In [84]:

```
import pandas as pd

senate_df = pd.read_csv('senate.csv', encoding='ISO-8859-1')
pres_df = pd.read_csv('pres.csv', encoding='ISO-8859-1')
house_df = pd.read_csv('house.csv', encoding='ISO-8859-1')
data_2024 = pd.read_csv('candidate_summary_2024.csv', encoding='ISO-8859-1')
candidate_summaries = {}  # Create a dictionary to store the candidate summary DataFrames
for year in range(2008, 2024, 2):
    filename = f'candidate_summary_{year}.csv'
    candidate_summaries[year] = pd.read_csv(filename, encoding='ISO-8859-1')
```

In [85]:

```
combined_df = pd.concat([senate_df, pres_df, house_df], ignore_index=True)
```

In [86]:

```
print(combined_df.columns)
print(combined_df.nunique())
```

```
Index(['year', 'state', 'office', 'candidate_votes', 'total_votes', 'party'], dtype='object')
year                  25
state                 51
office                 3
candidate_votes    30636
total_votes        11421
party                  3
dtype: int64
```

In [87]:

```
# Read each CSV file into a DataFrame
candidate_summary_2008_df = pd.read_csv("candidate_summary_2008.csv")
candidate_summary_2010_df = pd.read_csv("candidate_summary_2010.csv")
candidate_summary_2012_df = pd.read_csv("candidate_summary_2012.csv")
candidate_summary_2014_df = pd.read_csv("candidate_summary_2014.csv")
candidate_summary_2016_df = pd.read_csv("candidate_summary_2016.csv")
candidate_summary_2018_df = pd.read_csv("candidate_summary_2018.csv")
candidate_summary_2020_df = pd.read_csv("candidate_summary_2020.csv")
candidate_summary_2022_df = pd.read_csv("candidate_summary_2022.csv")
candidate_summary_2024_df = pd.read_csv("candidate_summary_2024.csv")

# Print the first few rows of each DataFrame to verify they were read correctly
print(candidate_summary_2008_df.head())
print(candidate_summary_2010_df.head())
print(candidate_summary_2012_df.head())
print(candidate_summary_2014_df.head())
print(candidate_summary_2016_df.head())
print(candidate_summary_2018_df.head())
print(candidate_summary_2020_df.head())
print(candidate_summary_2022_df.head())
print(candidate_summary_2024_df.head())
```

```
   year        office state_init party Cand_Incumbent_Challenger_Open_Seat  \
0  2008  US PRESIDENT         US   DEM                                OPEN
1  2008      US HOUSE         FL   DEM                          CHALLENGER
2  2008      US HOUSE         FL   REP                           INCUMBENT
3  2008      US HOUSE         FL   DEM                          CHALLENGER
4  2008      US HOUSE         FL   DEM                          CHALLENGER

   Total_Receipt  Total_Disbursement  Cash_On_Hand_COP  \
0   7.786430e+08        7.603702e+08       18272367.39
1   0.000000e+00        2.760000e+02              0.00
2   8.064492e+05        7.897812e+05        2272965.45
3   2.497569e+05        2.497569e+05              0.00
4   3.055300e+04        2.967000e+04              0.00

   Debt_Owed_By_Committee Coverage_End_Date  ... Individual_Refund  \
0                434954.4        12/31/2008  ...         5744310.2
1                     0.0         9/30/2008  ...               0.0
2                     0.0        12/31/2008  ...             300.0
3                     0.0        12/31/2008  ...              25.0
4                 16245.0         9/30/2008  ...               0.0
```

In [88]:

```
# Concatenate all the DataFrames into a single DataFrame
combined_df1 = pd.concat([candidate_summary_2008_df, candidate_summary_2010_df, candidate_summary_2012_df,
                          candidate_summary_2014_df, candidate_summary_2016_df, candidate_summary_2018_df,
                          candidate_summary_2020_df, candidate_summary_2022_df, candidate_summary_2024_df])

# Print the first few rows of the combined DataFrame to verify it was created correctly
print(combined_df1.head())
```

```
4  2008      US HOUSE          FL    DEM                         CHALLENGER

   Total_Receipt  Total_Disbursement  Cash_On_Hand_COP  \
0   7.786430e+08        7.603702e+08       18272367.39   
1   0.000000e+00        2.760000e+02              0.00   
2   8.064492e+05        7.897812e+05        2272965.45   
3   2.497569e+05        2.497569e+05              0.00   
4   3.055300e+04        2.967000e+04              0.00   

   Debt_Owed_By_Committee Coverage_End_Date  ... Party_Committee_Refund  \
0                434954.4        12/31/2008  ...                  300.0   
1                     0.0         9/30/2008  ...                    0.0   
2                     0.0        12/31/2008  ...                    0.0   
3                     0.0        12/31/2008  ...                    0.0   
4                 16245.0         9/30/2008  ...                    0.0   

  Other_Committee_Refund Total_Contribution_Refund Other_Disbursements  \
0                11345.0                 5755955.2         47945662.98   
1                    0.0                       0.0                0.00   
2                    0.0                     300.0           267040.00
```

In [89]:

```
print(combined_df1.columns)
print(combined_df1.nunique())
```

```
Index(['year', 'office', 'state_init', 'party',
       'Cand_Incumbent_Challenger_Open_Seat', 'Total_Receipt',
       'Total_Disbursement', 'Cash_On_Hand_COP', 'Debt_Owed_By_Committee',
       'Coverage_End_Date', 'Cand_Street_1', 'Cand_Street_2', 'Cand_City',
       'Cand_State', 'Cand_Zip', 'Individual_Itemized_Contribution',
       'Individual_Unitemized_Contribution', 'Individual_Contribution',
       'Other_Committee_Contribution', 'Party_Committee_Contribution',
       'Cand_Contribution', 'Total_Contribution',
       'Transfer_From_Other_Auth_Committee', 'Cand_Loan', 'Other_Loan',
       'Total_Loan', 'Offsets_To_Operating_Expenditure',
       'Offsets_To_Fundraising', 'Offsets_To_Leagal_Accounting',
       'Other_Receipts', 'Operating_Expenditure',
       'Exempt_Legal_Accounting_Disbursement', 'Fundraising_Disbursement',
       'Transfer_To_Other_Auth_Committee', 'Cand_Loan_Repayment',
       'Other_Loan_Repayment', 'Total_Loan_Repayment', 'Individual_Refund',
       'Party_Committee_Refund', 'Other_Committee_Refund',
       'Total_Contribution_Refund', 'Other_Disbursements', 'Net_Contribution',
       'Net_Operating_Expenditure', 'Cash_On_Hand_BOP',
       'Debt_Owe_To_Committee', 'Coverage_Start_Date', 'state'],
      dtype='object')
year                                        9
office                                      3
state_init                                 57
party                                       3
Cand_Incumbent_Challenger_Open_Seat         3
Total_Receipt                           17053
Total_Disbursement                      17348
Cash_On_Hand_COP                        12532
Debt_Owed_By_Committee                   5118
Coverage_End_Date                        1816
Cand_Street_1                           26403
Cand_Street_2                            2080
Cand_City                                6578
Cand_State                                 59
Cand_Zip                                12620
Individual_Itemized_Contribution        13822
Individual_Unitemized_Contribution      13964
Individual_Contribution                 15709
Other_Committee_Contribution             7097
Party_Committee_Contribution             2131
Cand_Contribution                        4926
Total_Contribution                      16570
Transfer_From_Other_Auth_Committee       2909
Cand_Loan                                3498
Other_Loan                                384
Total_Loan                               3589
Offsets_To_Operating_Expenditure         7262
Offsets_To_Fundraising                     12
Offsets_To_Leagal_Accounting                8
Other_Receipts                           4762
Operating_Expenditure                   17237
Exempt_Legal_Accounting_Disbursement       21
Fundraising_Disbursement                   63
Transfer_To_Other_Auth_Committee         1079
```

```
Cand_Loan_Repayment                  2447
Other_Loan_Repayment                  287
Total_Loan_Repayment                 2556
Individual_Refund                    5267
Party_Committee_Refund                144
Other_Committee_Refund                905
Total_Contribution_Refund            5543
Other_Disbursements                  6098
Net_Contribution                    16402
Net_Operating_Expenditure           17025
Cash_On_Hand_BOP                     6145
Debt_Owe_To_Committee                 166
Coverage_Start_Date                  2082
state                                  57
dtype: int64
```

In [90]:

```python
# Select only the common columns from combined_df1
combined_df1_common = combined_df1[['year', 'state_init', 'office', 'party', 'Total_Receipt']].copy()

# Rename the columns in combined_df1_common to match the column names in combined_df
combined_df1_common.columns = ['year', 'state', 'office', 'party', 'candidate_votes']

# Merge the two DataFrames
merged_df = pd.concat([combined_df, combined_df1_common], ignore_index=True)
```

In [91]:

```python
import us

# Define a function to convert full state names to their abbreviations
def state_name_to_abbreviation(state_name):
    state = us.states.lookup(state_name)
    return state.abbr if state else state_name

# Apply the function to the 'state' column in combined_df
combined_df['state'] = combined_df['state'].apply(state_name_to_abbreviation)
```

In [92]:

```python
data_2024['state'] = data_2024['state'].apply(state_name_to_abbreviation)
```

In [93]:

```python
combined_df.head()
```

Out[93]:

	year	state	office	candidate_votes	total_votes	party
0	1976	AZ	US SENATE	321236	741210	REP
1	1976	AZ	US SENATE	1565	741210	OTHER
2	1976	AZ	US SENATE	400334	741210	DEM
3	1976	AZ	US SENATE	7310	741210	OTHER
4	1976	AZ	US SENATE	10765	741210	OTHER

In [94]:

```
# Rename the 'state_init' column in combined_df1 to 'state'
combined_df1 = combined_df1.rename(columns={'state_init': 'state'})
```

In [95]:

```
# Reset the index in both DataFrames to ensure they have unique index values
combined_df = combined_df.reset_index(drop=True)
combined_df1 = combined_df1.reset_index(drop=True)

# Perform a left outer join using the index as the key
merged_df = combined_df.merge(combined_df1, left_index=True, right_index=True, how='left', suffixes=('', '_y'))
```

In [96]:

```
merged_df.head()
```

Out[96]:

	year	state	office	candidate_votes	total_votes	party	year_y	office_y	state_y	party_y	...	Party_Committee_Refund	Other_Committee_Refund	Total_Contribution_Refund	Other_Disbursements	Net_Contribution	Net_Ope
0	1976	AZ	US SENATE	321236	741210	REP	2008.0	US PRESIDENT	US	DEM	...	300.0	11345.0	5755955.2	47945662.98	4.306975e+09	
1	1976	AZ	US SENATE	1565	741210	OTHER	2008.0	US HOUSE	FL	DEM	...	0.0	0.0	0.0	0.00	0.000000e+00	
2	1976	AZ	US SENATE	400334	741210	DEM	2008.0	US HOUSE	FL	REP	...	0.0	0.0	300.0	267040.00	5.845734e+05	
3	1976	AZ	US SENATE	7310	741210	OTHER	2008.0	US HOUSE	FL	DEM	...	0.0	0.0	25.0	0.00	1.287319e+05	
4	1976	AZ	US SENATE	10765	741210	OTHER	2008.0	US HOUSE	FL	DEM	...	0.0	0.0	0.0	1013.00	1.339800e+04	

5 rows × 54 columns

In [97]:

```
# Display summary statistics
print(merged_df.describe())
```

```
                year  candidate_votes   total_votes        year_y  \
count  40368.000000     4.036800e+04  4.036800e+04  33200.000000   
mean    1999.700456     1.236241e+05  6.238114e+05   2016.450422   
std       13.663653     3.656838e+05  1.318696e+06      4.889096   
min     1976.000000    -1.000000e+00 -1.000000e+00   2008.000000   
25%     1988.000000     3.736750e+03  1.733390e+05   2012.000000   
50%     2000.000000     5.216100e+04  2.289855e+05   2016.000000   
75%     2012.000000     1.181552e+05  3.234108e+05   2020.000000   
max     2022.000000     1.111025e+07  1.750088e+07   2024.000000   

       Total_Receipt  Total_Disbursement  Cash_On_Hand_COP  \
count   3.320000e+04        3.320000e+04      3.305600e+04   
mean    8.509738e+05        8.209538e+05      1.069792e+05   
std     1.309545e+07        1.301504e+07      6.741723e+05   
min     0.000000e+00       -1.632000e+03     -4.182223e+05   
25%     0.000000e+00        0.000000e+00      0.000000e+00   
50%     2.860000e+03        2.531725e+03      0.000000e+00   
75%     1.989130e+05        1.823867e+05      2.190258e+03   
max     1.124593e+09        1.121170e+09      2.980086e+07   
```

In [98]:

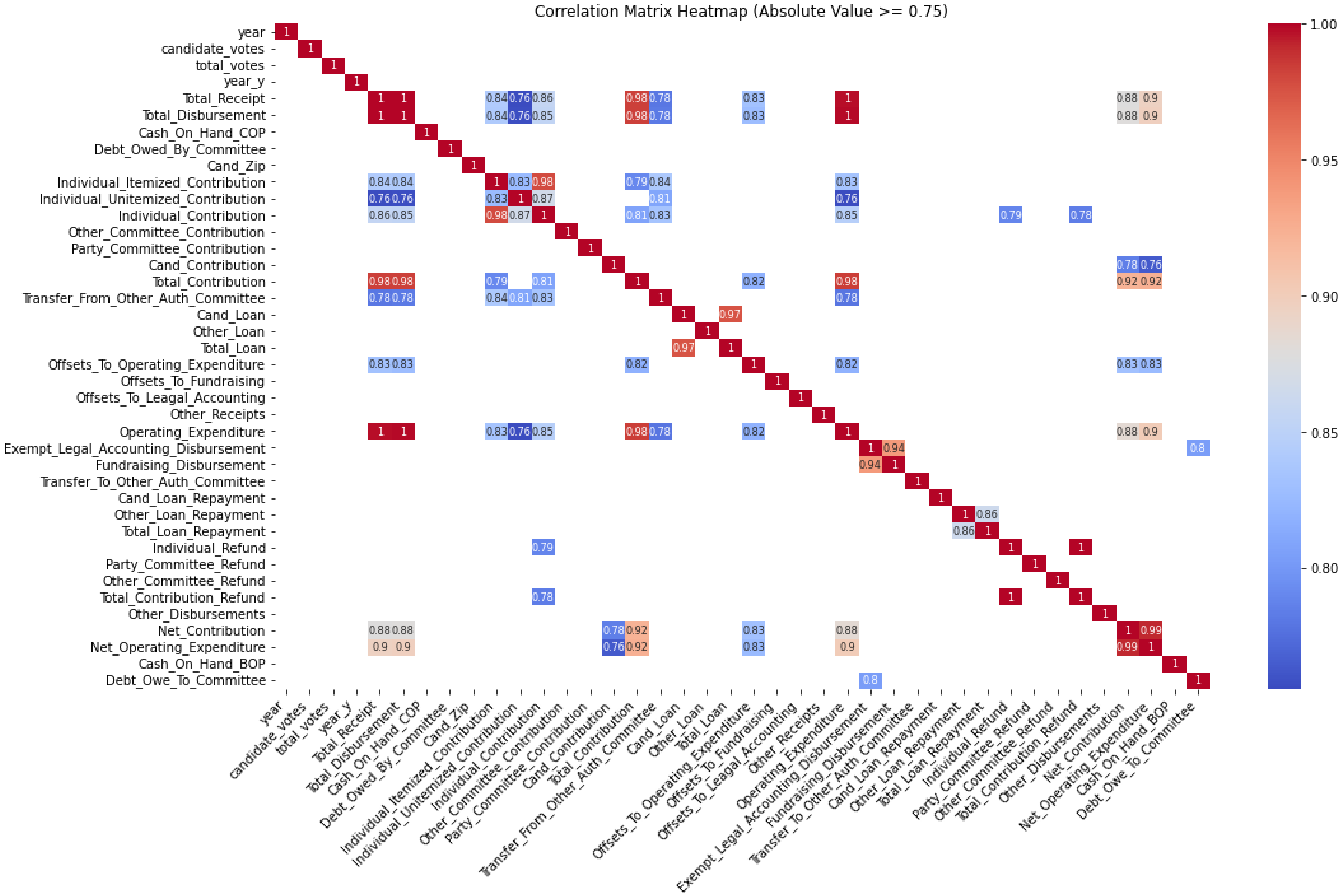
```
# Check for missing values
print(merged_df.isnull().sum())
```

```
year                                        0
state                                       0
office                                      0
candidate_votes                             0
total_votes                                 0
party                                       0
year_y                                   7168
office_y                                 7168
state_y                                  9867
party_y                                  7168
Cand_Incumbent_Challenger_Open_Seat      7357
Total_Receipt                            7168
Total_Disbursement                       7168
Cash_On_Hand_COP                         7312
Debt_Owed_By_Committee                   7675
Coverage_End_Date                       22132
Cand_Street_1                            7582
Cand_Street_2                           37482
Cand_City                                7190
Cand_State                               7539
Cand_Zip                                 7595
Individual_Itemized_Contribution         7168
Individual_Unitemized_Contribution       7168
Individual_Contribution                  7168
Other_Committee_Contribution             7168
Party_Committee_Contribution             7168
Cand_Contribution                        7168
Total_Contribution                       7168
Transfer_From_Other_Auth_Committee       7168
Cand_Loan                                7168
Other_Loan                               7168
Total_Loan                               7168
Offsets_To_Operating_Expenditure         7168
Offsets_To_Fundraising                   7168
Offsets_To_Leagal_Accounting             7168
Other_Receipts                           7168
Operating_Expenditure                    7168
Exempt_Legal_Accounting_Disbursement     7168
Fundraising_Disbursement                 7168
Transfer_To_Other_Auth_Committee         7168
Cand_Loan_Repayment                      7168
Other_Loan_Repayment                     7168
Total_Loan_Repayment                     7168
Individual_Refund                        7168
Party_Committee_Refund                   7168
Other_Committee_Refund                   7168
Total_Contribution_Refund                7168
Other_Disbursements                      7168
Net_Contribution                         7168
Net_Operating_Expenditure                7168
Cash_On_Hand_BOP                         7340
Debt_Owe_To_Committee                    7847
Coverage_Start_Date                     22132
```

```
state_y                                     37669
dtype: int64
```

