TASK-1

BUSINESS PROBLEM-Create a bar chart or histogram to visualize the distribution of a categorical or continuous variable, such as the distribution 0f ages or genders in a population.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
=pd.read_csv(r"C:\Users\Kokat\Downloads\prodigy_tech\prodigy_tech\SP.POP.TOTL\adult.csv")
.head()
```

Out[3]:

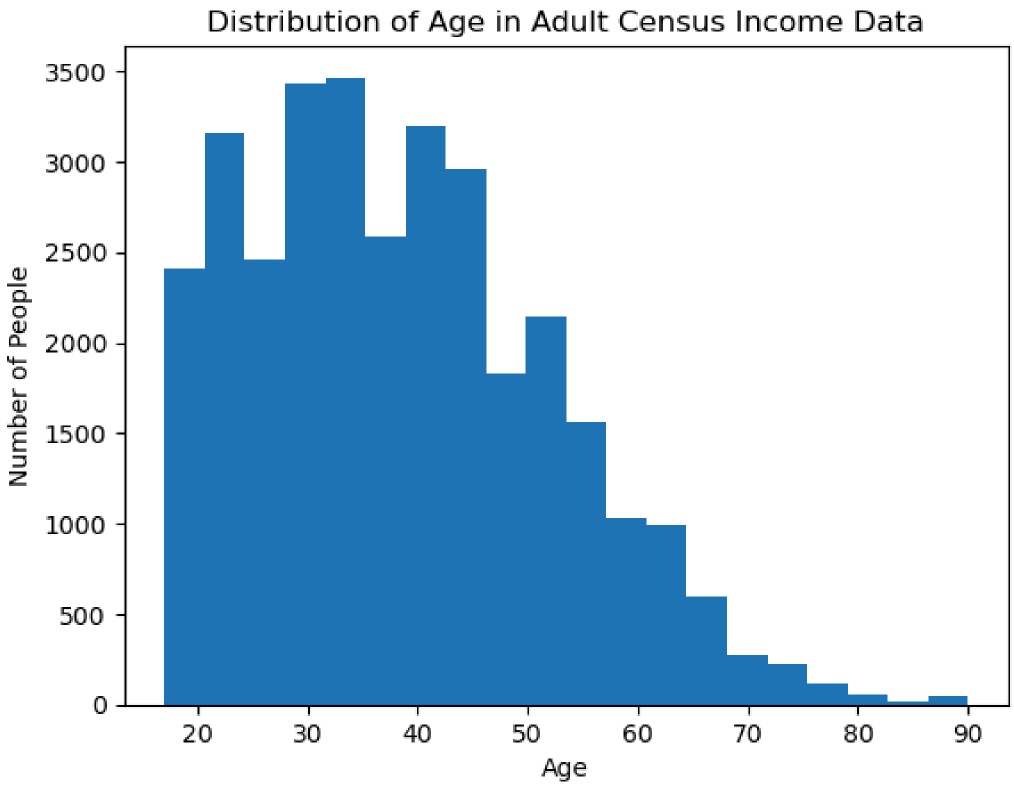
on.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.c
9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-
9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-
10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-
4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-
10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-

In [4]:
```
df.info()
```

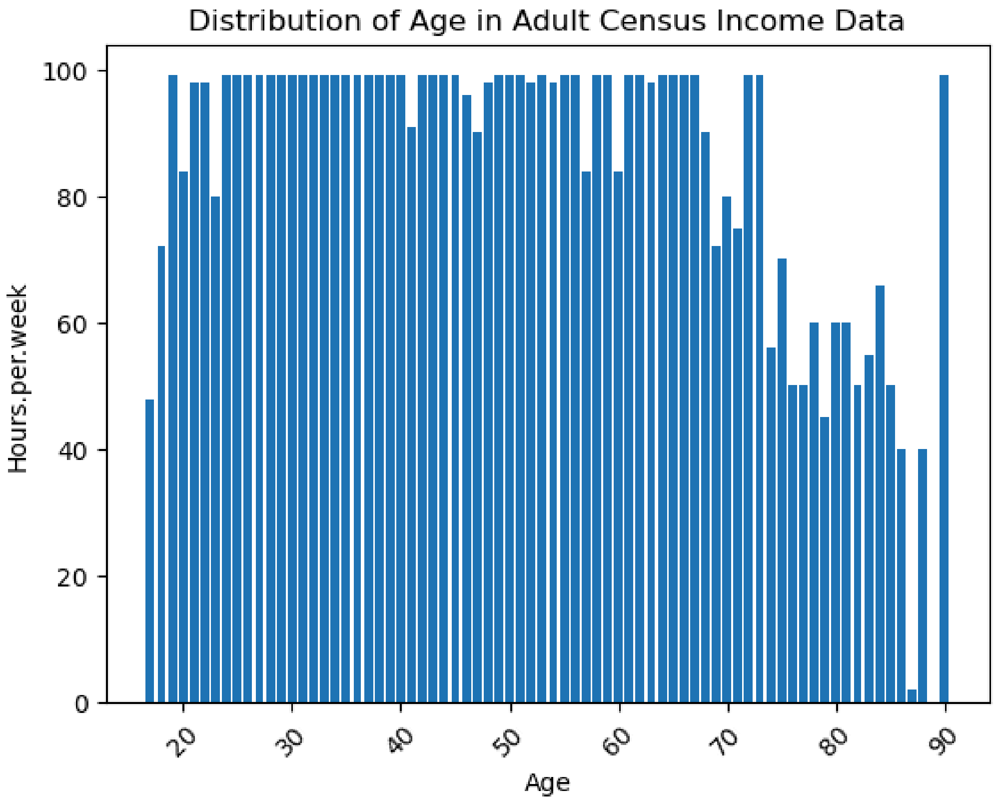
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64 
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64 
 11  capital.loss    32561 non-null  int64 
 12  hours.per.week  32561 non-null  int64 
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

In [5]:
```
ages = df["age"]

# Create the histogram
plt.hist(ages, bins=20)  # Adjust the number of bins as needed
plt.xlabel("Age")
plt.ylabel("Number of People")
plt.title("Distribution of Age in Adult Census Income Data")
plt.show()
```



In [6]:
```
# Create the bar chart
plt.bar(df['age'],df['hours.per.week'])
plt.xlabel("Age")
plt.ylabel("Hours.per.week")
plt.title("Distribution of Age in Adult Census Income Data")
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



In [ ]:

TASK-2

BUSINESS PROBLEM - Perform data cleaning and exploratory data analysis (EDA) on a dataset of your choice, such as the Titanic dataset from Kaggle. Explore the relationships between variables and identify patterns and trends in the data.

In [7]:
```
df=pd.read_csv(r"C:\Users\Kokat\Downloads\prodigy_tech\prodigy_tech\titanic\train.csv")
df.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [10]:
```
df.drop('Cabin', axis=1, inplace=True)
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Embarked         2
dtype: int64
```

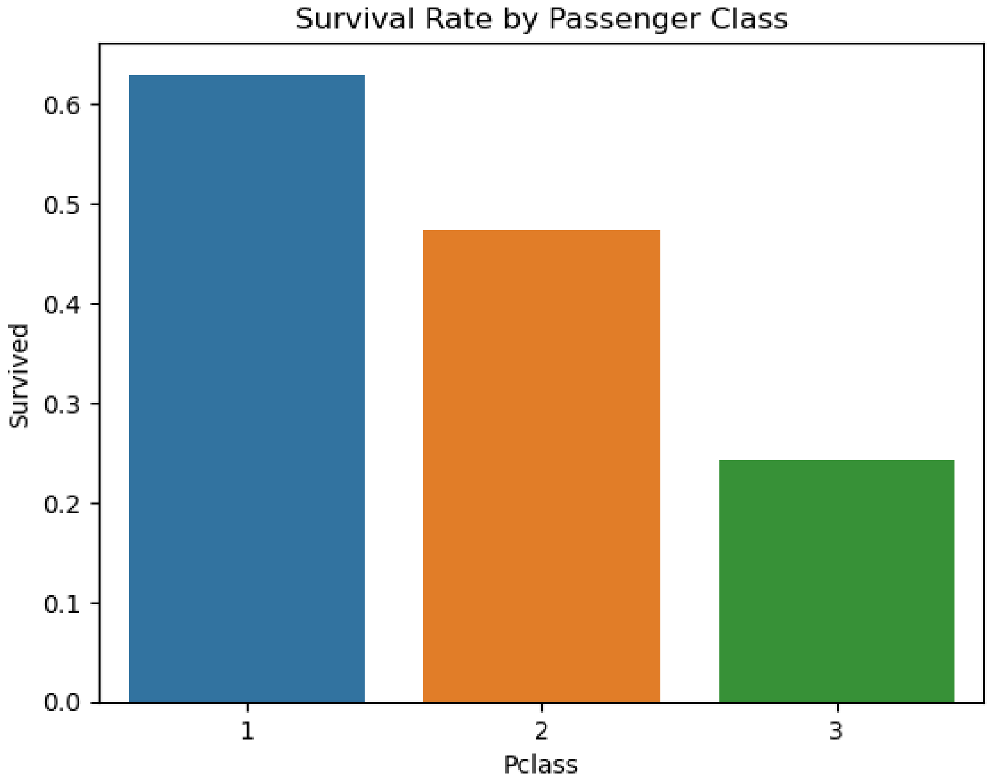
In [12]:
```
# Assuming you want to fill missing values in 'Age' with the mean value
mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)
# Assuming 'Embarked' is the categorical column
most_frequent_category = df['Embarked'].mode()[0]
df['Embarked'].fillna(most_frequent_category, inplace=True)
```

In [13]:
```
df.isnull().sum()
```

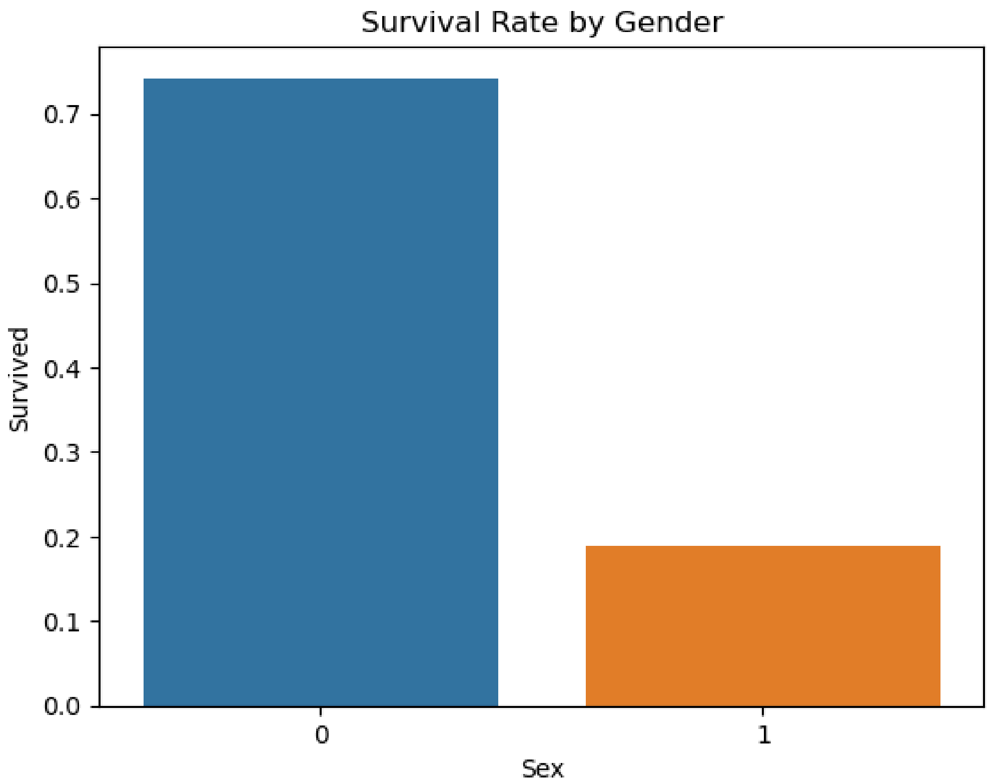
Out[13]:
```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Embarked       0
dtype: int64
```

In [14]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['Sex']=le.fit_transform(df['Sex'])
df['Embarked']=le.fit_transform(df['Embarked'])
```

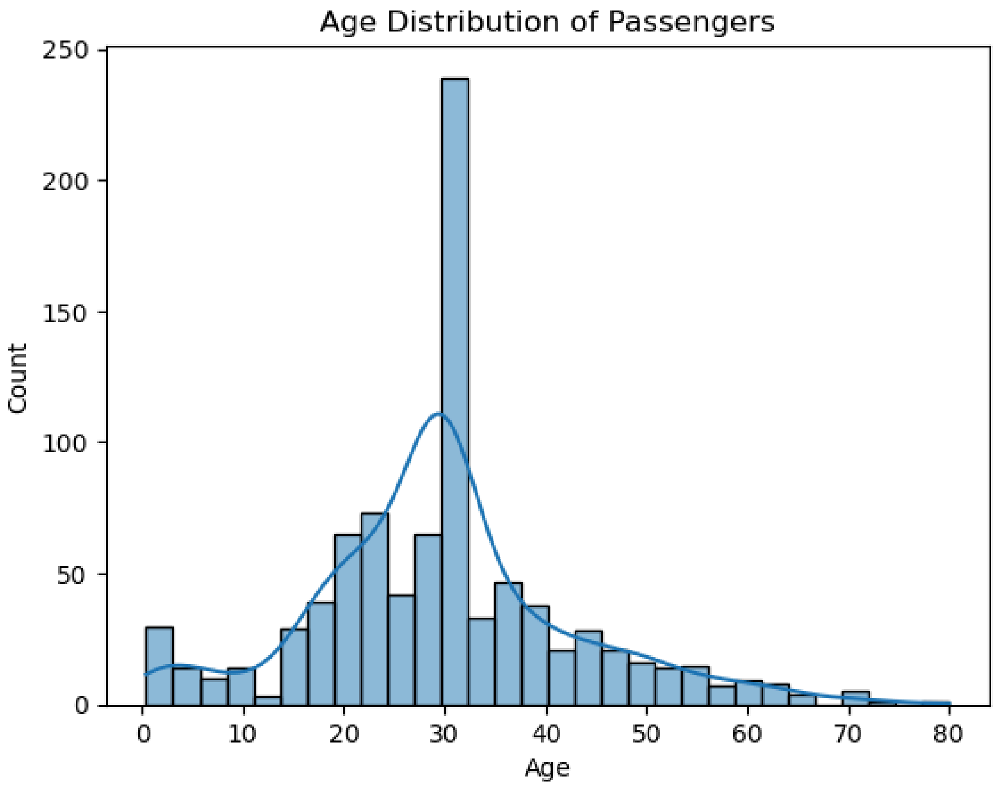
In [15]:
```
# Explore relationships between variables
# For example, visualize the survival rate based on passenger class
sns.barplot(x='Pclass', y='Survived', data=df, ci=None)
plt.title('Survival Rate by Passenger Class')
plt.show()
```



In [16]:
```
# Explore relationships between variables
# For example, visualize the survival rate based on gender
sns.barplot(x='Sex', y='Survived', data=df, ci=None)
plt.title('Survival Rate by Gender')
plt.show()
```



In [17]:
```
# Explore relationships between variables
# For example, visualize the age distribution of passengers
sns.histplot(df['Age'], bins=30, kde=True)
plt.title('Age Distribution of Passengers')
plt.show()
```



In [ ]:

TASK -3

BUSINESS PROBLEM - Build a decision tree classifier to predict whether a customer will purchase a product or service based on their demographic and behavioral data. Use a dataset such as the Bank Marketing dataset from the UCI Machine Learning Repository.

In [18]:
```
df=pd.read_csv(r"C:\Users\Kokat\Downloads\prodigy_tech\prodigy_tech\Bank Marketing\bank(t
df.head()
```

Out[18]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProdu
0	1	15634602	Hargrave	619	delhi	Female	42	2	0.00	
1	2	15647311	Hill	608	bangalore	Female	41	1	83807.86	
2	3	15619304	Onio	502	delhi	Female	42	8	159660.80	
3	4	15701354	Boni	699	delhi	Female	39	1	0.00	
4	5	15737888	Mitchell	850	bangalore	Female	43	2	125510.82	

In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [20]:
```
df.isnull().sum()
```

Out[20]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [21]:
```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Geography'] = label_encoder.fit_transform(df['Geography'])
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df.head()
```

Out[21]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProdu
0	1	15634602	Hargrave	619	1	0	42	2	0.00	
1	2	15647311	Hill	608	0	0	41	1	83807.86	
2	3	15619304	Onio	502	1	0	42	8	159660.80	
3	4	15701354	Boni	699	1	0	39	1	0.00	
4	5	15737888	Mitchell	850	0	0	43	2	125510.82	

In [22]:
```
X=df.iloc[:,3:13].values
y=df.iloc[:,-1].values
```

In [24]:
```
X
```

Out[24]:
```
array([[6.1900000e+02, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        1.0000000e+00, 1.0134888e+05],
       [6.0800000e+02, 0.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 1.1254258e+05],
       [5.0200000e+02, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 1.1393157e+05],
       ...,
       [7.0900000e+02, 1.0000000e+00, 0.0000000e+00, ..., 0.0000000e+00,
        1.0000000e+00, 4.2085580e+04],
       [7.7200000e+02, 2.0000000e+00, 1.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 9.2888520e+04],
       [7.9200000e+02, 1.0000000e+00, 0.0000000e+00, ..., 1.0000000e+00,
        0.0000000e+00, 3.8190780e+04]])
```

In [25]:
```
y
```

Out[25]:
```
array([1, 0, 1, ..., 1, 1, 0], dtype=int64)
```

In [26]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,random_state=0)
```

In [27]:
```
from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier()
classifier.fit(X_train,y_train)
```

Out[27]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [28]:
```
y_pred=classifier.predict(X_test)
```

In [29]:
```
y_pred
```

Out[29]:
```
array([0, 1, 0, ..., 0, 0, 1], dtype=int64)
```

In [30]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[1381  214]
 [ 168  237]]
```

In [31]:
```
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test,y_pred)
print(ac*100)
```

```
80.9
```

In [ ]:

TASK-4

BUSINESS PROBLEM -Analyze and visualize sentiment patterns in social media data to understand public opinion and attitudes towards specific topics or brands.

In [1]:

```
#Load the necessary libraries and datasets
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.pipeline import Pipeline
```

In [2]:

```
# Load the training and validation datasets without header
df_training = pd.read_csv('/kaggle/input/twitter-entity-sentiment-analysis/twitter_traini
df_validation = pd.read_csv('/kaggle/input/twitter-entity-sentiment-analysis/twitter_vali

# Merge the two datasets
df = pd.concat([df_training, df_validation], ignore_index=True)

# Filter the dataset by 'Tweet content' for LeagueOfLegends
df_league_of_legends = df[df['entity'].str.contains('LeagueOfLegends', case=False, na=Fal

# Drop rows with missing values in 'Tweet content'
df_league_of_legends = df_league_of_legends.dropna(subset=['Tweet content'])

# Shuffle the dataset
df_league_of_legends = shuffle(df_league_of_legends, random_state=42)

# Split the dataset into training and validation sets
X = df_league_of_legends['Tweet content']
y = df_league_of_legends['sentiment']

X_train, X_validation, y_train, y_validation = train_test_split(X, y, test_size=0.2, rand

# Display information about the resulting datasets
print("Training set size:", len(X_train))
print("Validation set size:", len(X_validation))
```

```
Training set size: 1931
Validation set size: 483
```

In [3]:
```
# Models
models = [
    MultinomialNB(),
    SVC(),
    RandomForestClassifier(),
    LogisticRegression(),
    GradientBoostingClassifier()
]

# Feature extraction methods
vectorizers = [
    ('TF-IDF', TfidfVectorizer()),
    ('Count Vectorizer', CountVectorizer())
]

# Example sentences
new_examples = [
    "I love playing League of Legends!",
    "This game is terrible. I hate it.",
    "Neutral tweet about League of Legends."
]

# Loop through models and vectorizers for training and evaluation
for model in models:
    for vectorizer_name, vectorizer in vectorizers:
        # Define a pipeline with feature engineering and a machine learning algorithm
        pipeline = Pipeline([
            ('vectorizer', vectorizer),
            ('classifier', model)
        ])

        # Train the model
        pipeline.fit(X_train, y_train)

        # Evaluate the model
        y_pred = pipeline.predict(X_validation)
        accuracy = accuracy_score(y_validation, y_pred)
        report = classification_report(y_validation, y_pred)

        # Display results
        print(f"\nModel: {model.__class__.__name__}, Vectorizer: {vectorizer_name}")
        print("Accuracy:", accuracy)
        print("Classification Report:\n", report)

        # Predict the sentiment of new examples
        predicted_sentiments = pipeline.predict(new_examples)
        for example, sentiment in zip(new_examples, predicted_sentiments):
            print(f"Example: '{example}' - Predicted Sentiment: {sentiment}")
        print("\n------------------------------------------------------------------------
```

```
Model: MultinomialNB, Vectorizer: TF-IDF
Accuracy: 0.8902691511387164
Classification Report:
              precision    recall  f1-score   support

  Irrelevant       1.00      0.52      0.68        66
    Negative       0.86      0.97      0.91       117
     Neutral       0.86      0.96      0.91       175
    Positive       0.95      0.92      0.93       125

    accuracy                           0.89       483
   macro avg       0.92      0.84      0.86       483
weighted avg       0.90      0.89      0.88       483

Example: 'I love playing League of Legends!' - Predicted Sentiment: Positive
Example: 'This game is terrible. I hate it.' - Predicted Sentiment: Negative
Example: 'Neutral tweet about League of Legends.' - Predicted Sentiment: Negative
```

In [ ]: