In [18]:
```
import numpy as np
import pandas as pd

# Load data into DataFrames
tweets = pd.read_csv(r"C:\Users\vkrat\Desktop\Social Marketing stock price Prediction\stock_tweets.csv")
yfinance = pd.read_csv(r"C:\Users\vkrat\Desktop\Social Marketing stock price Prediction\stock_yfinance_data.csv")

print('Both tweets and yfinance data files have been uploaded successfully!')
```

```
Both tweets and yfinance data files have been uploaded successfully!
```

In [23]:
```
# Display first few rows of each dataset
print("Stock Tweets Data:")
tweets.head()
```

```
Stock Tweets Data:
```

Out[23]:

	Date	Tweet	Stock Name	Company Name
0	2022-09-29 23:41:16+00:00	Mainstream media has done an amazing job at br...	TSLA	Tesla, Inc.
1	2022-09-29 23:24:43+00:00	Tesla delivery estimates are at around 364k fr...	TSLA	Tesla, Inc.
2	2022-09-29 23:18:08+00:00	3/ Even if I include 63.0M unvested RSUs as of...	TSLA	Tesla, Inc.
3	2022-09-29 22:40:07+00:00	@RealDanODowd @WholeMarsBlog @Tesla Hahaha why...	TSLA	Tesla, Inc.
4	2022-09-29 22:27:05+00:00	@RealDanODowd @Tesla Stop trying to kill kids,...	TSLA	Tesla, Inc.

In [24]:
```
print("Stock Yfinance Data:")
yfinance.head()
```

```
Stock Yfinance Data:
```

Out[24]:

	Date	Open	High	Low	Close	Adj Close	Volume	Stock Name
0	2021-09-30	260.333344	263.043335	258.333344	258.493347	258.493347	53868000	TSLA
1	2021-10-01	259.466675	260.260010	254.529999	258.406677	258.406677	51094200	TSLA
2	2021-10-04	265.500000	268.989990	258.706665	260.510010	260.510010	91449900	TSLA
3	2021-10-05	261.600006	265.769989	258.066681	260.196655	260.196655	55297800	TSLA
4	2021-10-06	258.733337	262.220001	257.739990	260.916656	260.916656	43898400	TSLA

In [25]:
```
# Display the column names of both datasets
print("Column names in Stock Tweets Data:")
tweets.columns.tolist()
```

```
Column names in Stock Tweets Data:
```

Out[25]:
```
['Date', 'Tweet', 'Stock Name', 'Company Name']
```

In [26]:
```
# Display the column names of both datasets
print("Column names in Stock yfinance Data:")
yfinance.columns.tolist()
```

```
Column names in Stock yfinance Data:
```

Out[26]:
```
['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume', 'Stock Name']
```

In [27]:
```
# Check for missing values
print("Missing values in tweets data:")
print(tweets.isnull().sum())
print("Missing values in yfinance data:")
print(yfinance.isnull().sum())
```

```
Missing values in tweets data:
Date            0
Tweet           0
Stock Name      0
Company Name    0
dtype: int64
Missing values in yfinance data:
Date          0
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
Stock Name    0
dtype: int64
```

In [32]:
```
# Convert the 'Date' column to datetime without timezone
from textblob import TextBlob
tweets['Date'] = pd.to_datetime(tweets['Date']).dt.tz_localize(None)
yfinance['Date'] = pd.to_datetime(yfinance['Date']).dt.tz_localize(None)
tweets['Polarity'] = tweets['Tweet'].apply(lambda tweet: TextBlob(tweet).sentiment.polarity)
tweets['Sentiment'] = tweets['Polarity'].apply(lambda x: 'Positive' if x > 0 else ('Negative' if x < 0 else 'Neutral'))
# Now try merging the two datasets on 'Date'
merged_df = pd.merge(tweets, yfinance, on='Date', how='inner')
# Check the merged data
print("Merged Data:")
merged_df.head()
```

```
Merged Data:
```

Out[32]:

	Date	Tweet	Stock Name_x	Company Name	Polarity	Sentiment	Open	High	Low	Close	Adj Close	Volume	Stock Name_y
0	2022-08-30	@fraggelcurris @latestinspace @elonmusk @Tesla...	TSLA	Tesla, Inc.	0.3125	Positive	287.869995	288.480011	272.649994	277.700012	277.700012	50541800	TSLA
1	2022-08-30	@fraggelcurris @latestinspace @elonmusk @Tesla...	TSLA	Tesla, Inc.	0.3125	Positive	266.670013	267.049988	260.660004	262.970001	262.230988	22767100	MSFT
2	2022-08-30	@fraggelcurris @latestinspace @elonmusk @Tesla...	TSLA	Tesla, Inc.	0.3125	Positive	142.410004	142.410004	139.910004	140.179993	139.192154	5203700	PG
3	2022-08-30	@fraggelcurris @latestinspace @elonmusk @Tesla...	TSLA	Tesla, Inc.	0.3125	Positive	160.350006	161.660004	155.910004	157.160004	157.160004	19567900	META
4	2022-08-30	@fraggelcurris @latestinspace @elonmusk @Tesla...	TSLA	Tesla, Inc.	0.3125	Positive	131.250000	132.070007	126.849998	128.729996	128.729996	49203000	AMZN

In [33]:
```
print(merged_df.columns)
```

```
Index(['Date', 'Tweet', 'Stock Name_x', 'Company Name', 'Polarity',
       'Sentiment', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume',
       'Stock Name_y'],
      dtype='object')
```

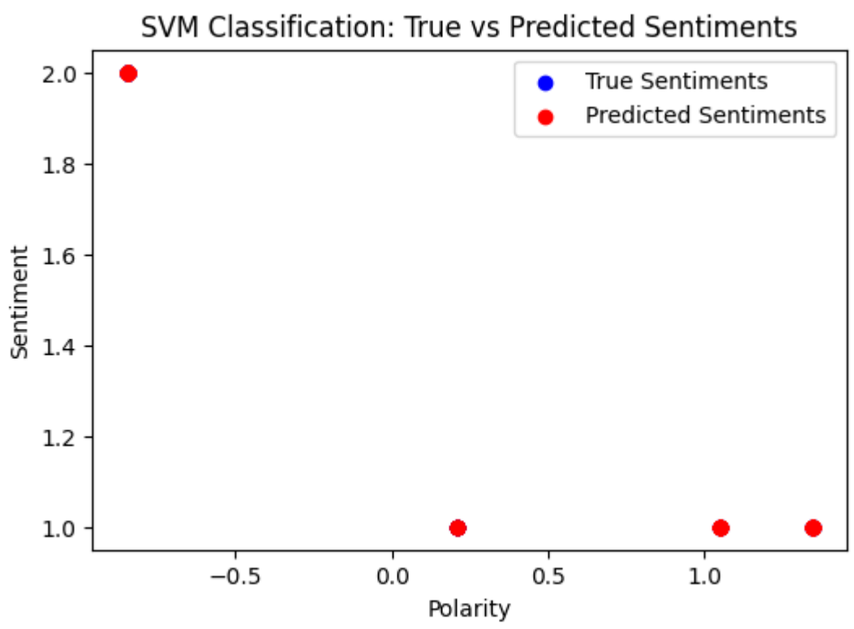
In [34]:
```
# Now check if the 'Polarity' column exists
if 'Polarity' in merged_df.columns:
 print("Polarity column is available.")
else:
 print("Polarity column is not found in the merged data.")
# Now check if the 'Sentiment' column exists
if 'Sentiment' in merged_df.columns:
 print("Sentiment column is available.")
else:
 print("Sentiment column is not found in the merged data.")
```

```
Polarity column is available.
Sentiment column is available.
```

In [36]:
```python
from textblob import TextBlob
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, mean_squared_error
X = merged_df[['Polarity']] # Using 'Polarity' as the feature
y = merged_df['Sentiment'].map({'Positive': 1, 'Negative': 0, 'Neutral': 2}) # Mapping sentiment to integers
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Standardize the features (important for SVM)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Train the SVM classifier
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
# Make predictions
y_pred = svm.predict(X_test)
# Evaluate the model
print(f"Accuracy of SVM: {accuracy_score(y_test, y_pred)}")
```

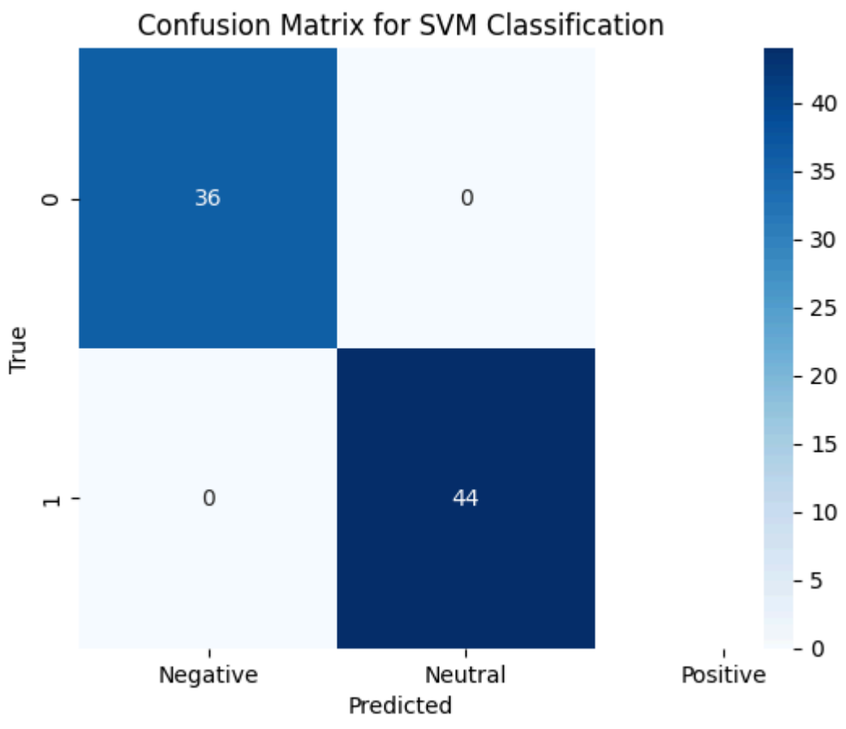
```
Accuracy of SVM: 1.0
```

In [37]:
```python
import matplotlib.pyplot as plt
# Step 8: Visualize SVM Results
plt.figure(figsize=(6, 4))
plt.scatter(X_test, y_test, color='blue', label='True Sentiments')
plt.scatter(X_test, y_pred, color='red', label='Predicted Sentiments')
plt.title('SVM Classification: True vs Predicted Sentiments')
plt.xlabel('Polarity')
plt.ylabel('Sentiment')
plt.legend()
plt.show()
```



In [42]:
```python
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
print("\nClassification Report for SVM:")
print(classification_report(y_test, y_pred))
# Plot the confusion matrix using Seaborn
print("\nConfusion Matrix for SVM:")
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Neutral', 'Positive'])
plt.title('Confusion Matrix for SVM Classification')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
Classification Report for SVM:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        36
           2       1.00      1.00      1.00        44

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80


Confusion Matrix for SVM:
```

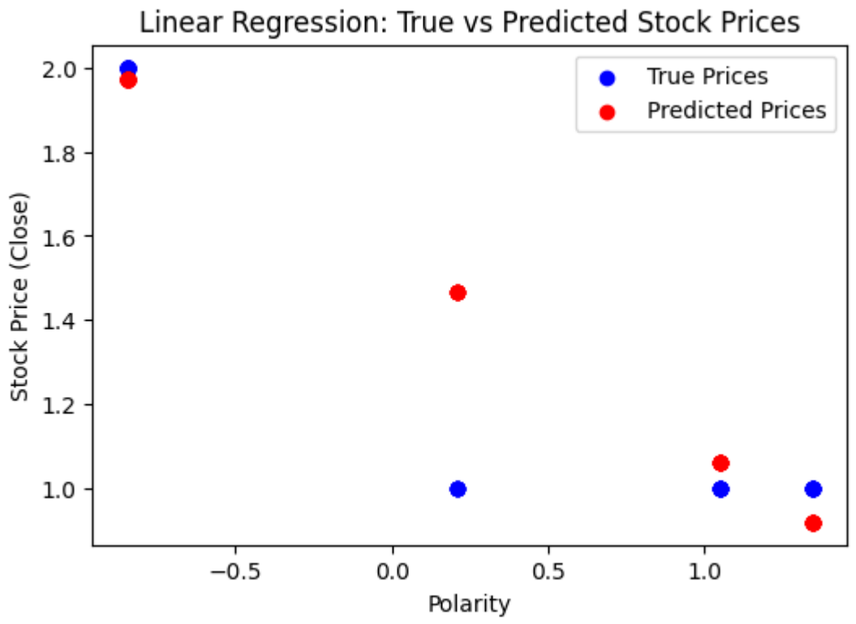


In [43]:
```python
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
# Step 9: Standardize the features for Linear Regression (stock price prediction)
scaler_regression = StandardScaler()
X_train = scaler_regression.fit_transform(X_train)
X_test= scaler_regression.transform(X_test)
# Step 10: Train and Evaluate Linear Regression Model (for stock price prediction)
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)
# Make predictions for regression
y_pred = linear_regressor.predict(X_test)
# Step 11: Evaluate Linear Regression performance (Mean Squared Error)
```

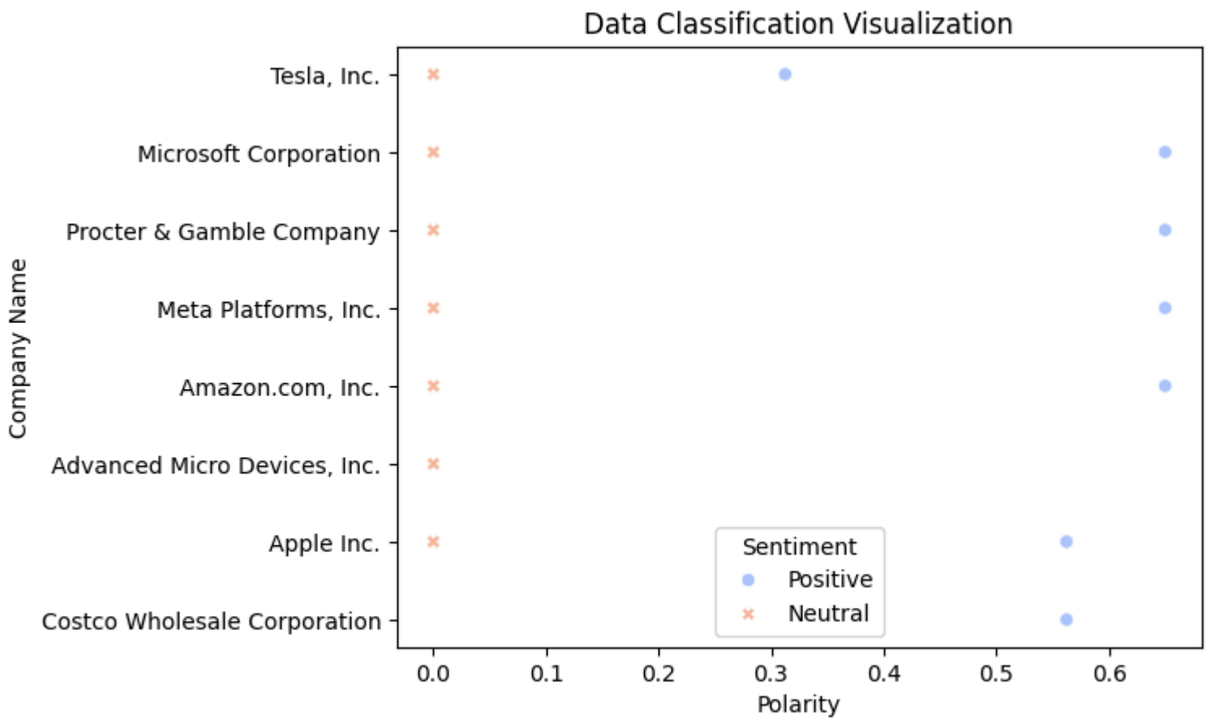
```
mse = mean_squared_error(y_test, y_pred)
print(f"\nMean Squared Error of Linear Regression: {mse}")
```

```
Mean Squared Error of Linear Regression: 0.016104298195439782
```

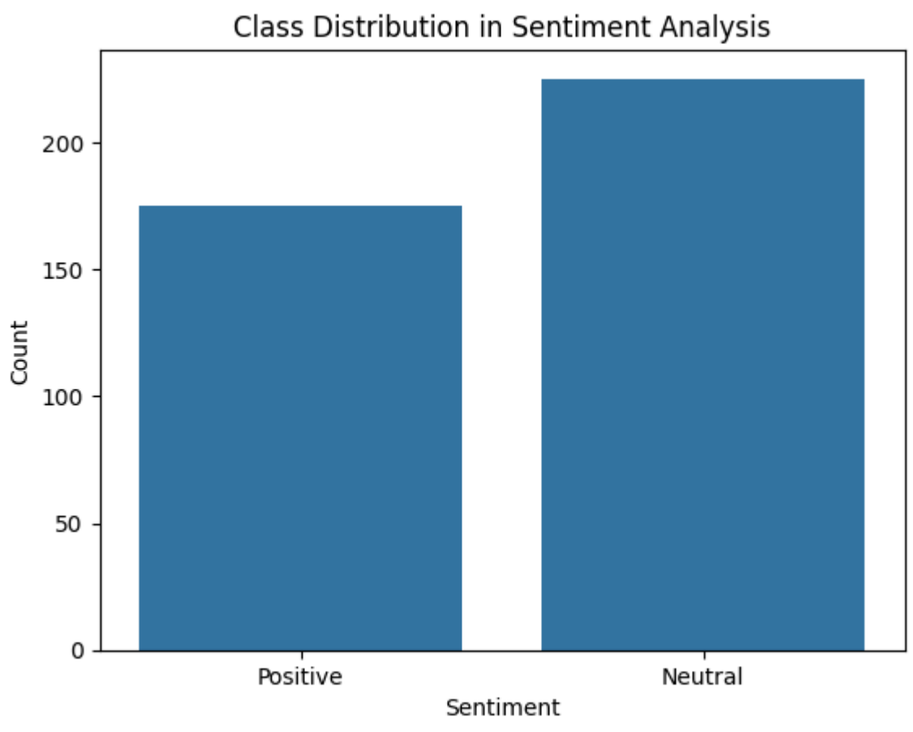
In [44]:
```
# Step 12: Visualize Linear Regression Results
plt.figure(figsize=(6, 4))
plt.scatter(X_test, y_test, color='blue', label='True Prices')
plt.scatter(X_test, y_pred, color='red', label='Predicted Prices')
plt.title('Linear Regression: True vs Predicted Stock Prices')
plt.xlabel('Polarity')
plt.ylabel('Stock Price (Close)')
plt.legend()
plt.show()
```



In [45]:
```
import matplotlib.pyplot as plt
import seaborn as sns
# Assuming your data contains features 'Polarity' and 'Company Name'
sns.scatterplot(data=merged_df, x='Polarity', y='Company Name', hue='Sentiment', palette='coolwarm', style='Sentiment')
plt.title('Data Classification Visualization')
plt.xlabel('Polarity')
plt.ylabel('Company Name')
plt.legend(title='Sentiment')
plt.show()
```

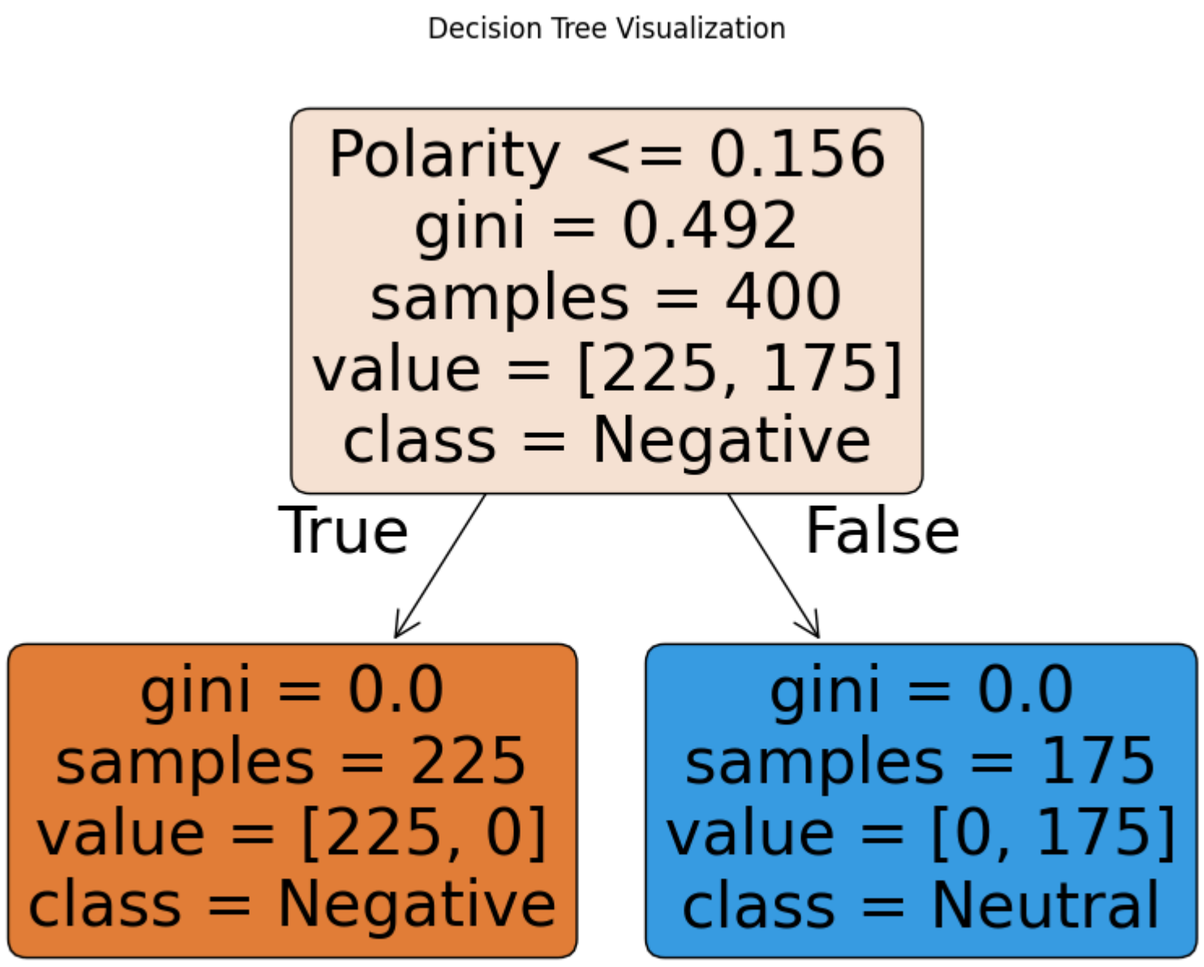


In [46]:
```
'''
 Visualize Data Classification
'''
# Plotting the distribution of the classes
sns.countplot(x='Sentiment', data=merged_df)
plt.title('Class Distribution in Sentiment Analysis')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

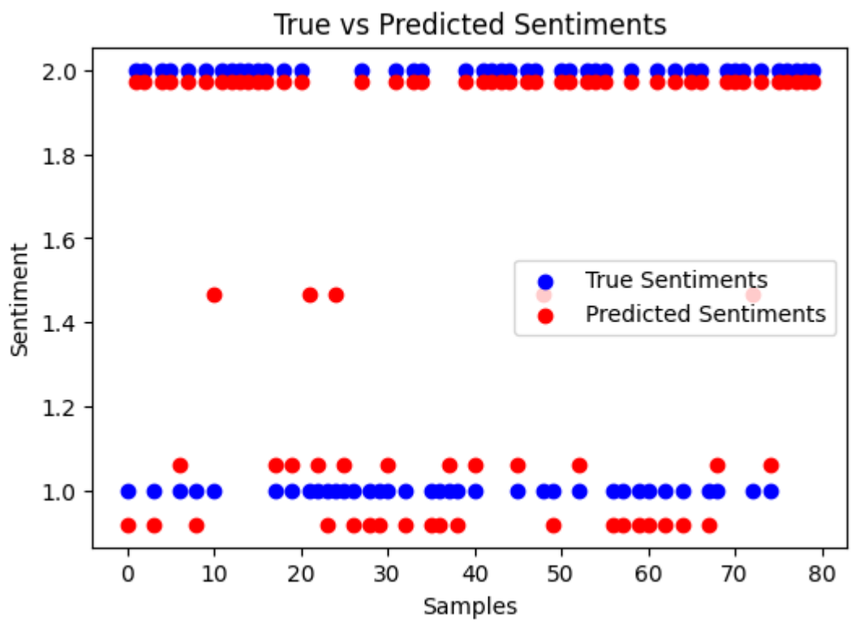


In [62]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt

# Example: Train the Decision Tree model
X = merged_df[['Polarity', 'High']]  # Assuming these are the features
y = merged_df['Sentiment']  # Target variable

# Train the decision tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X, y)

# Visualizing the tree
```

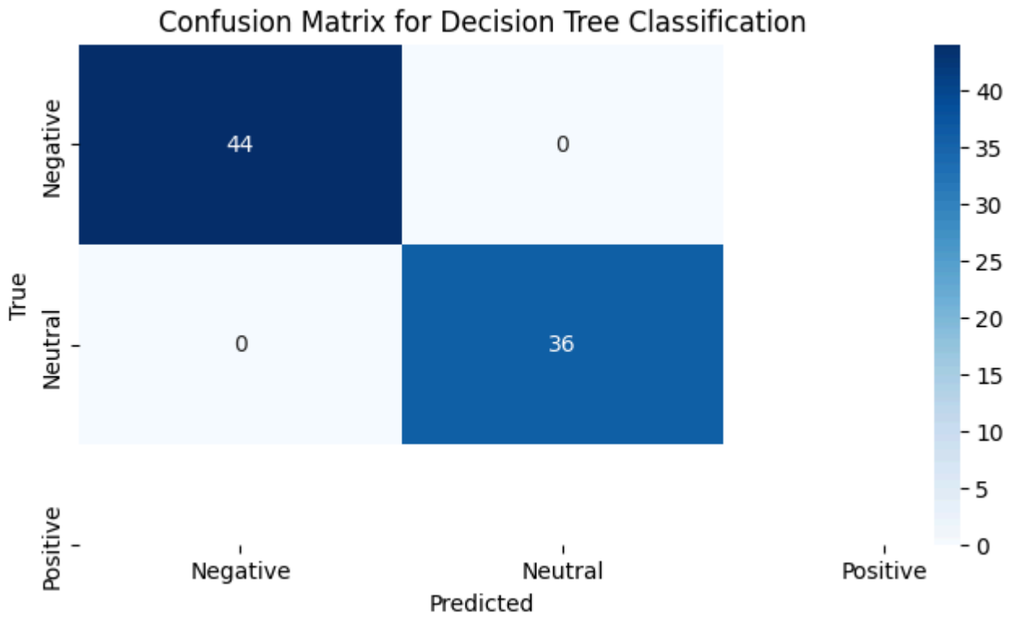
```
plt.figure(figsize=(10, 8))
plot_tree(
    dt,
    filled=True,
    feature_names=['Polarity', 'High'],
    class_names=['Negative', 'Neutral', 'Positive'],
    rounded=True
)
plt.title('Decision Tree Visualization')  # Title was misplaced
plt.show()
```



In [64]:
```
# Assuming y_test_classification and y_pred_classification are available
plt.figure(figsize=(6, 4))
plt.scatter(range(len(y_test)), y_test, color='blue', label='True Sentiments')
plt.scatter(range(len(y_pred)), y_pred, color='red', label='Predicted Sentiments')
plt.title('True vs Predicted Sentiments')
plt.xlabel('Samples')
plt.ylabel('Sentiment')
plt.legend()
plt.show()
```



In [73]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier

# Assuming 'X_train', 'X_test', 'y_train', 'y_test' are already defined
# Train the decision tree classifier
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)

# Make predictions on the test set
y_pred = dt.predict(X_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix
plt.figure(figsize=(8, 4))
sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=['Negative', 'Neutral', 'Positive'],
    yticklabels=['Negative', 'Neutral', 'Positive']
)
plt.title('Confusion Matrix for Decision Tree Classification')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```


Confusion Matrix for Decision Tree Classification
Negative
Neutral
Positive
True
44
0
0
36
Negative
Neutral
Positive
Predicted
40
35
30
25
20
15
10
5
0