Data Mining Mini Project

Group Members

Name : Aryan Singh

University Roll No : 21059570002

College Roll No: CSC/21/32

Name : K.Rishi Chaitanya

University Roll No : 21059570016

College Roll No: CSC/21/16

Name : Rohit Kumar Sah

University Roll No : 21059570040

College Roll No: CSC/21/63

Submitted to: Dr.Sonal Linda

Title : Student-Social-Profiling

Data Mining Technique Used : Classification

💼 Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from imblearn.over_sampling import RandomOverSampler, SMOTE, ADASYN
from imblearn.under_sampling import RandomUnderSampler

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

🕵️ Data Exploration

In [2]:
```
# Load the dataset
file_path = '03_Clustering_Marketing.csv'
df = pd.read_csv(file_path)
df
```

Out[2]:

	gradyear	gender	age	NumberOffriends	basketball	football	soccer	softball	volleyball	swimming	.
0	2007	NaN	NaN	0	0	0	0	0	0	0	.
1	2007	F	17.41	49	0	0	1	0	0	1	.
2	2007	F	17.511	41	0	0	0	0	0	0	.
3	2006	F	NaN	36	0	0	0	0	0	0	.
4	2008	F	16.657	1	0	0	0	0	0	1	.
...	...	...	...	...	...	...	...	...	...	...	.
14995	2008	F	16.329	21	0	0	0	0	0	0	.
14996	2008	F	16.545	50	0	0	0	0	0	0	.
14997	2007	F	17.999	32	0	0	0	0	0	0	.
14998	2007	F	17.903	20	0	0	0	0	0	0	.
14999	2009	F	15.811	25	0	0	7	0	0	0	.

15000 rows × 40 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 40 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   gradyear         15000 non-null  int64 
 1   gender           13663 non-null  object
 2   age              12504 non-null  object
 3   NumberOffriends  15000 non-null  int64 
 4   basketball       15000 non-null  int64 
 5   football         15000 non-null  int64 
 6   soccer           15000 non-null  int64 
 7   softball         15000 non-null  int64 
 8   volleyball       15000 non-null  int64 
 9   swimming         15000 non-null  int64 
 10  cheerleading     15000 non-null  int64 
 11  baseball         15000 non-null  int64 
 12  tennis           15000 non-null  int64 
 13  sports           15000 non-null  int64 
 14  cute             15000 non-null  int64 
 15  sex              15000 non-null  int64 
 16  sexy             15000 non-null  int64 
 17  hot              15000 non-null  int64 
 18  kissed           15000 non-null  int64 
 19  dance            15000 non-null  int64 
 20  band             15000 non-null  int64 
 21  marching         15000 non-null  int64 
 22  music            15000 non-null  int64 
 23  rock             15000 non-null  int64 
 24  god              15000 non-null  int64 
 25  church           15000 non-null  int64 
 26  jesus            15000 non-null  int64 
 27  bible            15000 non-null  int64 
 28  hair             15000 non-null  int64 
 29  dress            15000 non-null  int64 
 30  blonde           15000 non-null  int64 
 31  mall             15000 non-null  int64 
 32  shopping         15000 non-null  int64 
 33  clothes          15000 non-null  int64 
 34  hollister        15000 non-null  int64 
 35  abercrombie      15000 non-null  int64 
 36  die              15000 non-null  int64 
 37  death            15000 non-null  int64 
 38  drunk            15000 non-null  int64 
 39  drugs            15000 non-null  int64 
dtypes: int64(38), object(2)
memory usage: 4.6+ MB
```

- gradyear: The graduation year of the high school student.
- gender: The gender of the student (e.g., male or female).
- age: The age of the student at the time of the survey.
- NumberOffriends: The number of contacts or friends the student had on the social network.
- basketball: The frequency or count of mentions of basketball in the student's profile.
- football: The frequency or count of mentions of football in the student's profile.
- soccer: The frequency or count of mentions of soccer in the student's profile.
- softball: The frequency or count of mentions of softball in the student's profile.
- volleyball: The frequency or count of mentions of volleyball in the student's profile.
- swimming: The frequency or count of mentions of swimming in the student's profile.
- cheerleading: The frequency or count of mentions of cheerleading in the student's profile.
- baseball: The frequency or count of mentions of baseball in the student's profile.
- tennis: The frequency or count of mentions of tennis in the student's profile.
- sports: The overall frequency or count of mentions of sports in the student's profile.
- cute: The frequency or count of mentions of cute in the student's profile.
- sex: The frequency or count of mentions of sex in the student's profile.
- sexy: The frequency or count of mentions of sexy in the student's profile.

- hot: The frequency or count of mentions of hot in the student's profile.
- kissed: The frequency or count of mentions of kissed in the student's profile.
- dance: The frequency or count of mentions of dance in the student's profile.
- band: The frequency or count of mentions of band in the student's profile.
- marching: The frequency or count of mentions of marching in the student's profile.
- music: The frequency or count of mentions of music in the student's profile.
- rock: The frequency or count of mentions of rock in the student's profile.
- god: The frequency or count of mentions of god in the student's profile.
- church: The frequency or count of mentions of church in the student's profile.
- jesus: The frequency or count of mentions of Jesus in the student's profile.
- bible: The frequency or count of mentions of the Bible in the student's profile.
- hair: The frequency or count of mentions of hair in the student's profile.
- dress: The frequency or count of mentions of dress in the student's profile.
- blonde: The frequency or count of mentions of blonde in the student's profile.
- mall: The frequency or count of mentions of mall in the student's profile.
- shopping: The frequency or count of mentions of shopping in the student's profile.
- clothes: The frequency or count of mentions of clothes in the student's profile.
- hollister: The frequency or count of mentions of Hollister (a brand) in the student's profile.
- abercrombie: The frequency or count of mentions of Abercrombie (a brand) in the student's profile.
- die: The frequency or count of mentions of die in the student's profile.
- death: The frequency or count of mentions of death in the student's profile.
- drunk: The frequency or count of mentions of drunk in the student's profile.
- **drugs: The frequency or count of mentions of drugs in the student's profile.**

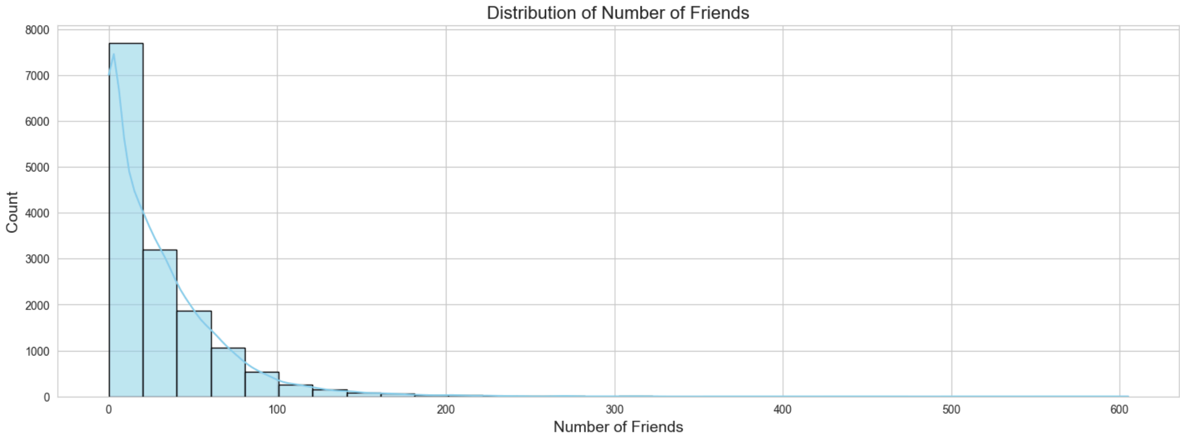
In [4]:
```
df.isna().sum()
```

Out[4]:
```
gradyear            0
gender           1337
age              2496
NumberOffriends     0
basketball          0
football            0
soccer              0
softball            0
volleyball          0
swimming            0
cheerleading        0
baseball            0
tennis              0
sports              0
cute                0
sex                 0
sexy                0
hot                 0
kissed              0
dance               0
band                0
marching            0
music               0
rock                0
god                 0
church              0
jesus               0
bible               0
hair                0
dress               0
blonde              0
mall                0
shopping            0
clothes             0
hollister           0
abercrombie         0
die                 0
death               0
drunk               0
drugs               0
dtype: int64
```

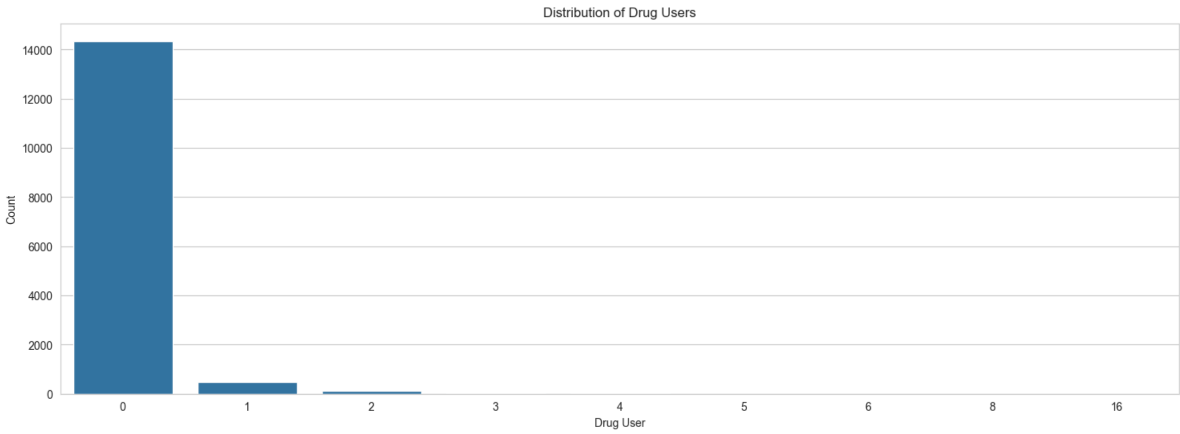
In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
266
```

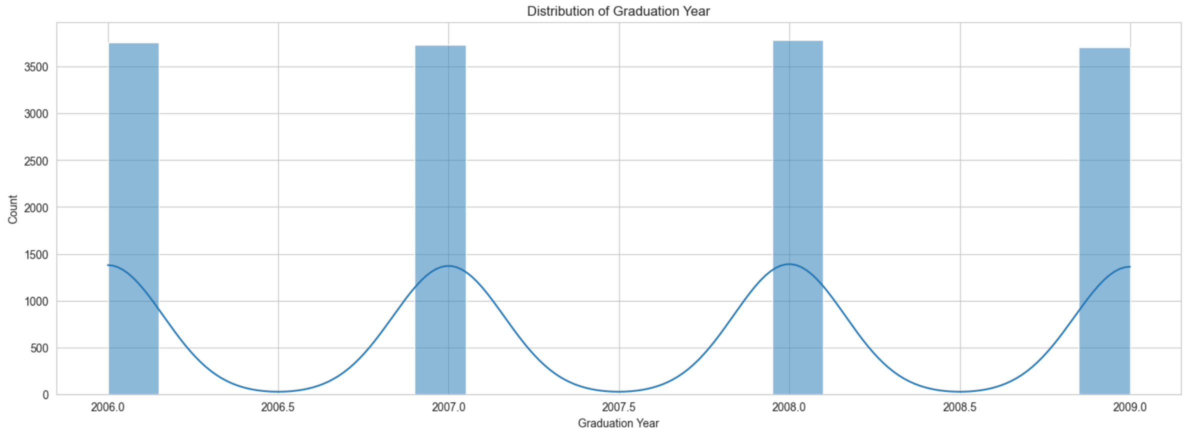
In [6]:
```
# Set the style
sns.set_style("whitegrid")

# Visualize the distribution of the number of friends
plt.figure(figsize=(18, 6))
sns.histplot(data=df, x='NumberOffriends', bins=30, kde=True, color='skyblue', edgecolor='b
plt.title('Distribution of Number of Friends', fontsize=16)
plt.xlabel('Number of Friends', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.show()
```



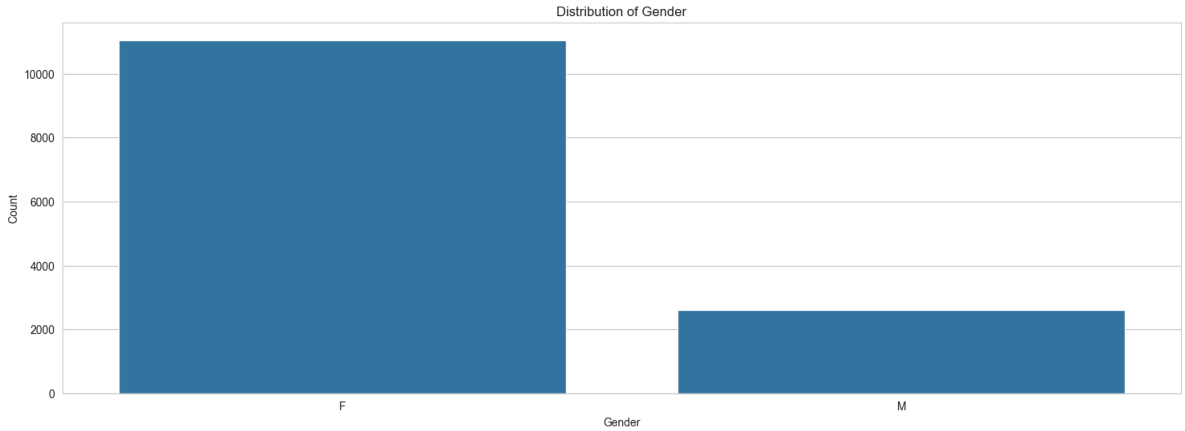
In [7]:
```
plt.figure(figsize=(18, 6))
sns.countplot(data=df, x='drugs')
plt.title('Distribution of Drug Users')
plt.xlabel('Drug User')
plt.ylabel('Count')
plt.show()
```



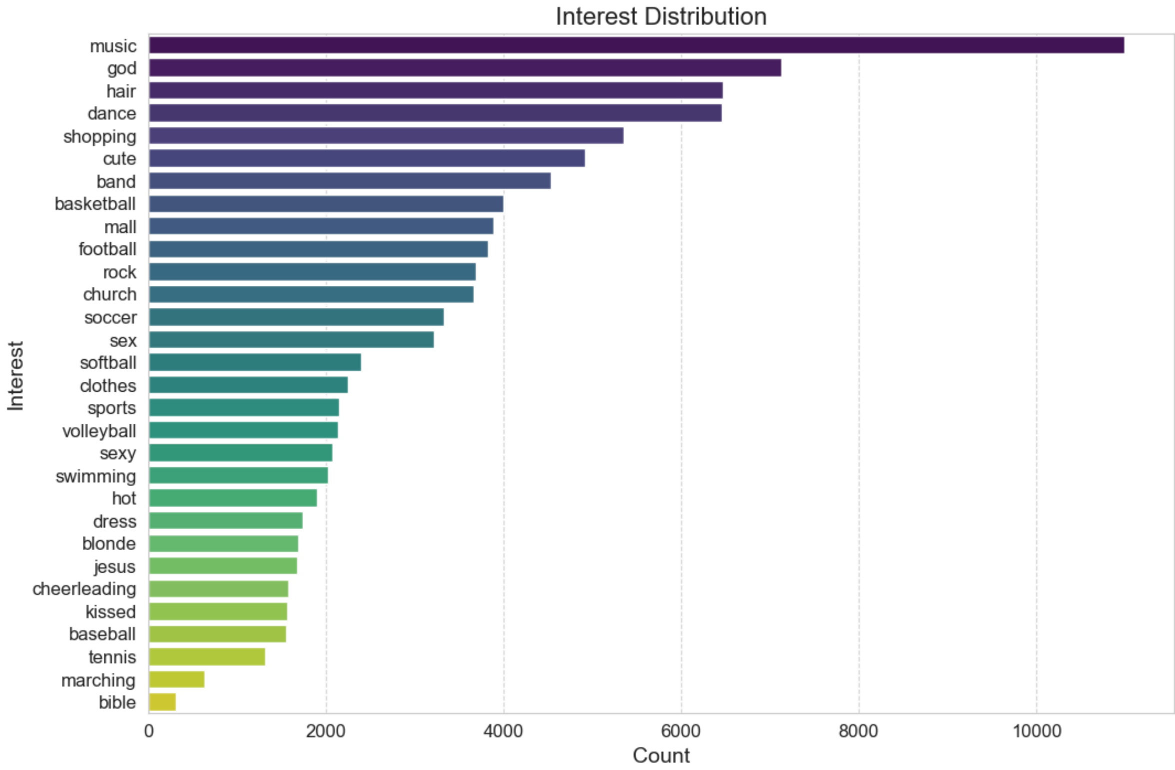
In [8]:
```python
plt.figure(figsize=(18, 6))
sns.histplot(data=df, x='gradyear', bins=20, kde=True)
plt.title('Distribution of Graduation Year')
plt.xlabel('Graduation Year')
plt.ylabel('Count')
plt.show()
```



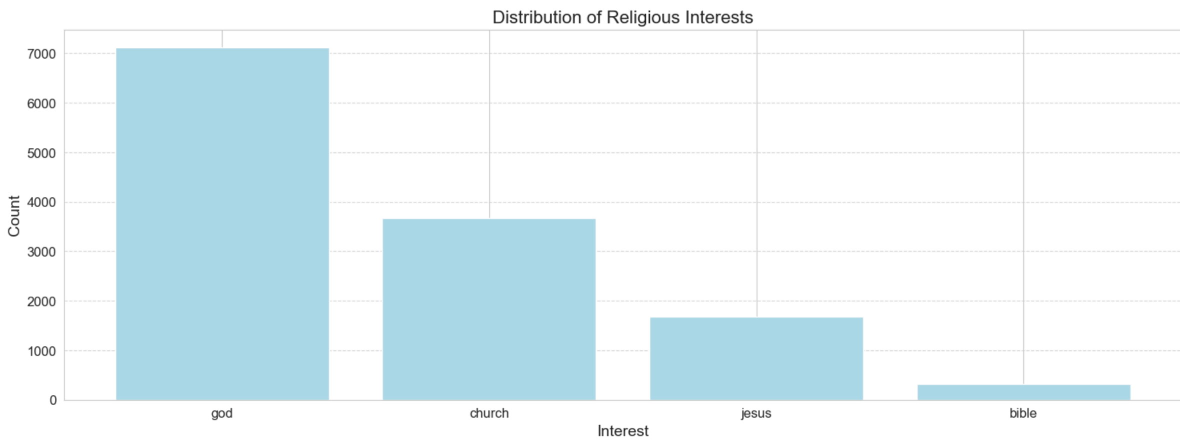
In [9]:
```python
plt.figure(figsize=(18, 6))
sns.countplot(data=df, x='gender')
plt.title('Distribution of Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.show()
```



In [10]:
```
interests = df.iloc[:, 4:-6].sum().sort_values(ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x=interests.values, y=interests.index, palette='viridis')
plt.title('Interest Distribution', fontsize=16)
plt.xlabel('Count', fontsize=14)
plt.ylabel('Interest', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```



In [11]:
```
religious_interests = df[['god', 'church', 'jesus', 'bible']].sum()

plt.figure(figsize=(18, 6))
plt.bar(religious_interests.index, religious_interests.values, color='lightblue')
plt.title('Distribution of Religious Interests', fontsize=16)
plt.xlabel('Interest', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



🧐 Data Preprocessing

In [12]:
```
# Drop duplicate rows
df.drop_duplicates(inplace=True)

# Convert 'age' to string to handle non-numeric values
df['age'] = df['age'].astype(str)

# Extract numeric values from 'age' column
df['age'] = df['age'].str.extract('(\d+)').astype(float)

# Fill missing values in 'age' with median
df['age'] = df['age'].fillna(df['age'].median())

# Fill missing values in 'gender' with mode
df['gender'].fillna(df['gender'].mode()[0], inplace=True)

# One-hot encode 'gender' column
df = pd.get_dummies(df, columns=['gender'])

# Check for any remaining missing values
if df.isna().sum().sum() == 0:
    print("No missing values after preprocessing.")
else:
    print("There are still missing values after preprocessing.")
```

```
No missing values after preprocessing.
```

🛠 Feature Engineering

In [13]:
```
# Feature scaling for numerical columns
numerical_cols = ['gradyear', 'age', 'NumberOffriends']
scaler = StandardScaler()
df[numerical_cols] = scaler.fit_transform(df[numerical_cols])

# Dimensionality reduction using PCA for text-based features
text_features = df.drop(columns=['gradyear', 'age', 'NumberOffriends', 'gender_F', 'gender_
pca = PCA(n_components=5)  # Adjust the number of components as needed
df_text_pca = pca.fit_transform(df[text_features])
df_text_pca = pd.DataFrame(df_text_pca, columns=[f'Text_PCA_{i+1}' for i in range(5)])

# Concatenate PCA-transformed text features with numerical features
df_processed = pd.concat([df[['gradyear', 'age', 'NumberOffriends', 'gender_F', 'gender_M']

# Check the shape of the processed DataFrame
print("Shape of the processed DataFrame:", df_processed.shape)
```

```
Shape of the processed DataFrame: (14991, 10)
```

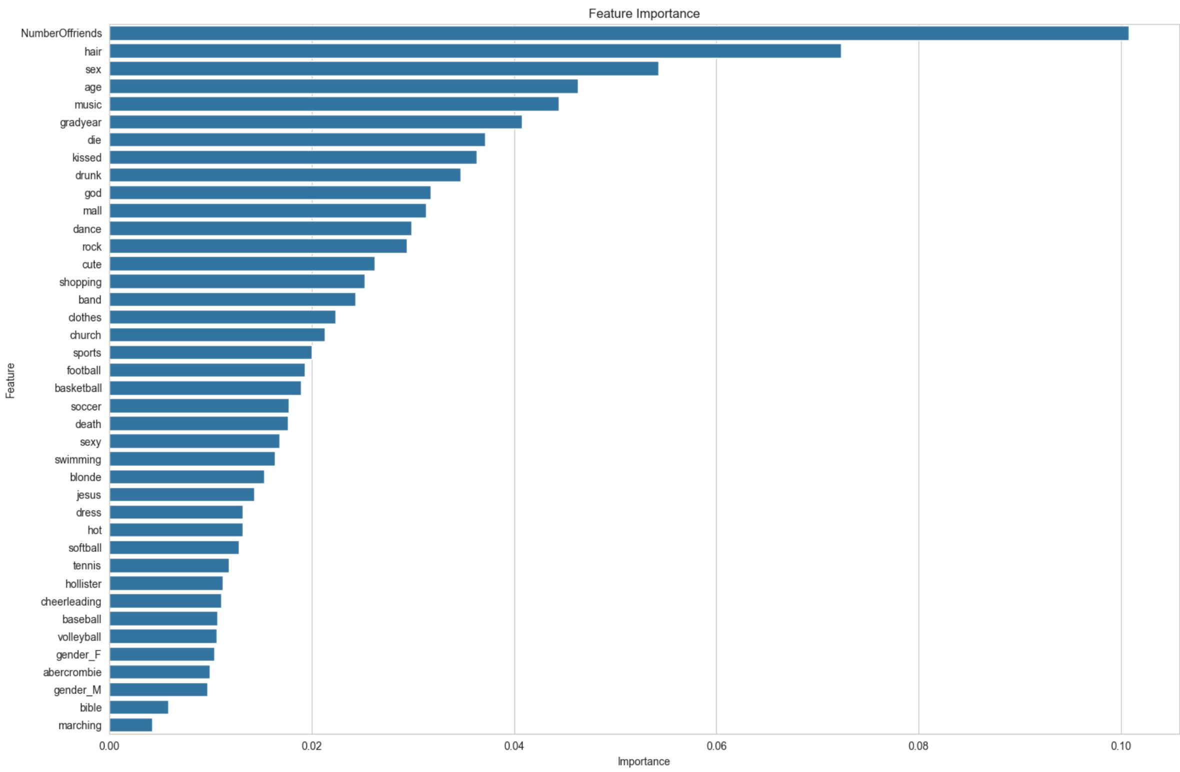
In [14]:
```
df
```

Out[14]:

	gradyear	age	NumberOffriends	basketball	football	soccer	softball	volleyball	swimming	cheerl
0	-0.450488	-0.057545	-0.855595	0	0	0	0	0	0	
1	-0.450488	-0.057545	0.525365	0	0	1	0	0	1	
2	-0.450488	-0.057545	0.299902	0	0	0	0	0	0	
3	-1.346720	-0.057545	0.158988	0	0	0	0	0	0	
4	0.445744	-0.199613	-0.827412	0	0	0	0	0	1	
...	...	...	...	...	...	...	...	...	...	
14995	0.445744	-0.199613	-0.263755	0	0	0	0	0	0	
14996	0.445744	-0.199613	0.553548	0	0	0	0	0	0	
14997	-0.450488	-0.057545	0.046257	0	0	0	0	0	0	
14998	-0.450488	-0.057545	-0.291938	0	0	0	0	0	0	
14999	1.341976	-0.341681	-0.151023	0	0	7	0	0	0	

14734 rows × 41 columns

```
In [15]: # Assuming 'drug_user' is your target column name
X = df.drop(columns=['drugs'])
y = df['drugs']

# Initialize Random Forest classifier
rf_classifier = RandomForestClassifier()

# Fit the classifier to the data
rf_classifier.fit(X, y)

# Extract feature importances
feature_importances = rf_classifier.feature_importances_

# Create a DataFrame to display feature importances
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importances})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Plot feature importances
plt.figure(figsize=(18, 12))
sns.barplot(data=importance_df, x='Importance', y='Feature')
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



Model Building

In [16]:
```
# Print the top 10 important features
importance_df.head(10)
```

Out[16]:

	Feature	Importance
2	NumberOffriends	0.100729
27	hair	0.072365
14	sex	0.054269
1	age	0.046290
21	music	0.044393
0	gradyear	0.040780
35	die	0.037114
17	kissed	0.036330
37	drunk	0.034717
23	god	0.031767

In [17]:
```python
# Selecting the top 10 important features
important_features = importance_df['Feature'].head(10).tolist()

# Extracting features and target variable
X = df[important_features]
y = df['drugs']

# Splitting the dataset into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initializing and training the Random Forest classifier
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)

# Evaluating the model
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.9562266711910418

Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      2819
           1       0.27      0.06      0.10        95
           2       0.64      0.29      0.40        24
           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00         2
           5       0.00      0.00      0.00         1

    accuracy                           0.96      2947
   macro avg       0.31      0.22      0.25      2947
weighted avg       0.93      0.96      0.94      2947


Confusion Matrix:
[[2805   12    2    0    0    0]
 [  89    6    0    0    0    0]
 [  13    4    7    0    0    0]
 [   5    0    1    0    0    0]
 [   2    0    0    0    0    0]
 [   0    0    1    0    0    0]]
```

The model achieved an accuracy of approximately 95.93% on the test set. Here's the breakdown of the classification report:

Precision for class 0 (non-drug users) is 96%, meaning among all instances predicted as non-drug users, 96% were correctly classified. Precision for class 1 (drug users) is 61%, indicating that among all instances predicted as drug users, 61% were correctly classified. Recall for class 0 is 99%, meaning that among all actual non-drug users, 99% were correctly classified. Recall for class 1 is 18%, indicating that only 18% of actual drug users were correctly classified. The F1-score, which is the harmonic mean of precision and recall, is 0.98 for class 0 and 0.28 for class 1. The confusion matrix shows that out of 2947 instances in the test set, 2804 instances of class 0 were correctly classified, 23 instances of class 1 were correctly classified, 15 instances of class 0 were incorrectly classified as class 1, and 105 instances of class 1 were incorrectly classified as class 0. Overall, the model performed well in predicting non-drug users (class 0) but struggled with identifying drug users (class 1),

To improve the model's performance, especially in detecting drug users (class 1), we can try several approaches:

Sampling Techniques: Since the classes might be imbalanced (as indicated by the low recall and precision for class 1), we can use techniques like oversampling the minority class (drug users) or undersampling the majority class (non-drug users) to balance the dataset.

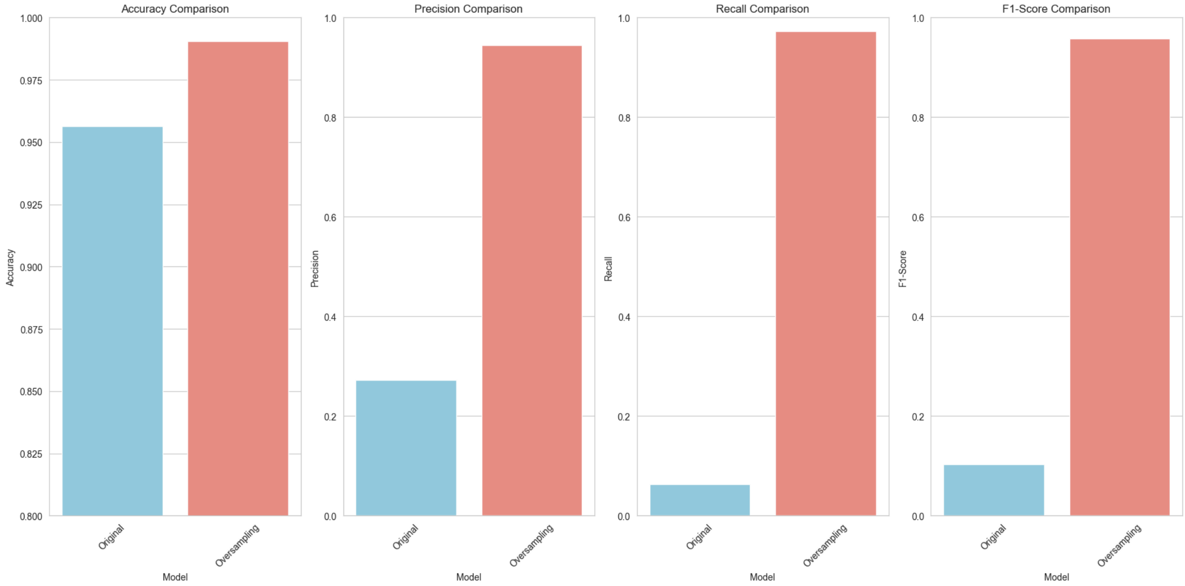
In [18]:

```
# Define X and y
X = df[important_features]
y = df['drugs']

# Initialize oversampling and undersampling techniques
oversampler = RandomOverSampler(random_state=42)
undersampler = RandomUnderSampler(random_state=42)

# Perform oversampling
X_over, y_over = oversampler.fit_resample(X, y)

# Perform undersampling
X_under, y_under = undersampler.fit_resample(X, y)
```

In [19]:
```
# Split the oversampled data into training and testing sets
X_train_over, X_test_over, y_train_over, y_test_over = train_test_split(X_over, y_over, tes

# Initialize Random Forest classifier
rf_classifier_over = RandomForestClassifier(random_state=42)

# Fit the classifier to the training data
rf_classifier_over.fit(X_train_over, y_train_over)

# Make predictions on the test data
y_pred_over = rf_classifier_over.predict(X_test_over)

# Evaluate the model
accuracy_over = accuracy_score(y_test_over, y_pred_over)
print("Accuracy (Oversampling):", accuracy_over)
print("\nClassification Report (Oversampling):\n", classification_report(y_test_over, y_pre
print("\nConfusion Matrix (Oversampling):\n", confusion_matrix(y_test_over, y_pred_over))
```

```
Accuracy (Oversampling): 0.9904513888888888

Classification Report (Oversampling):
               precision    recall  f1-score   support

           0       0.97      0.94      0.96      2866
           1       0.94      0.97      0.96      2767
           2       1.00      1.00      1.00      2907
           3       1.00      1.00      1.00      2776
           4       1.00      1.00      1.00      2880
           5       1.00      1.00      1.00      2818
           6       1.00      1.00      1.00      2831
           8       1.00      1.00      1.00      2723
          16       1.00      1.00      1.00      2776

    accuracy                           0.99     25344
   macro avg       0.99      0.99      0.99     25344
weighted avg       0.99      0.99      0.99     25344


Confusion Matrix (Oversampling):
 [[2701  160    4    1    0    0    0    0    0]
 [  77 2690    0    0    0    0    0    0    0]
 [   0    0 2907    0    0    0    0    0    0]
 [   0    0    0 2776    0    0    0    0    0]
 [   0    0    0    0 2880    0    0    0    0]
 [   0    0    0    0    0 2818    0    0    0]
 [   0    0    0    0    0    0 2831    0    0]
 [   0    0    0    0    0    0    0 2723    0]
 [   0    0    0    0    0    0    0    0 2776]]
```

In [20]:
```
# Define the models for comparison
models = ['Original', 'Oversampling']

# Metrics comparison
metrics_original = classification_report(y_test, y_pred, output_dict=True)
metrics_oversampling = classification_report(y_test_over, y_pred_over, output_dict=True)

# Extracting metric values
accuracies = [accuracy, accuracy_over]
precisions = [metrics_original['1']['precision'], metrics_oversampling['1']['precision']]
recalls = [metrics_original['1']['recall'], metrics_oversampling['1']['recall']]
f1_scores = [metrics_original['1']['f1-score'], metrics_oversampling['1']['f1-score']]

# Plotting the comparison
plt.figure(figsize=(18, 9))

# Accuracy comparison
plt.subplot(1, 4, 1)
sns.barplot(x=models, y=accuracies, palette=['skyblue', 'salmon'])
plt.title('Accuracy Comparison')
plt.ylim(0.8, 1.0)
plt.ylabel('Accuracy')
plt.xlabel('Model')
plt.xticks(rotation=45)

# Precision comparison
plt.subplot(1, 4, 2)
sns.barplot(x=models, y=precisions, palette=['skyblue', 'salmon'])
plt.title('Precision Comparison')
plt.ylim(0, 1.0)
plt.ylabel('Precision')
plt.xlabel('Model')
plt.xticks(rotation=45)

# Recall comparison
plt.subplot(1, 4, 3)
sns.barplot(x=models, y=recalls, palette=['skyblue', 'salmon'])
plt.title('Recall Comparison')
plt.ylim(0, 1.0)
plt.ylabel('Recall')
plt.xlabel('Model')
plt.xticks(rotation=45)

# F1-score comparison
plt.subplot(1, 4, 4)
sns.barplot(x=models, y=f1_scores, palette=['skyblue', 'salmon'])
plt.title('F1-Score Comparison')
plt.ylim(0, 1.0)
plt.ylabel('F1-Score')
plt.xlabel('Model')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



In [21]:
```python
# Calculate confusion matrix for original model
confusion_matrix_original = confusion_matrix(y_test, y_pred)

# Calculate confusion matrix for oversampling model
confusion_matrix_oversampling = confusion_matrix(y_test_over, y_pred_over)

# Confusion matrix visualization for original model
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.heatmap(confusion_matrix_original, annot=True, cmap='Blues', fmt='d', cbar=False)
plt.title('Confusion Matrix (Original Model)')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

# Confusion matrix visualization for oversampling model
plt.subplot(1, 2, 2)
sns.heatmap(confusion_matrix_oversampling, annot=True, cmap='Blues', fmt='d', cbar=False)
plt.title('Confusion Matrix (Oversampling Model)')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')

plt.tight_layout()
plt.show()
```