In [99]:

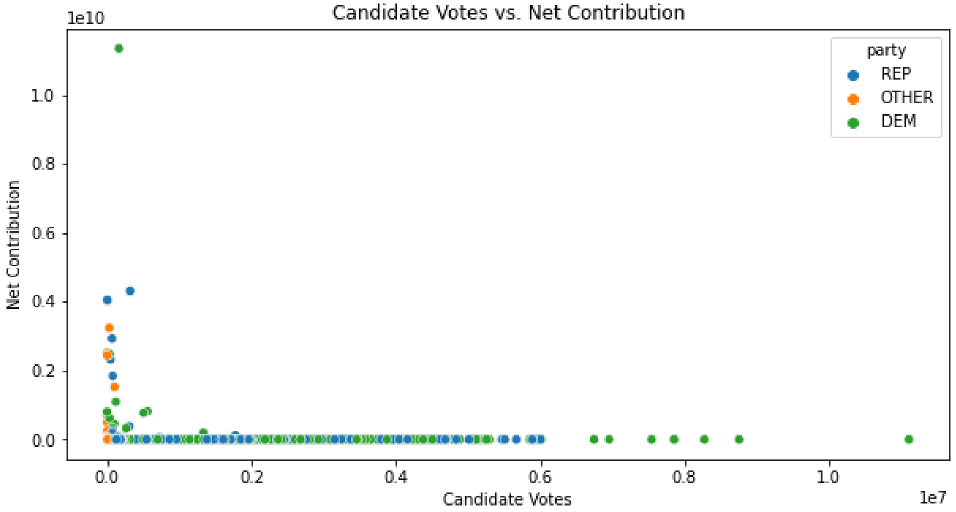
```
# Calculate the correlation matrix
import matplotlib.pyplot as plt
import seaborn as sns

corr_matrix = merged_df.corr()

# Mask the correlation matrix to only show correlations with an absolute value of 0.75 and above
mask = (abs(corr_matrix) >= 0.75)

# Plot the heatmap
plt.figure(figsize=(16, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', annot_kws={'size': 8}, mask=~mask)
plt.title('Correlation Matrix Heatmap (Absolute Value >= 0.75)')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels
plt.tight_layout()  # Adjust layout for better spacing
plt.show()
```



Correlation Matrix Heatmap (Absolute Value >= 0.75)

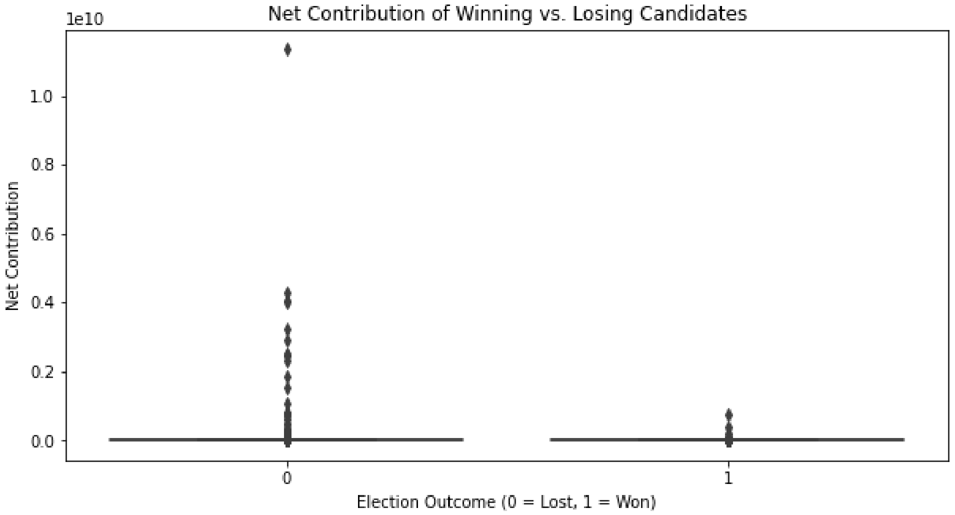
In [100]:

```
# Scatter plot of candidate_votes vs. Net_Contribution
plt.figure(figsize=(10, 5))
sns.scatterplot(data=merged_df, x='candidate_votes', y='Net_Contribution', hue='party')
plt.title('Candidate Votes vs. Net Contribution')
plt.xlabel('Candidate Votes')
plt.ylabel('Net Contribution')
plt.show()
```



In [101]:

```
# Identify the winning candidate for each election
winning_candidates = merged_df.loc[merged_df.groupby(['year', 'state', 'office'])['candidate_votes'].idxmax()]

# Create a new column 'won_election' with a value of 1 for winning candidates and 0 for losing candidates
merged_df['won_election'] = 0
merged_df.loc[winning_candidates.index, 'won_election'] = 1

# Create a box plot comparing the 'Net_Contribution' of winning and losing candidates
plt.figure(figsize=(10, 5))
sns.boxplot(data=merged_df, x='won_election', y='Net_Contribution')
plt.title('Net Contribution of Winning vs. Losing Candidates')
plt.xlabel('Election Outcome (0 = Lost, 1 = Won)')
plt.ylabel('Net Contribution')
plt.show()
```



In [102]:

```
data_2024.head()
```

Out[102]:

	year	office	state	party	Cand_Incumbent_Challenger_Open_Seat	Total_Receipt	Total_Disbursement	Cash_On_Hand_COP	Debt_Owed_By_Committee	Coverage_End_Date	...	Individual_Refund	Party_Committee_Refund	Other_
0	2024	US HOUSE	MD	DEM	INCUMBENT	215321.94	249088.58	875803.55	0.0	6/30/2023	...	3500.00	0	
1	2024	US HOUSE	FL	REP	INCUMBENT	303517.17	122531.78	900791.48	0.0	6/30/2023	...	500.00	0	
2	2024	US HOUSE	GA	REP	INCUMBENT	300379.63	205390.26	825952.82	0.0	6/30/2023	...	0.00	0	
3	2024	US HOUSE	NY	DEM	INCUMBENT	2380863.55	2388793.63	5143974.12	0.0	6/30/2023	...	23447.72	0	
4	2024	US HOUSE	CA	OTHER	CHALLENGER	0.00	0.00	0.00	0.0	NaN	...	0.00	0	

5 rows × 47 columns

In [103]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 1. Preprocess the data further by selecting only relevant features and handling missing values
merged_df = merged_df.dropna(subset=['Net_Contribution'])
X = merged_df[['year', 'state', 'party', 'Net_Contribution']]
y = merged_df['won_election']

# Convert categorical features to numerical using one-hot encoding
X = pd.get_dummies(X, columns=['state', 'party'])

# Create a new DataFrame called data_2024_filtered for filtering and preprocessing
data_2024_filtered = data_2024_original.copy()

# Filter the data_2024_filtered DataFrame to include only presidential elections
data_2024_filtered = data_2024_filtered[data_2024_filtered['office'] == 'US PRESIDENT']

# Add dummy 'state' and 'party' columns if they are not present
if 'state' not in data_2024_filtered.columns:
    data_2024_filtered['state'] = 'Unknown'
if 'party' not in data_2024_filtered.columns:
    data_2024_filtered['party'] = 'Unknown'

# 2. Split the 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)

# 3. Normalize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 4. Create a neural network model
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# 5. Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)

# 6. Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Accuracy: {accuracy}")

# Select only the relevant features
data_2024 = data_2024_filtered[['year', 'state', 'party', 'Net_Contribution']]
```

```python
# Convert categorical features to numerical using one-hot encoding
data_2024 = pd.get_dummies(data_2024, columns=['state', 'party'])

# Align the columns with those in the training data
data_2024 = data_2024.reindex(columns=X.columns, fill_value=0)

# Normalize the data
data_2024_processed = scaler.transform(data_2024)

# Make predictions
predictions = model.predict(data_2024_processed)

# Find the index with the highest probability
winner_index = np.argmax(predictions)

# Get the party affiliation of the winner
winner_party = "DEM" if data_2024_filtered.iloc[winner_index]['party'] == "DEM" else "REP"

# Print the winner's party affiliation
print(f"Winner of the 2024 presidential election: {winner_party}")
```

```
Epoch 1/10
664/664 [==============================] - 1s 937us/step - loss: 0.2308 - accuracy: 0.9171 - val_loss: 0.1829 - val_accuracy: 0.9332
Epoch 2/10
664/664 [==============================] - 1s 800us/step - loss: 0.1865 - accuracy: 0.9307 - val_loss: 0.1817 - val_accuracy: 0.9341
Epoch 3/10
664/664 [==============================] - 1s 783us/step - loss: 0.1826 - accuracy: 0.9312 - val_loss: 0.1759 - val_accuracy: 0.9366
Epoch 4/10
664/664 [==============================] - 1s 800us/step - loss: 0.1804 - accuracy: 0.9330 - val_loss: 0.1750 - val_accuracy: 0.9379
Epoch 5/10
664/664 [==============================] - 1s 776us/step - loss: 0.1788 - accuracy: 0.9325 - val_loss: 0.1732 - val_accuracy: 0.9381
Epoch 6/10
664/664 [==============================] - 1s 786us/step - loss: 0.1778 - accuracy: 0.9336 - val_loss: 0.1789 - val_accuracy: 0.9309
Epoch 7/10
664/664 [==============================] - 1s 808us/step - loss: 0.1762 - accuracy: 0.9332 - val_loss: 0.1724 - val_accuracy: 0.9364
Epoch 8/10
664/664 [==============================] - 1s 803us/step - loss: 0.1749 - accuracy: 0.9353 - val_loss: 0.1730 - val_accuracy: 0.9366
Epoch 9/10
664/664 [==============================] - 1s 823us/step - loss: 0.1745 - accuracy: 0.9338 - val_loss: 0.1707 - val_accuracy: 0.9377
Epoch 10/10
664/664 [==============================] - 1s 823us/step - loss: 0.1736 - accuracy: 0.9354 - val_loss: 0.1745 - val_accuracy: 0.9339
208/208 [==============================] - 0s 578us/step - loss: 0.1745 - accuracy: 0.9355
Accuracy: 0.9355421662330627
34/34 [==============================] - 0s 814us/step
Winner of the 2024 presidential election: DEM
```

Citations

- MIT Election Data and Science Lab, 2017, "U.S. President 1976–2020", https://doi.org/10.7910/DVN/42MVDX (https://doi.org/10.7910/DVN/42MVDX), Harvard Dataverse, V7, UNF:6:MkQHX147hJCgscG5lqK77g== [fileUNF]
- MIT Election Data and Science Lab, 2017, "U.S. Senate statewide 1976–2020" https://doi.org/10.7910/DVN/PEJ5QU (https://doi.org/10.7910/DVN/PEJ5QU), Harvard Dataverse, V6, UNF:6:dogvks8KPD0c/hzNi9kaag== [fileUNF]
- MIT Election Data and Science Lab, 2017, "U.S. House 1976–2022", https://doi.org/10.7910/DVN/IG0UN2 (https://doi.org/10.7910/DVN/IG0UN2), Harvard Dataverse, V12, UNF:6:A6RSZvlhh8eRZ4+mvT/HRQ== [fileUNF]
- Federal Election Commission. (n.d.). Candidates datasets. Retrieved 09/09/2023, from https://www.fec.gov/data/browse-data/?tab=candidates (https://www.fec.gov/data/browse-data/?tab=candidates)

In [ ]: