In [15]:

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import matplotlib as plt
```

In [16]:

```
!pip install spotipy
```

```
Requirement already satisfied: spotipy in ./anaconda3/lib/python3.11/
site-packages (2.23.0)
Requirement already satisfied: redis>=3.5.3 in ./anaconda3/lib/python
3.11/site-packages (from spotipy) (5.0.2)
Requirement already satisfied: requests>=2.25.0 in ./anaconda3/lib/py
thon3.11/site-packages (from spotipy) (2.31.0)
Requirement already satisfied: six>=1.15.0 in ./anaconda3/lib/python
3.11/site-packages (from spotipy) (1.16.0)
Requirement already satisfied: urllib3>=1.26.0 in ./anaconda3/lib/pyt
hon3.11/site-packages (from spotipy) (1.26.16)
Requirement already satisfied: async-timeout>=4.0.3 in ./anaconda3/li
b/python3.11/site-packages (from redis>=3.5.3->spotipy) (4.0.3)
Requirement already satisfied: charset-normalizer<4,>=2 in ./anaconda
3/lib/python3.11/site-packages (from requests>=2.25.0->spotipy) (2.0.
4)
Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python
3.11/site-packages (from requests>=2.25.0->spotipy) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in ./anaconda3/lib/
python3.11/site-packages (from requests>=2.25.0->spotipy) (2023.7.22)
```

In [17]:

```
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
```

In [18]:

```
# Replace these with your Spotify API credentials
client_id = os.getenv('SPOTIFY_CLIENT_ID')
client_secret = os.getenv('SPOTIFY_CLIENT_SECRET')

auth_manager = SpotifyClientCredentials(client_id=client_id, client_se
sp = spotipy.Spotify(auth_manager=auth_manager)
```

In [19]:
```
def search_playlists(sp, query, limit=10):
    playlists = sp.search(q=f'playlist:"{query}"', type='playlist', li
    return [playlist['id'] for playlist in playlists['playlists']['ite

def get_tracks_from_playlists(sp, playlist_ids):
    track_ids = []
    for playlist_id in playlist_ids:
        results = sp.playlist_tracks(playlist_id)
        track_ids.extend([item['track']['id'] for item in results['ite
    return track_ids
```

In [20]:
```
def fetch_audio_features(sp, track_ids):
    features_list = []
    for track_id in track_ids:
        features = sp.audio_features(track_id)[0]
        if features:
            selected_features = {
                'track_id': track_id,
                'danceability': features['danceability'],
                'energy': features['energy'],
                'valence': features['valence'],
                'tempo': features['tempo'],
                'acousticness': features['acousticness']
            }
            features_list.append(selected_features)
    return features_list
```

In [21]:
```
# Example: Adjust the limit as needed, considering API rate limits and
happy_playlist_ids = search_playlists(sp, "Happy", limit=5)
sad_playlist_ids = search_playlists(sp, "Sad", limit=5)

# Fetch track IDs
happy_track_ids = get_tracks_from_playlists(sp, happy_playlist_ids)
sad_track_ids = get_tracks_from_playlists(sp, sad_playlist_ids)

# Fetch audio features and label them
happy_features = fetch_audio_features(sp, happy_track_ids)
for feature in happy_features:
    feature['mood'] = 'Happy'

sad_features = fetch_audio_features(sp, sad_track_ids)
for feature in sad_features:
    feature['mood'] = 'Sad'

# Combine happy and sad features
all_features = happy_features + sad_features

# Create DataFrame
df = pd.DataFrame(all_features)

# Check the combined DataFrame
print(df.head())  # For the first few rows
print(df.tail())  # For the last few rows to verify sad tracks are inc
print(df['mood'].value_counts())  # To check the distribution of moods

# Save to CSV
df.to_csv('spotify_mood_audio_features.csv', index=False)
```

```
                 track_id  danceability  energy  valence    tempo  \
0  3EslM7q3nwkGRvY7HNbGBx         0.754   0.543    0.458  104.678   
1  2okho7vU7Nsq1UZD0kgIMi         0.598   0.714    0.467   99.979   
2  53Iv1sdaJ8TYFfhtTBGyv0         0.728   0.491    0.700  129.988   
3  77sMIMlNaSURUAXq5coCxE         0.569   0.741    0.430  100.118   
4  0Oa9mHOnH14GZ4toWwkf4c         0.572   0.778    0.540  168.073   

   acousticness   mood  
0       0.09400  Happy  
1       0.03640  Happy  
2       0.84100  Happy  
3       0.01220  Happy  
4       0.00638  Happy  
                   track_id  danceability  energy  valence    tempo  
\
656  0V5cvmTKsYmF5FmGGEAfmS         0.303  0.1870   0.2130  132.731   
657  3IYU2BjHdPyTmuk81fUZXZ         0.499  0.2070   0.1890   75.036   
658  2ZosoKioRi0SEFODWFKfsJ         0.461  0.0605   0.1950  126.556   
659  3eTgg18rKBD30Hef1gv0wz         0.312  0.0766   0.0493  136.866   
660  4R2kfaDFhslZEMJqAFNpdd         0.613  0.5810   0.5510  130.033   

     acousticness mood  
656         0.989  Sad  
657         0.912  Sad  
658         0.980  Sad  
659         0.967  Sad  
660         0.537  Sad  
mood
Happy    335
Sad      326
Name: count, dtype: int64
```

In [22]:

```
from IPython.display import FileLink
FileLink('spotify_mood_audio_features.csv')
```

Out[22]:

spotify_mood_audio_features.csv (spotify_mood_audio_features.csv)

In [23]:
```
# Display the first few rows to verify the content
print(df.head())
```

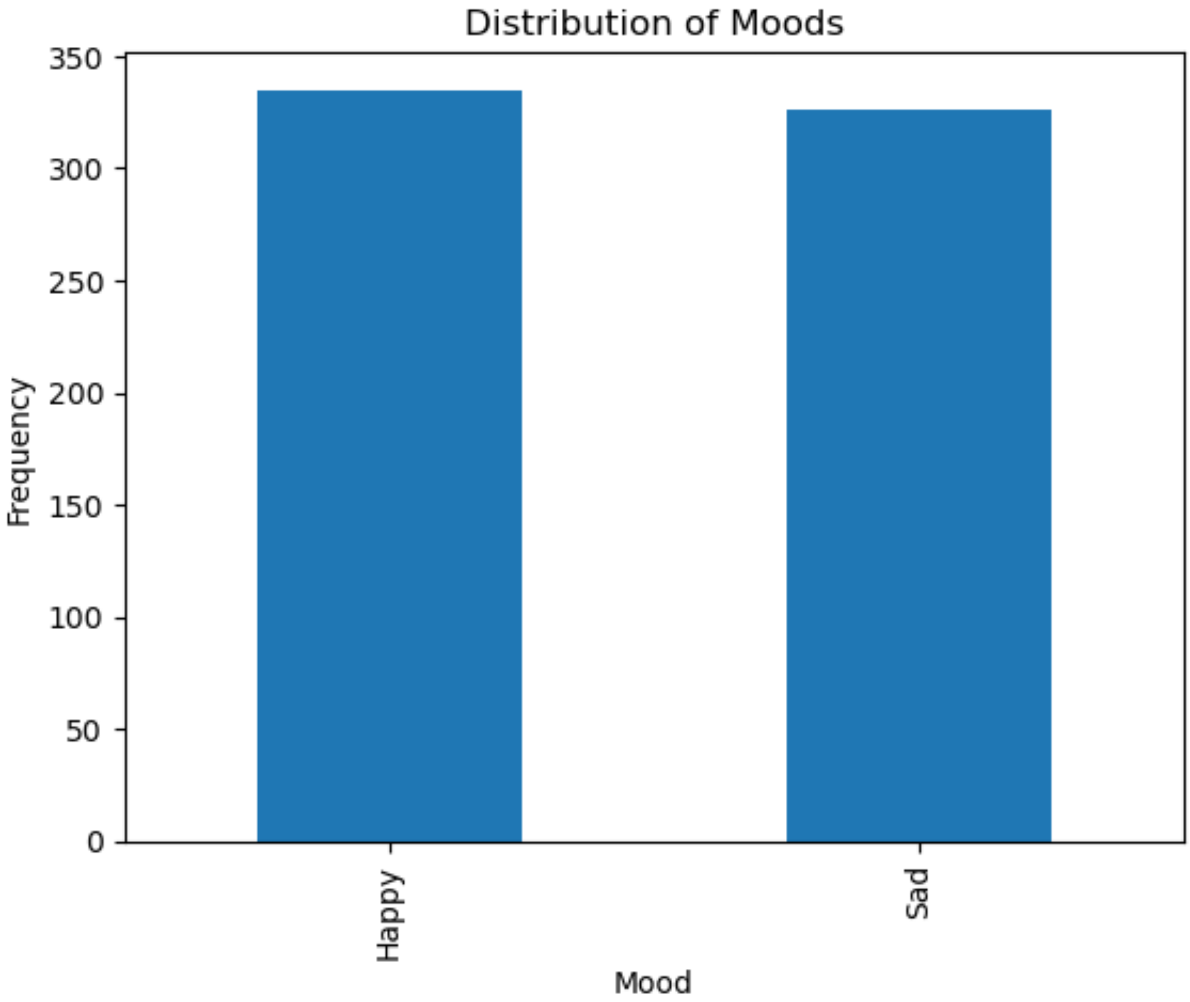
```
                 track_id  danceability  energy  valence    tempo  \
0  3EslM7q3nwkGRvY7HNbGBx         0.754   0.543    0.458  104.678   
1  2okho7vU7Nsq1UZD0kgIMi         0.598   0.714    0.467   99.979   
2  53Iv1sdaJ8TYFfhtTBGyv0         0.728   0.491    0.700  129.988   
3  77sMIMlNaSURUAXq5coCxE         0.569   0.741    0.430  100.118   
4  0Oa9mHOnH14GZ4toWwkf4c         0.572   0.778    0.540  168.073   

   acousticness   mood  
0       0.09400  Happy  
1       0.03640  Happy  
2       0.84100  Happy  
3       0.01220  Happy  
4       0.00638  Happy  
```

In [24]:
```
# Display the shape of the dataframe
print("Shape of the DataFrame:", df.shape)

# Check for missing values
print("Missing values in each column:\n", df.isnull().sum())
```

```
Shape of the DataFrame: (661, 7)
Missing values in each column:
 track_id        0
danceability    0
energy          0
valence         0
tempo           0
acousticness    0
mood            0
dtype: int64
```

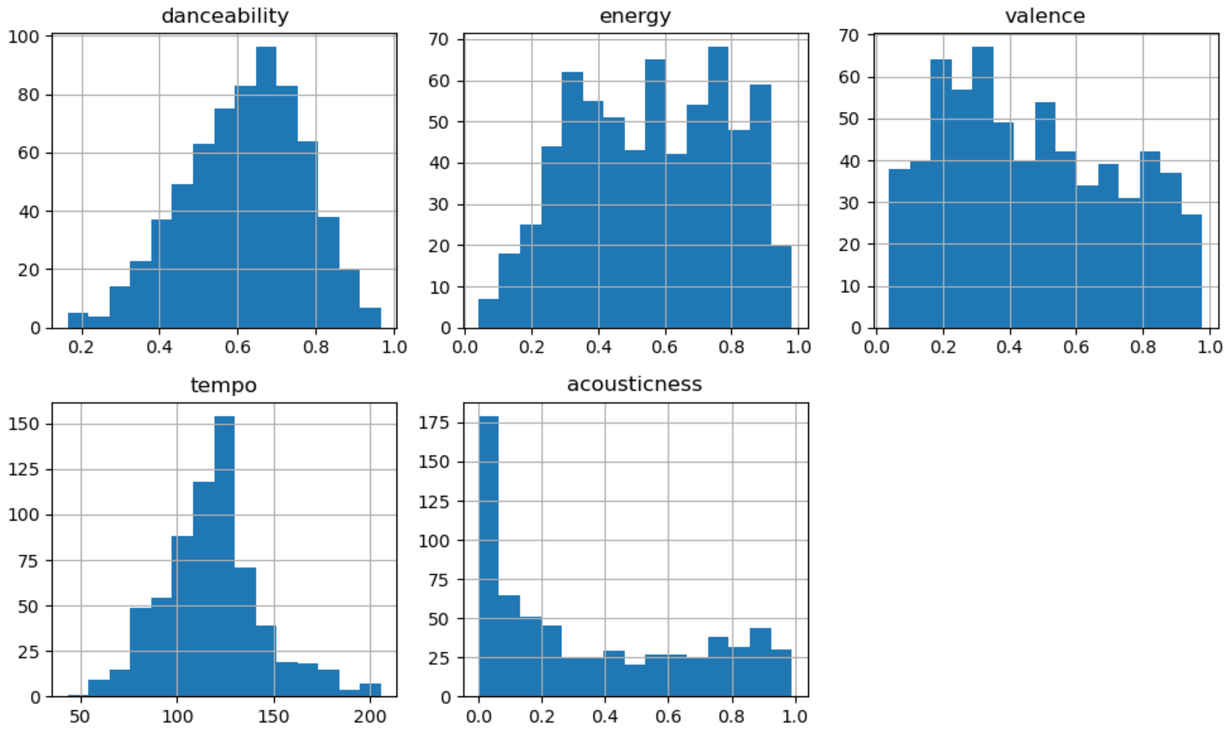
In [26]:

```
import matplotlib.pyplot as plt
# Plotting the distribution of moods
df['mood'].value_counts().plot(kind='bar')
plt.title('Distribution of Moods')
plt.xlabel('Mood')
plt.ylabel('Frequency')
plt.show()
```



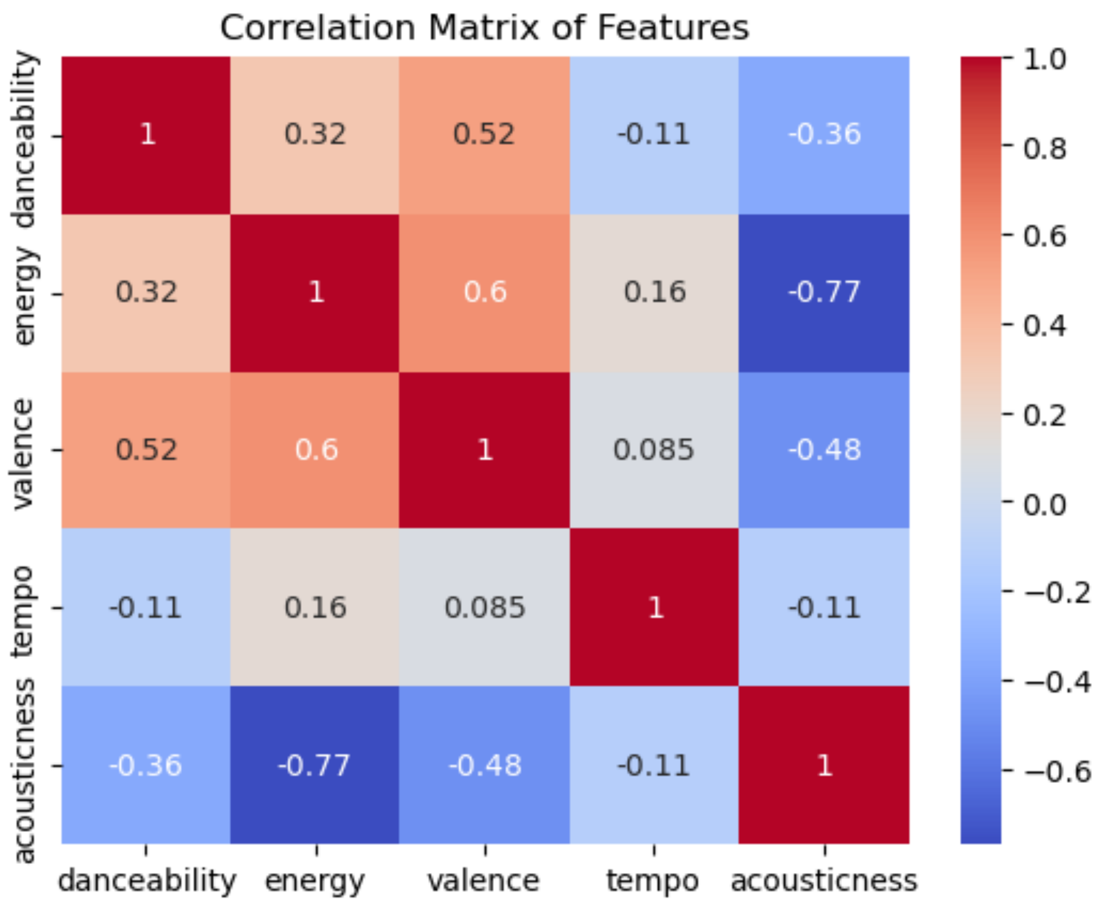
In [27]:

```
# Visualizing distributions of numeric features
features = ['danceability', 'energy', 'valence', 'tempo', 'acousticnes
df[features].hist(bins=15, figsize=(10, 6), layout=(2, 3))
plt.tight_layout()
plt.show()
```



In [29]:

```
import seaborn as sns

# Heatmap of feature correlations
sns.heatmap(df[features].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Features')
plt.show()
```



In [31]:

```
from sklearn.preprocessing import LabelEncoder

# Encoding the 'mood' column
df['mood_encoded'] = LabelEncoder().fit_transform(df['mood'])
```

In [32]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

In [33]:

```
# Splitting the dataset
X_train, X_test, y_train, y_test = train_test_split(df[features], df['

# Initializing and training the Random Forest model
model_rf = RandomForestClassifier(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)

# Making predictions
y_pred_rf = model_rf.predict(X_test)
```

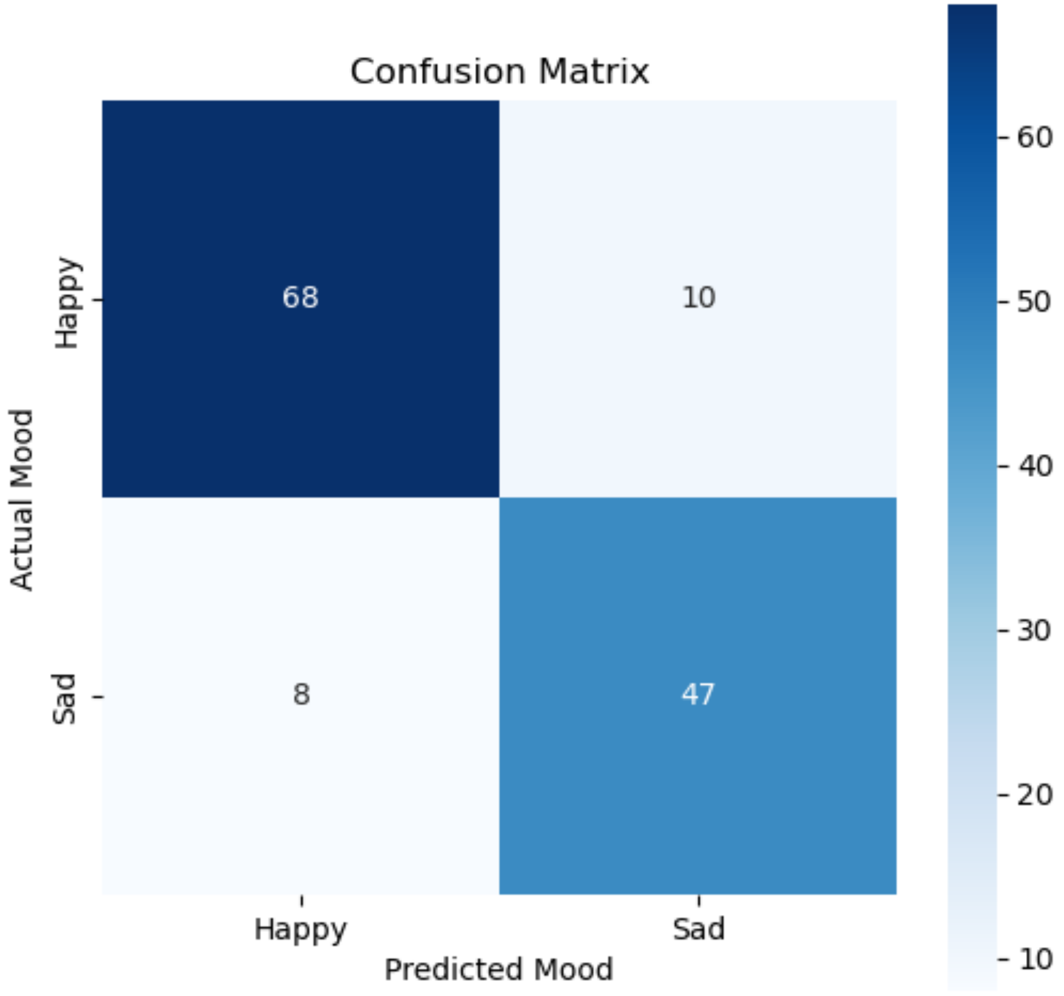
In [36]:

```
from sklearn.metrics import accuracy_score,classification_report

accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Accuracy of the Random Forest classifier:", accuracy_rf)
print(classification_report(y_test, y_pred_rf))
```

```
Accuracy of the Random Forest classifier: 0.8646616541353384
              precision    recall  f1-score   support

           0       0.89      0.87      0.88        78
           1       0.82      0.85      0.84        55

    accuracy                           0.86       133
   macro avg       0.86      0.86      0.86       133
weighted avg       0.87      0.86      0.86       133
```

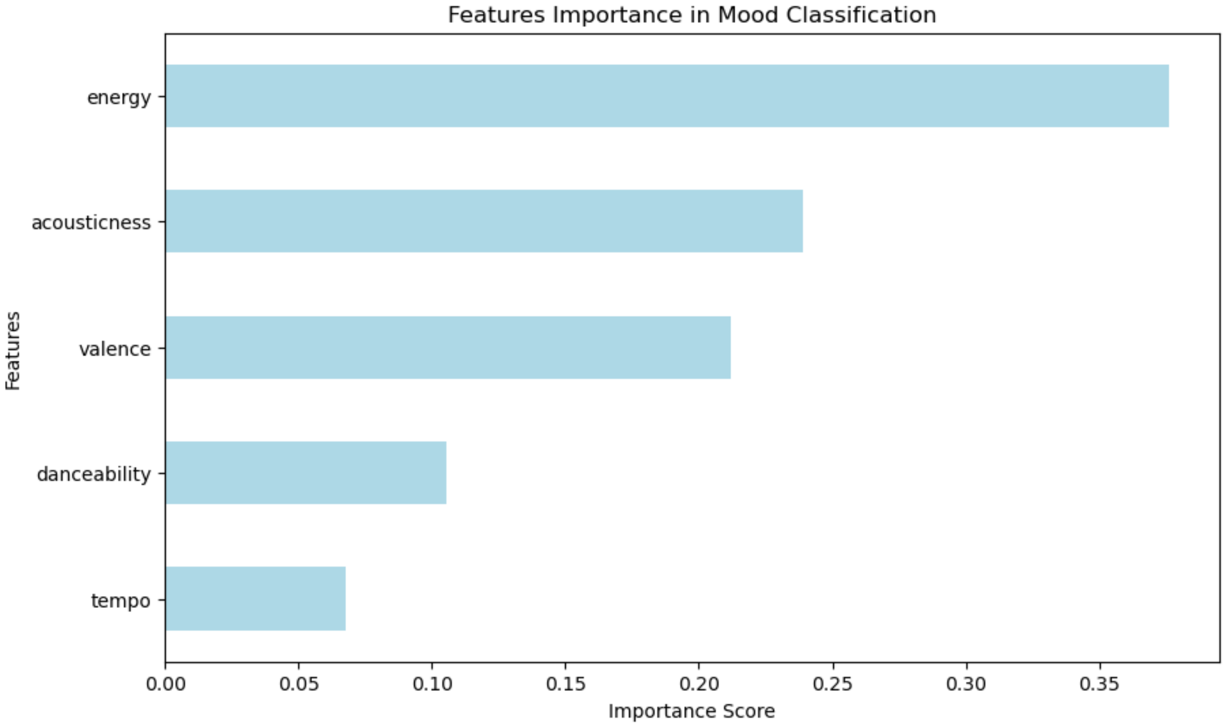
In [37]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred_rf)

# Plot the confusion matrix using Seaborn
plt.figure(figsize=(6,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", square=True,
            xticklabels=['Happy', 'Sad'],
            yticklabels=['Happy', 'Sad'])
plt.xlabel('Predicted Mood')
plt.ylabel('Actual Mood')
plt.title('Confusion Matrix')
plt.show()
```



In [40]:

```
# Extract feature importances
feature_importances = model_rf.feature_importances_

# Create a pandas series to hold the feature names and their importanc
importances = pd.Series(feature_importances, index=features)

# Sort the feature importances in descending order
importances_sorted = importances.sort_values()

# Visualize the feature importances
plt.figure(figsize=(10, 6))
importances_sorted.plot(kind='barh', color='lightblue')
plt.title('Features Importance in Mood Classification')
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.show()
```



In [ ]: