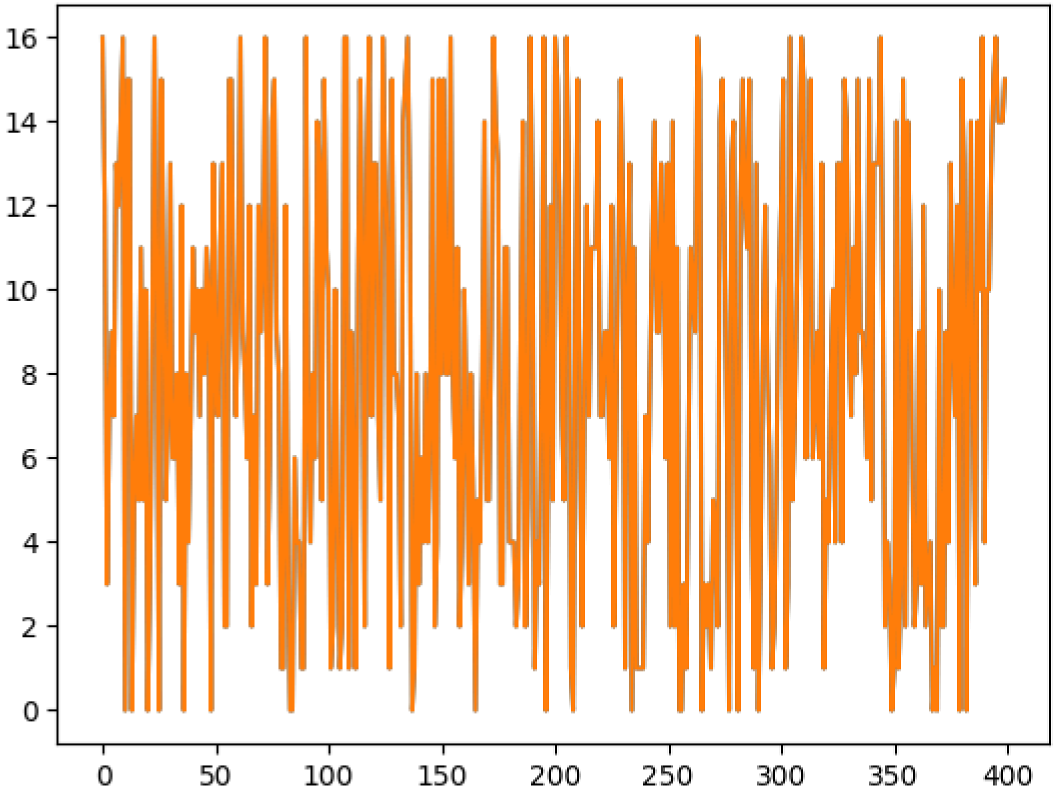
```
First few rows of the dataset:
    Database Fundamentals  Computer Architecture  \
0                      1                      6   
1                      2                      2   
2                      4                      4   
3                      1                      0   
4                      1                      1   

   Distributed Computing Systems  Cyber Security  Networking  \
0                              1               1           4   
1                              2               2           2   
2                              4               4           4   
3                              1               1           1   
4                              1               6           1   

   Software Development  Programming Skills  Project Management  \
0                     1                   1                   1   
1                     2                   2                   2   
2                     4                   4                   4   
3                     1                   1                   1   
4                     1                   1                   1   

   Computer Forensics Fundamentals  Technical Communication  AI ML  \
0                                1                        1      1   
1                                2                        2      2   
2                                4                        4      4   
3                                1                        1      1   
4                                1                        2      1   

   Software Engineering  Business Analysis  Communication skills  \
0                     1                  1                     1   
1                     2                  2                     2   
2                     4                  4                     5   
3                     1                  1                     6   
4                     1                  1                     1   

   Data Science  Troubleshooting skills  Graphics Designing  Role  
0             1                       1                   1     9  
1             2                       6                   2    10  
2             4                       4                   6     8  
3             1                       1                   1     4  
4             1                       1                   1     5  
Accuracy: 0.955
Precision: 0.9603399144333926
Recall: 0.955
F1 Score: 0.9559363316972531
Confusion Matrix:
 [[23  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 24  0  0  0  1  0  0  0  0  0  0  0  0  1  0  0]
 [ 0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 23  1  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 17  0  0  0  0  0  1  0  0  1  0  0]
 [ 0  0  0  0  0  1 20  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 27  0  0  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0 23  0  0  0  0  0  0  1  0]
 [ 0  0  0  0  1  1  0  1  0 20  0  0  0  0  1  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 21  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 1  0  0  0  0  0  0  0  0  0  0  0 19  0  0  0  0]
```

```
 [ 0  0  0  0  1  0  0  0  0  0  0  0  0 25  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0]
 [ 0  0  0  0  0  2  0  0  0  0  0  0  0  0  1 27  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.96      0.96        24
           1       1.00      0.92      0.96        26
           2       1.00      1.00      1.00        24
           3       0.95      1.00      0.98        21
           4       0.92      0.96      0.94        24
           5       0.74      0.89      0.81        19
           6       1.00      0.95      0.98        21
           7       0.96      0.96      0.96        28
           8       1.00      0.96      0.98        24
           9       1.00      0.83      0.91        24
          10       1.00      1.00      1.00        21
          11       0.96      1.00      0.98        25
          12       1.00      0.95      0.97        20
          13       1.00      0.96      0.98        26
          14       0.80      1.00      0.89        20
          15       0.96      0.90      0.93        30
          16       1.00      1.00      1.00        23

    accuracy                           0.95       400
   macro avg       0.96      0.96      0.95       400
weighted avg       0.96      0.95      0.96       400
```

In [31]:

```
#Decision tree
#dt_clf=DecisionTreeClassifier(random_state=0)
gnb_clf=DecisionTreeClassifier(random_state=0)
gnb_clf.fit(X_train,Y_train)
gnb_clf_pred=gnb_clf.predict(X_test)
print("accuracy for DT",accuracy_score(Y_test,gnb_clf_pred) )
print("precision",precision_score(Y_test,gnb_clf_pred,average='weighted') )
print("recall",recall_score(Y_test,gnb_clf_pred,average='weighted') )
print("f1 score",f1_score(Y_test,gnb_clf_pred,average='weighted') )
plt.plot(Y_test)
plt.plot(gnb_clf_pred)
plt.show()
```

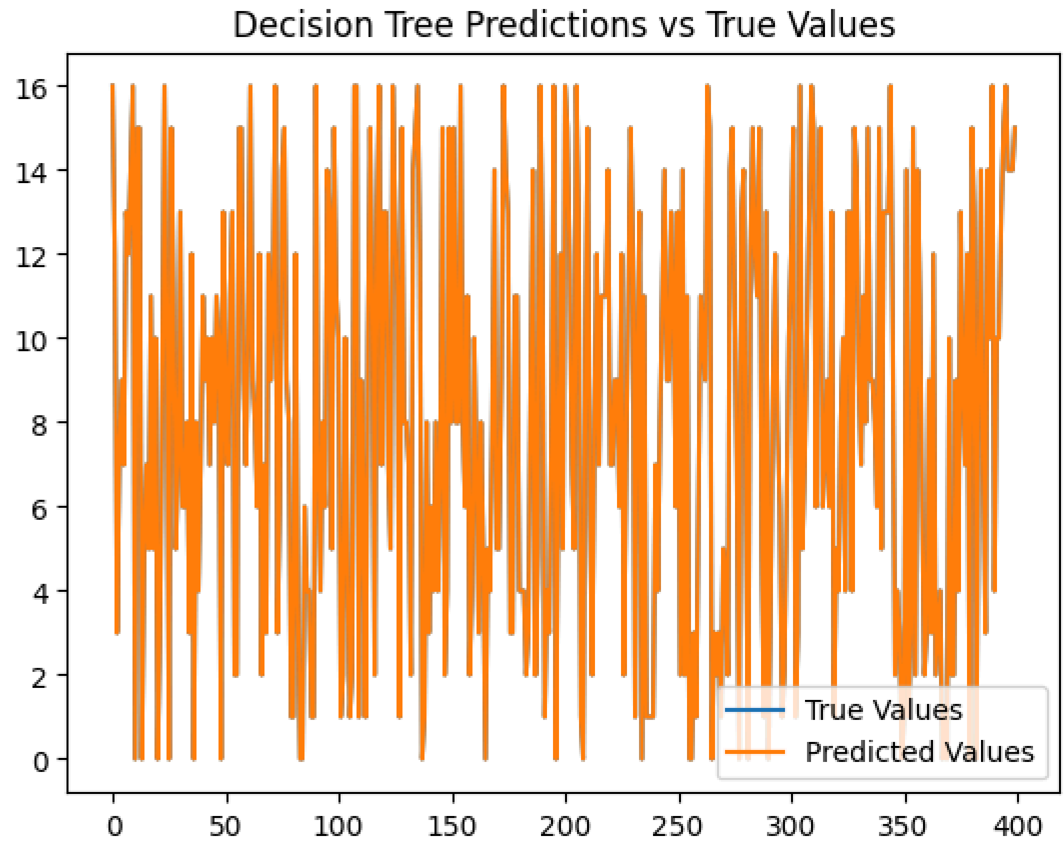
```
accuracy for DT 1.0
precision 1.0
recall 1.0
f1 score 1.0
```



In [37]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("D:/finalized_dataset.csv")  # Update path if needed

# Encode categorical columns
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
        print(f"Encoded column: {col}")

# Split features and target
X = df.iloc[:, :-1].values  # all columns except the last
Y = df.iloc[:, -1].values   # last column as target

# Split into training and test sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_

# Decision Tree Classifier
dt_clf = DecisionTreeClassifier(random_state=0)
dt_clf.fit(X_train, Y_train)
dt_clf_pred = dt_clf.predict(X_test)

# Evaluation
print("Accuracy for Decision Tree:", accuracy_score(Y_test, dt_clf_pred))
print("Precision:", precision_score(Y_test, dt_clf_pred, average='weighted', zer
print("Recall:", recall_score(Y_test, dt_clf_pred, average='weighted'))
print("F1 Score:", f1_score(Y_test, dt_clf_pred, average='weighted'))
```

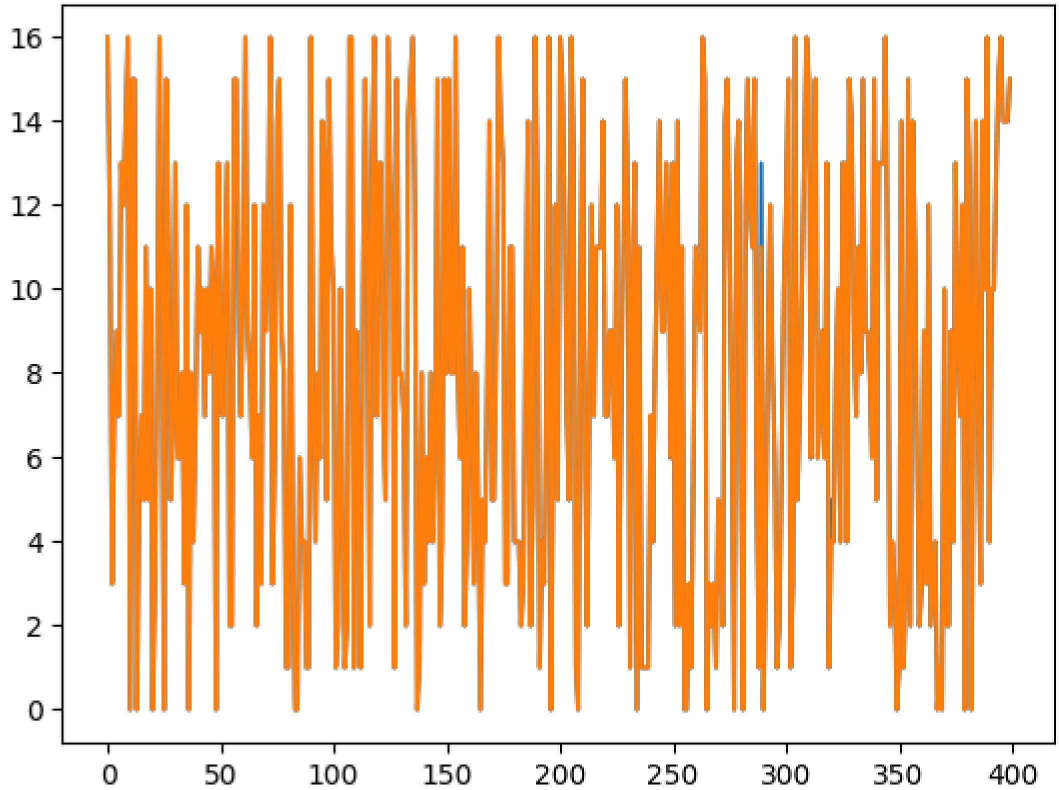
```
print("Confusion Matrix:\n", confusion_matrix(Y_test, dt_clf_pred))
print("Classification Report:\n", classification_report(Y_test, dt_clf_pred))

# Plotting
plt.plot(Y_test, label='True Values')
plt.plot(dt_clf_pred, label='Predicted Values')
plt.title("Decision Tree Predictions vs True Values")
plt.legend()
plt.show()
```

```
Accuracy for Decision Tree: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0
Confusion Matrix:
 [[24  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0 24  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0 19  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0 28  0  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0 24  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0 24  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0 21  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0 20  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 30  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 23]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        24
           1       1.00      1.00      1.00        26
           2       1.00      1.00      1.00        24
           3       1.00      1.00      1.00        21
           4       1.00      1.00      1.00        24
           5       1.00      1.00      1.00        19
           6       1.00      1.00      1.00        21
           7       1.00      1.00      1.00        28
           8       1.00      1.00      1.00        24
           9       1.00      1.00      1.00        24
          10       1.00      1.00      1.00        21
          11       1.00      1.00      1.00        25
          12       1.00      1.00      1.00        20
          13       1.00      1.00      1.00        26
          14       1.00      1.00      1.00        20
          15       1.00      1.00      1.00        30
          16       1.00      1.00      1.00        23

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```



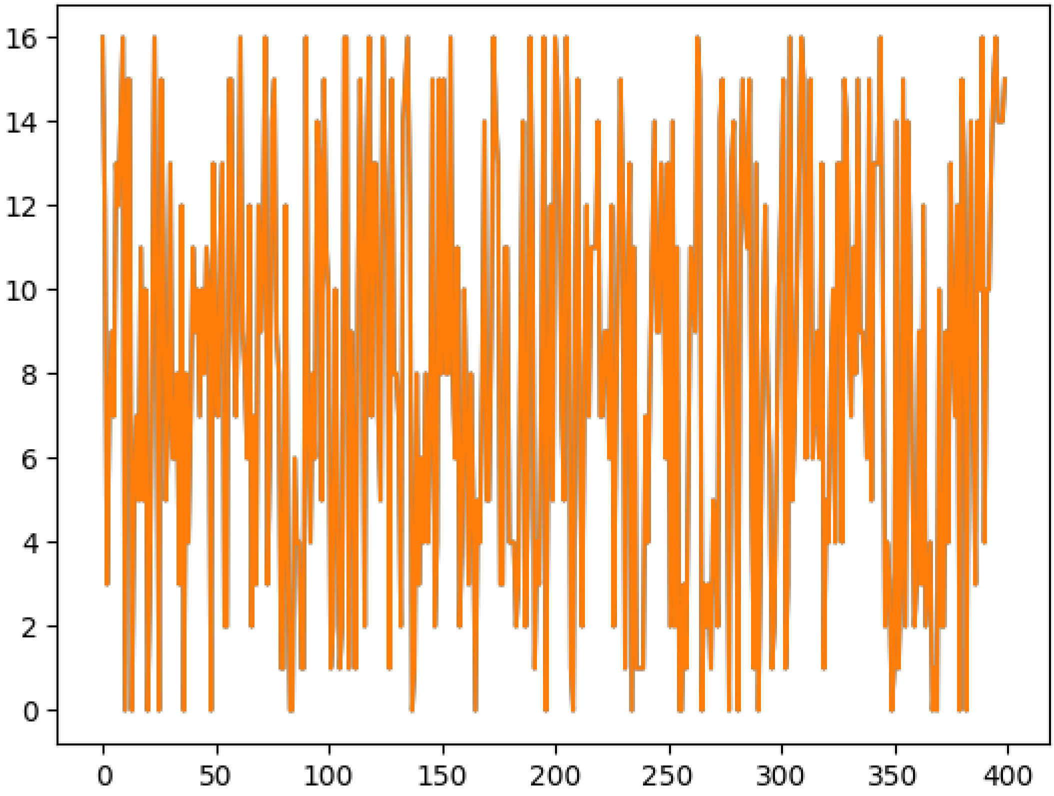
In [39]:
```
#logistic regression
from sklearn.linear_model import LogisticRegression

gnb_clf=LogisticRegression(solver='liblinear')
gnb_clf.fit(X_train,Y_train)
gnb_clf_pred=gnb_clf.predict(X_test)
print("accuracy for LR",accuracy_score(Y_test,gnb_clf_pred) )
print("precision",precision_score(Y_test,gnb_clf_pred,average='weighted') )
print("recall",recall_score(Y_test,gnb_clf_pred,average='weighted') )
print("f1 score",f1_score(Y_test,gnb_clf_pred,average='weighted') )
plt.plot(Y_test)
plt.plot(gnb_clf_pred)
plt.show()
```

```
accuracy for LR 0.9925
precision 0.9929703703703703
recall 0.9925
f1 score 0.9925367747178493
```



In [41]:
```
gnb_clf=GaussianNB()
gnb_clf.fit(X_train,Y_train)
gnb_clf_pred=gnb_clf.predict(X_test)
print("accuracy for NB",accuracy_score(Y_test,gnb_clf_pred) )
print("precision",precision_score(Y_test,gnb_clf_pred,average='weighted') )
print("recall",recall_score(Y_test,gnb_clf_pred,average='weighted') )
print("f1 score",f1_score(Y_test,gnb_clf_pred,average='weighted') )
plt.plot(Y_test)
plt.plot(gnb_clf_pred)
plt.show()
```

```
accuracy for NB 1.0
precision 1.0
recall 1.0
f1 score 1.0
```

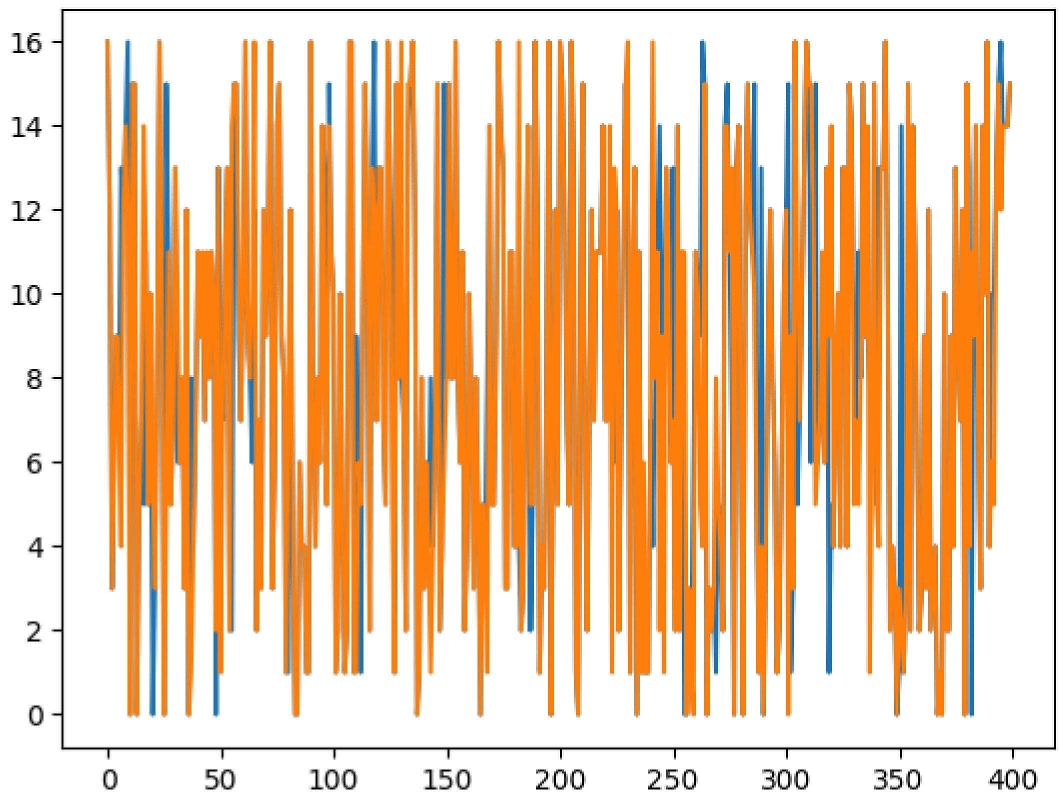


In [43]:
```
#MLP
clf=MLPClassifier(hidden_layer_sizes=(6,5),random_state=5,verbose=True,learning_
clf.fit(X_train,Y_train)
clf_pred=clf.predict(X_test)
print("accuracy for MLP",accuracy_score(Y_test,clf_pred))
plt.plot(Y_test)
plt.plot(clf_pred)
plt.show()
```

```
Iteration 1, loss = 3.20401823
Iteration 2, loss = 2.81121002
Iteration 3, loss = 2.71988719
Iteration 4, loss = 2.58823699
Iteration 5, loss = 2.48774612
Iteration 6, loss = 2.36580104
Iteration 7, loss = 2.23786767
Iteration 8, loss = 2.10747775
Iteration 9, loss = 1.99572370
Iteration 10, loss = 1.88580768
Iteration 11, loss = 1.76644117
Iteration 12, loss = 1.68678728
Iteration 13, loss = 1.61397890
Iteration 14, loss = 1.53963861
Iteration 15, loss = 1.47085081
Iteration 16, loss = 1.44660482
Iteration 17, loss = 1.44099006
Iteration 18, loss = 1.40292811
Iteration 19, loss = 1.34125128
Iteration 20, loss = 1.31132894
Iteration 21, loss = 1.29839071
Iteration 22, loss = 1.26030000
Iteration 23, loss = 1.26119033
Iteration 24, loss = 1.24165638
Iteration 25, loss = 1.20685466
Iteration 26, loss = 1.16713793
Iteration 27, loss = 1.16299244
Iteration 28, loss = 1.12242036
Iteration 29, loss = 1.10039827
Iteration 30, loss = 1.08722594
Iteration 31, loss = 1.06493209
Iteration 32, loss = 1.05482697
Iteration 33, loss = 1.02701479
Iteration 34, loss = 1.00900967
Iteration 35, loss = 0.99206918
Iteration 36, loss = 0.97603844
Iteration 37, loss = 0.96237396
Iteration 38, loss = 0.95010471
Iteration 39, loss = 0.92719574
Iteration 40, loss = 0.92524838
Iteration 41, loss = 0.90863036
Iteration 42, loss = 0.88906292
Iteration 43, loss = 0.88338384
Iteration 44, loss = 0.86861295
Iteration 45, loss = 0.85677168
Iteration 46, loss = 0.84103052
Iteration 47, loss = 0.82776423
Iteration 48, loss = 0.81494309
Iteration 49, loss = 0.80819571
Iteration 50, loss = 0.79283865
Iteration 51, loss = 0.79455249
Iteration 52, loss = 0.77749600
Iteration 53, loss = 0.76339898
Iteration 54, loss = 0.75823840
Iteration 55, loss = 0.75021541
Iteration 56, loss = 0.75004674
Iteration 57, loss = 0.74743565
Iteration 58, loss = 0.74786132
Iteration 59, loss = 0.74977742
Iteration 60, loss = 0.75455112
```

```
Iteration 61, loss = 0.76149201
Iteration 62, loss = 0.72984151
Iteration 63, loss = 0.70746727
Iteration 64, loss = 0.70857437
Iteration 65, loss = 0.69898258
Iteration 66, loss = 0.67391068
Iteration 67, loss = 0.67724710
Iteration 68, loss = 0.67168212
Iteration 69, loss = 0.67134770
Iteration 70, loss = 0.65935367
Iteration 71, loss = 0.65804287
Iteration 72, loss = 0.66384322
Iteration 73, loss = 0.66824438
Iteration 74, loss = 0.66128438
Iteration 75, loss = 0.63567519
Iteration 76, loss = 0.63552655
Iteration 77, loss = 0.63621900
Iteration 78, loss = 0.63380830
Iteration 79, loss = 0.64792562
Iteration 80, loss = 0.63577630
Iteration 81, loss = 0.62668023
Iteration 82, loss = 0.61999423
Iteration 83, loss = 0.61594503
Iteration 84, loss = 0.64962018
Iteration 85, loss = 0.69415780
Iteration 86, loss = 0.64747202
Iteration 87, loss = 0.63433838
Iteration 88, loss = 0.60064368
Iteration 89, loss = 0.58216491
Iteration 90, loss = 0.58069227
Iteration 91, loss = 0.57130478
Iteration 92, loss = 0.57185385
Iteration 93, loss = 0.56553612
Iteration 94, loss = 0.56590965
Iteration 95, loss = 0.56543984
Iteration 96, loss = 0.56655611
Iteration 97, loss = 0.57454781
Iteration 98, loss = 0.57918801
Iteration 99, loss = 0.58683519
Iteration 100, loss = 0.58186827
Iteration 101, loss = 0.56109616
Iteration 102, loss = 0.56431328
Iteration 103, loss = 0.56110141
Iteration 104, loss = 0.57832536
Iteration 105, loss = 0.56961952
Iteration 106, loss = 0.55600170
Iteration 107, loss = 0.57520027
Iteration 108, loss = 0.57760339
Iteration 109, loss = 0.57385702
Iteration 110, loss = 0.56304203
Iteration 111, loss = 0.54927167
Iteration 112, loss = 0.53816345
Iteration 113, loss = 0.55086742
Iteration 114, loss = 0.55028093
Iteration 115, loss = 0.55223366
Iteration 116, loss = 0.55106650
Iteration 117, loss = 0.52935474
Iteration 118, loss = 0.51492060
Iteration 119, loss = 0.51956048
Iteration 120, loss = 0.51723938
```

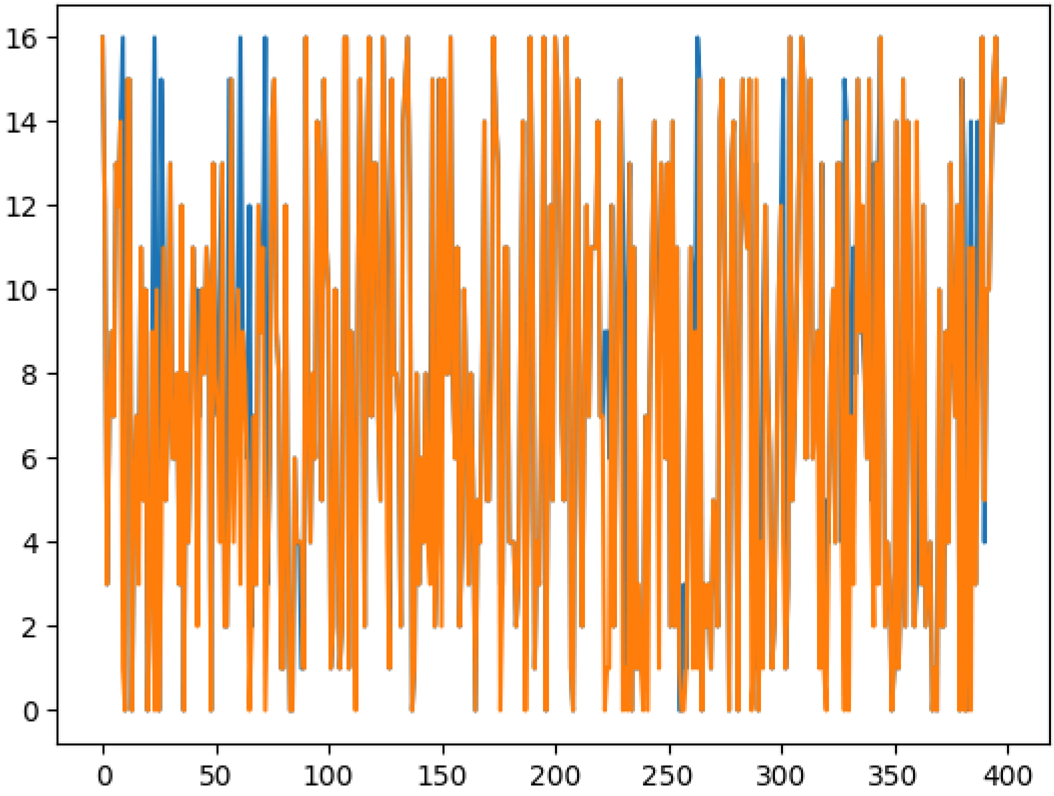
```
Iteration 121, loss = 0.51514800
Iteration 122, loss = 0.53092981
Iteration 123, loss = 0.52742511
Iteration 124, loss = 0.52075992
Iteration 125, loss = 0.51673652
Iteration 126, loss = 0.51850353
Iteration 127, loss = 0.50931954
Iteration 128, loss = 0.51195739
Iteration 129, loss = 0.50977405
Iteration 130, loss = 0.51947502
Iteration 131, loss = 0.53256806
Iteration 132, loss = 0.50989600
Iteration 133, loss = 0.50799590
Iteration 134, loss = 0.51628196
Iteration 135, loss = 0.50770879
Iteration 136, loss = 0.50860405
Iteration 137, loss = 0.49604500
Iteration 138, loss = 0.49582465
Iteration 139, loss = 0.50212263
Iteration 140, loss = 0.49096992
Iteration 141, loss = 0.49768031
Iteration 142, loss = 0.51530417
Iteration 143, loss = 0.49467412
Iteration 144, loss = 0.49598090
Iteration 145, loss = 0.49426572
Iteration 146, loss = 0.48322285
Iteration 147, loss = 0.49364454
Iteration 148, loss = 0.50794922
Iteration 149, loss = 0.48432083
Iteration 150, loss = 0.48405384
Iteration 151, loss = 0.49614968
Iteration 152, loss = 0.49546256
Iteration 153, loss = 0.51921162
Iteration 154, loss = 0.51790864
Iteration 155, loss = 0.50497942
Iteration 156, loss = 0.50295495
Iteration 157, loss = 0.50224823
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. S
topping.
accuracy for MLP 0.83
```



In [45]:

```
#KNN
from sklearn.neighbors import KNeighborsClassifier

gnb_clf=KNeighborsClassifier()
gnb_clf.fit(X_train,Y_train)
gnb_clf_pred=gnb_clf.predict(X_test)
print("accuracy for KNN",accuracy_score(Y_test,gnb_clf_pred) )
print("precision",precision_score(Y_test,gnb_clf_pred,average='weighted') )
print("recall",recall_score(Y_test,gnb_clf_pred,average='weighted') )
print("f1 score",f1_score(Y_test,gnb_clf_pred,average='weighted') )
plt.plot(Y_test)
plt.plot(gnb_clf_pred)
plt.show()
```

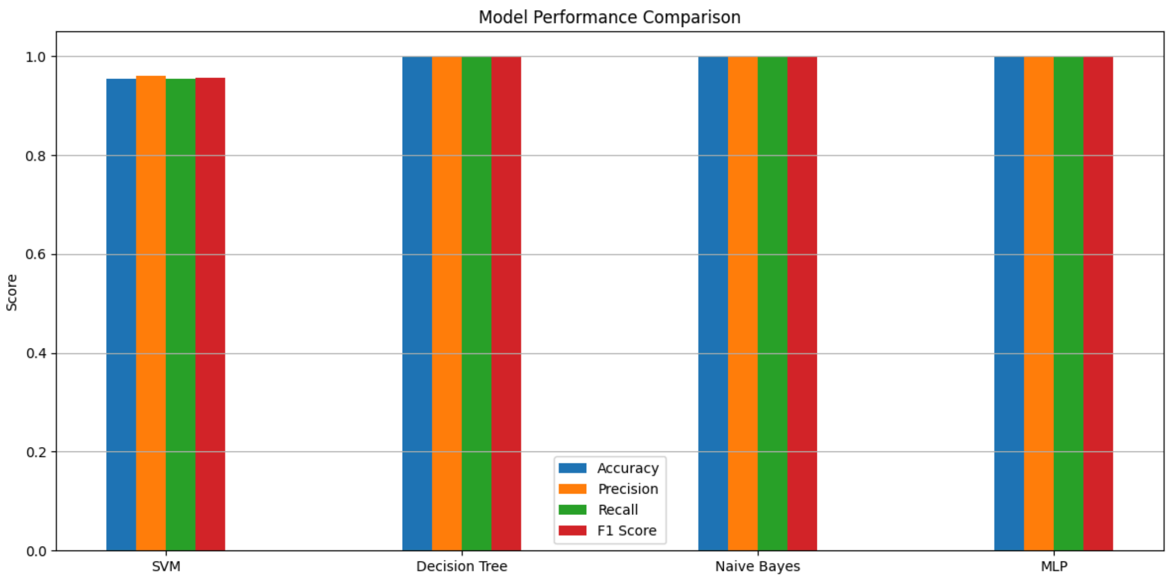
```
accuracy for KNN 0.8725
precision 0.8982303022795848
recall 0.8725
f1 score 0.8767909496467446
```



In [57]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc

# Load and preprocess dataset
df = pd.read_csv("D:/finalized_dataset.csv")  # Update this path accordingly

# Encode categorical columns
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_

# Models to evaluate
models = {
    'SVM': SVC(kernel='rbf'),
    'Decision Tree': DecisionTreeClassifier(random_state=0),
    'Naive Bayes': GaussianNB(),
    'MLP': MLPClassifier(max_iter=1000, random_state=1)
}

# Store metrics
metrics = {
    'Model': [],
```

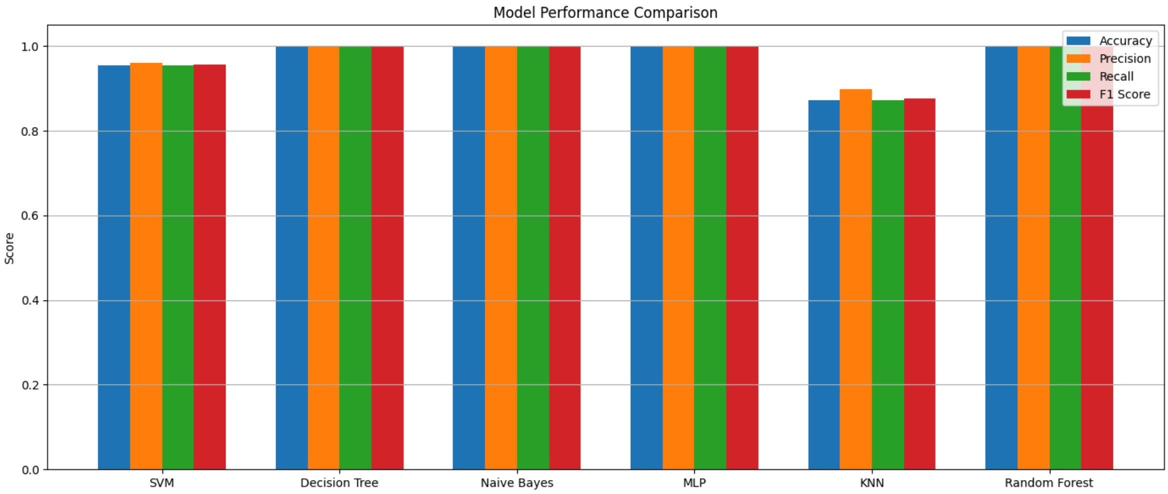
```
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1 Score': []
}

# Evaluate each model
for name, model in models.items():
    model.fit(X_train, Y_train)
    preds = model.predict(X_test)

    metrics['Model'].append(name)
    metrics['Accuracy'].append(accuracy_score(Y_test, preds))
    metrics['Precision'].append(precision_score(Y_test, preds, average='weighted
    metrics['Recall'].append(recall_score(Y_test, preds, average='weighted'))
    metrics['F1 Score'].append(f1_score(Y_test, preds, average='weighted'))

# Convert to DataFrame for easier plotting
metrics_df = pd.DataFrame(metrics)

# Plot as grouped bar chart
x = range(len(metrics_df['Model']))
width = 0.1

plt.figure(figsize=(12, 6))
plt.bar([p - 1.5*width for p in x], metrics_df['Accuracy'], width=width, label='
plt.bar([p - 0.5*width for p in x], metrics_df['Precision'], width=width, label=
plt.bar([p + 0.5*width for p in x], metrics_df['Recall'], width=width, label='Re
plt.bar([p + 1.5*width for p in x], metrics_df['F1 Score'], width=width, label='

plt.xticks(x, metrics_df['Model'])
plt.ylabel("Score")
plt.title("Model Performance Comparison")
plt.legend()
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [59]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc

# Load and preprocess dataset
df = pd.read_csv("D:/finalized_dataset.csv")  # Update path if needed

# Encode categorical columns
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

# Features and target
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values

# Train/test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_

# Models dictionary
models = {
    'SVM': SVC(kernel='rbf'),
    'Decision Tree': DecisionTreeClassifier(random_state=0),
    'Naive Bayes': GaussianNB(),
    'MLP': MLPClassifier(max_iter=1000, random_state=1),
    'KNN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(random_state=1)
}

# Initialize metric storage
metrics = {
    'Model': [],
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1 Score': []
}

# Evaluate each model
for name, model in models.items():
    model.fit(X_train, Y_train)
    preds = model.predict(X_test)

    metrics['Model'].append(name)
    metrics['Accuracy'].append(accuracy_score(Y_test, preds))
    metrics['Precision'].append(precision_score(Y_test, preds, average='weighted
    metrics['Recall'].append(recall_score(Y_test, preds, average='weighted'))
    metrics['F1 Score'].append(f1_score(Y_test, preds, average='weighted'))

# Convert to DataFrame
metrics_df = pd.DataFrame(metrics)

# Plotting grouped bar chart
x = range(len(metrics_df['Model']))
width = 0.18
```

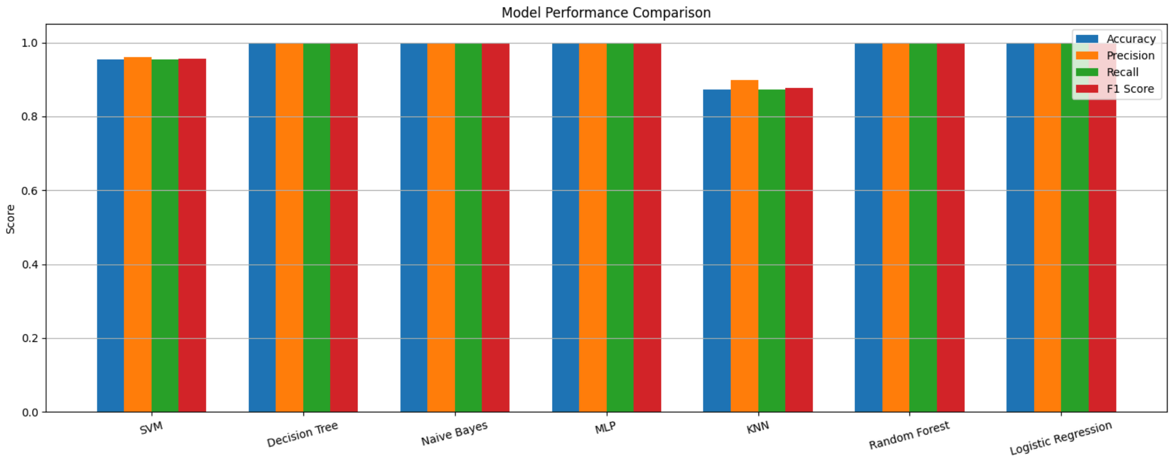
```
plt.figure(figsize=(14, 6))
plt.bar([p - 1.5*width for p in x], metrics_df['Accuracy'], width=width, label='
plt.bar([p - 0.5*width for p in x], metrics_df['Precision'], width=width, label=
plt.bar([p + 0.5*width for p in x], metrics_df['Recall'], width=width, label='Re
plt.bar([p + 1.5*width for p in x], metrics_df['F1 Score'], width=width, label='

plt.xticks(x, metrics_df['Model'])
plt.ylabel("Score")
plt.title("Model Performance Comparison")
plt.legend()
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [61]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc

# Load dataset
df = pd.read_csv("D:/finalized_dataset.csv")  # Adjust path if needed

# Encode categorical features
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

# Split data into features and target
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values

# Train-test split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_
```

```
# Define models
models = {
    'SVM': SVC(kernel='rbf'),
    'Decision Tree': DecisionTreeClassifier(random_state=0),
    'Naive Bayes': GaussianNB(),
    'MLP': MLPClassifier(max_iter=1000, random_state=1),
    'KNN': KNeighborsClassifier(),
    'Random Forest': RandomForestClassifier(random_state=1),
    'Logistic Regression': LogisticRegression(max_iter=1000)
}

# Store metrics
metrics = {
    'Model': [],
    'Accuracy': [],
    'Precision': [],
    'Recall': [],
    'F1 Score': []
}

# Train and evaluate models
for name, model in models.items():
    model.fit(X_train, Y_train)
    preds = model.predict(X_test)

    metrics['Model'].append(name)
    metrics['Accuracy'].append(accuracy_score(Y_test, preds))
    metrics['Precision'].append(precision_score(Y_test, preds, average='weighted
    metrics['Recall'].append(recall_score(Y_test, preds, average='weighted'))
    metrics['F1 Score'].append(f1_score(Y_test, preds, average='weighted'))

# Create DataFrame
metrics_df = pd.DataFrame(metrics)

# Plot Histogram
x = range(len(metrics_df['Model']))
width = 0.18

plt.figure(figsize=(15, 6))
plt.bar([p - 1.5 * width for p in x], metrics_df['Accuracy'], width=width, label
plt.bar([p - 0.5 * width for p in x], metrics_df['Precision'], width=width, labe
plt.bar([p + 0.5 * width for p in x], metrics_df['Recall'], width=width, label='
plt.bar([p + 1.5 * width for p in x], metrics_df['F1 Score'], width=width, label

plt.xticks(x, metrics_df['Model'], rotation=15)
plt.ylabel("Score")
plt.title("Model Performance Comparison")
plt.legend()
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



In [63]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc

# Load dataset
df = pd.read_csv("D:/finalized_dataset.csv")

# Encode categorical columns
for col in df.columns:
    if df[col].dtype == 'object':
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

# Features and target
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values

# Split data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_

# Random Forest model
rf = RandomForestClassifier(random_state=1)
rf.fit(X_train, Y_train)

# Predict
pred = rf.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(Y_test, pred))
print("Precision:", precision_score(Y_test, pred, average='weighted', zero_divis
print("Recall:", recall_score(Y_test, pred, average='weighted'))
print("F1 Score:", f1_score(Y_test, pred, average='weighted'))
print("\nClassification Report:\n", classification_report(Y_test, pred))
```

```
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
F1 Score: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        24
           1       1.00      1.00      1.00        26
           2       1.00      1.00      1.00        24
           3       1.00      1.00      1.00        21
           4       1.00      1.00      1.00        24
           5       1.00      1.00      1.00        19
           6       1.00      1.00      1.00        21
           7       1.00      1.00      1.00        28
           8       1.00      1.00      1.00        24
           9       1.00      1.00      1.00        24
          10       1.00      1.00      1.00        21
          11       1.00      1.00      1.00        25
          12       1.00      1.00      1.00        20
          13       1.00      1.00      1.00        26
          14       1.00      1.00      1.00        20
          15       1.00      1.00      1.00        30
          16       1.00      1.00      1.00        23

    accuracy                           1.00       400
   macro avg       1.00      1.00      1.00       400
weighted avg       1.00      1.00      1.00       400
```

In [19]:

```
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\subod\appdata\roaming\python
\python312\site-packages (7.16.6)
Requirement already satisfied: beautifulsoup4 in c:\users\subod\appdata\roaming\p
ython\python312\site-packages (from nbconvert) (4.13.3)
Requirement already satisfied: bleach!=5.0.0 in c:\users\subod\appdata\roaming\py
thon\python312\site-packages (from bleach[css]!=5.0.0->nbconvert) (6.2.0)
Requirement already satisfied: defusedxml in c:\users\subod\appdata\roaming\pytho
n\python312\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\users\subod\appdata\roaming\pyth
on\python312\site-packages (from nbconvert) (3.1.5)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\subod\appdata\roamin
g\python\python312\site-packages (from nbconvert) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in c:\users\subod\appdata\roam
ing\python\python312\site-packages (from nbconvert) (0.3.0)
Requirement already satisfied: markupsafe>=2.0 in c:\users\subod\appdata\roaming
\python\python312\site-packages (from nbconvert) (3.0.2)
Requirement already satisfied: mistune<4,>=2.0.3 in c:\users\subod\appdata\roamin
g\python\python312\site-packages (from nbconvert) (3.1.2)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\subod\appdata\roaming
\python\python312\site-packages (from nbconvert) (0.10.2)
Requirement already satisfied: nbformat>=5.7 in c:\users\subod\appdata\roaming\py
thon\python312\site-packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in c:\users\subod\appdata\roaming\python
\python312\site-packages (from nbconvert) (24.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\subod\appdata\roa
ming\python\python312\site-packages (from nbconvert) (1.5.1)
Requirement already satisfied: pygments>=2.4.1 in c:\users\subod\appdata\roaming
\python\python312\site-packages (from nbconvert) (2.19.1)
Requirement already satisfied: traitlets>=5.1 in c:\users\subod\appdata\roaming\p
ython\python312\site-packages (from nbconvert) (5.14.3)
Requirement already satisfied: webencodings in c:\users\subod\appdata\roaming\pyt
hon\python312\site-packages (from bleach!=5.0.0->bleach[css]!=5.0.0->nbconvert)
(0.5.1)
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in c:\users\subod\appdata\roa
ming\python\python312\site-packages (from bleach[css]!=5.0.0->nbconvert) (1.4.0)
Requirement already satisfied: platformdirs>=2.5 in c:\users\subod\appdata\roamin
g\python\python312\site-packages (from jupyter-core>=4.7->nbconvert) (4.3.6)
Requirement already satisfied: pywin32>=300 in c:\users\subod\appdata\roaming\pyt
hon\python312\site-packages (from jupyter-core>=4.7->nbconvert) (308)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\subod\appdata\r
oaming\python\python312\site-packages (from nbclient>=0.5.0->nbconvert) (8.6.3)
Requirement already satisfied: fastjsonschema>=2.15 in c:\users\subod\appdata\roa
ming\python\python312\site-packages (from nbformat>=5.7->nbconvert) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in c:\users\subod\appdata\roaming
\python\python312\site-packages (from nbformat>=5.7->nbconvert) (4.23.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\subod\appdata\roaming\py
thon\python312\site-packages (from beautifulsoup4->nbconvert) (2.6)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\subod\appdata
\roaming\python\python312\site-packages (from beautifulsoup4->nbconvert) (4.12.2)
Requirement already satisfied: attrs>=22.2.0 in c:\users\subod\appdata\roaming\py
thon\python312\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (2
5.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\s
ubod\appdata\roaming\python\python312\site-packages (from jsonschema>=2.6->nbform
at>=5.7->nbconvert) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\subod\appdata\roam
ing\python\python312\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconver
t) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\subod\appdata\roaming\p
ython\python312\site-packages (from jsonschema>=2.6->nbformat>=5.7->nbconvert)
```

```
(0.23.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\subod\appdata\r
oaming\python\python312\site-packages (from jupyter-client>=6.1.12->nbclient>=0.
5.0->nbconvert) (2.9.0.post0)
Requirement already satisfied: pyzmq>=23.0 in c:\users\subod\appdata\roaming\pyth
on\python312\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconve
rt) (26.2.1)
Requirement already satisfied: tornado>=6.2 in c:\users\subod\appdata\roaming\pyt
hon\python312\site-packages (from jupyter-client>=6.1.12->nbclient>=0.5.0->nbconv
ert) (6.4.2)
Requirement already satisfied: six>=1.5 in c:\users\subod\appdata\roaming\python
\python312\site-packages (from python-dateutil>=2.8.2->jupyter-client>=6.1.12->nb
client>=0.5.0->nbconvert) (1.17.0)
```

In [21]:
```
$ jupyter nbconvert --to LATEX notebook.ipynb
```

```
  Cell In[21], line 1
    $ jupyter nbconvert --to LATEX notebook.ipynb
    ^
SyntaxError: invalid syntax
```

In [23]:
```
jupyter nbconvert --to LATEX notebook.ipynb
```

```
  Cell In[23], line 1
    jupyter nbconvert --to LATEX notebook.ipynb
            ^
SyntaxError: invalid syntax
```

In [ ]:
```
```