In [28]:
```
import pandas as pd
import random
from sklearn.metrics import cohen_kappa_score
data = pd.read_excel(r'/Users/alisharai/Desktop/judgements1.xls')
choose_columns = random.sample(list(data.columns[2:9]), 2)
ratings1 = data[choose_columns[0]]
ratings2 = data[choose_columns[1]]
total_ratings = len(ratings1)
agreements = sum(ratings1==ratings2)

observed_agreements= agreements/total_ratings
print("Chosen columns for ratng:", choose_columns)

expected_agreement = 0
unique_ratings = set(ratings1).union(set(ratings2))

for rating in unique_ratings:
    p1= sum(ratings1 == rating)/ total_ratings
    p2= sum(ratings2 == rating)/ total_ratings
    expected_agreement += p1 * p2

kappa = (observed_agreements - expected_agreement)/ (1 - expected_agreement)

print("Overall Cohen's kappa score:", kappa)
```

```
Chosen columns for ratng: [1, 5]
Overall Cohen's kappa score: -0.0069423708569723176
```

In [ ]:
```

```

In [32]:
```
import pandas as pd
import random
from sklearn.metrics import cohen_kappa_score
data = pd.read_excel(r'/Users/alisharai/Desktop/judgements1.xls')
choose_columns = random.sample(list(data.columns[2:9]), 2)
ratings1 = data[choose_columns[0]]
ratings2 = data[choose_columns[1]]
print("Chosen columns for ratng:", choose_columns)
kappa = cohen_kappa_score(ratings1,ratings2)

print("Overall Cohen's kappa score:", kappa)
```

```
Chosen columns for ratng: [1, 7]
Overall Cohen's kappa score: -0.006873344311591589
```

In [33]:
```
import pandas as pd
import random
from sklearn.metrics import cohen_kappa_score

# Read the Excel file
data = pd.read_excel(r'/Users/alisharai/Desktop/judgements1.xls')

# Extract the essay sources from the first column
essay_sources = data.iloc[:, 0].str.extract(r'(\D+)')[0]

# Exclude the first two columns and columns beyond 9
valid_columns = data.columns[2:9]

# Dictionary to store the Cohen's kappa scores for each essay source
kappa_scores = {}

# Iterate over essay sources and calculate Cohen's kappa score
for source in essay_sources.unique():
    # Filter data for the current essay source
    source_data = data[essay_sources == source]

    # Randomly choose two column names for ratings
    chosen_columns = random.sample(list(valid_columns), 2)

    # Extract the ratings from the chosen columns
    rating1 = source_data[chosen_columns[0]]
    rating2 = source_data[chosen_columns[1]]

    # Calculate Cohen's kappa score
    kappa = cohen_kappa_score(rating1, rating2)

    # Store the score for the current essay source
    kappa_scores[source] = kappa

# Print the Cohen's kappa scores for each essay source
for source, kappa in kappa_scores.items():
    print(f"Cohen's kappa score for essay source {source}: {kappa}")
```

```
Cohen's kappa score for essay source G: 0.0
Cohen's kappa score for essay source nan: nan
Cohen's kappa score for essay source F: 0.0
Cohen's kappa score for essay source E: -0.07692307692307687
/Users/alisharai/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py:663: RuntimeWarning: invalid value encountered in true_divide
  k = np.sum(w_mat * confusion) / np.sum(w_mat * expected)
```

In [1]:
```
import pandas as pd
import random
import numpy as np
from sklearn.metrics import cohen_kappa_score

# Read the Excel file
data = pd.read_excel(r'/Users/alisharai/Desktop/judgements1.xls')

# Extract the essay sources from the first column
essay_sources = data.iloc[:, 0].str.extract(r'(\D+)')[0]

# Exclude the first two columns and columns beyond 9
valid_columns = data.columns[2:9]

# Dictionary to store the Cohen's kappa scores for each essay source
kappa_scores = {}

# Iterate over essay sources and calculate Cohen's kappa score
for source in essay_sources.unique():
    # Filter data for the current essay source
    source_data = data[essay_sources == source]

    # Randomly choose two column names for ratings
    chosen_columns = random.sample(list(valid_columns), 2)

    # Extract the ratings from the chosen columns
    rating1 = source_data[chosen_columns[0]]
    rating2 = source_data[chosen_columns[1]]

    # Handle missing values
    rating1 = rating1.replace(' ', np.nan)
    rating2 = rating2.replace(' ', np.nan)

    # Get unique ratings and fill missing values with zeros
    unique_ratings = sorted(set(rating1.dropna().unique()) | set(rating2.dropna().unique()))
    rating1 = rating1.fillna(0)
    rating2 = rating2.fillna(0)

    # Calculate Cohen's kappa score
    kappa = cohen_kappa_score(rating1, rating2, labels=unique_ratings)

    # Store the score for the current essay source
    kappa_scores[source] = kappa

# Print the Cohen's kappa scores for each essay source
for source, kappa in kappa_scores.items():
    print(f"Cohen's kappa score for essay source {source}: {kappa}")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/g9/vbkvkjrj0nq6l1bnbjbp_ds40000gn/T/ipykernel_3895/440397171.py in <module>
     38
     39     # Calculate Cohen's kappa score
---> 40     kappa = cohen_kappa_score(rating1, rating2, labels=unique_ratings)
     41
     42     # Store the score for the current essay source

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py in cohen_kappa_score(y1, y2, labels, weights, sample_weight)
    642             <https://en.wikipedia.org/wiki/Cohen%27s_kappa>`_.
    643     """
--> 644     confusion = confusion_matrix(y1, y2, labels=labels, sample_weight=sample_weight)
    645     n_classes = confusion.shape[0]
    646     sum0 = np.sum(confusion, axis=0)

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/metrics/_classification.py in confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    315         n_labels = labels.size
    316         if n_labels == 0:
--> 317             raise ValueError("'labels' should contains at least one label.")
    318         elif y_true.size == 0:
    319             return np.zeros((n_labels, n_labels), dtype=int)

ValueError: 'labels' should contains at least one label.
```

In [37]:
```
#Code for finding the cohen kappa score of each essay source


import pandas as pd
import random
import numpy as np
from sklearn.metrics import cohen_kappa_score

# Read the Excel file
data = pd.read_excel(r'/Users/alisharai/Desktop/judgements1.xls')

# Extract the essay sources from the first column
essay_sources = data.iloc[:, 0].str.extract(r'(\D+)')[0]



# Dictionary to store the Cohen's kappa scores for each essay source
kappa_scores = {}

# Iterate over essay sources and calculate Cohen's kappa score
for source in essay_sources.unique():
    # Filter data for the current essay source
    source_data = data[essay_sources == source]

    # Randomly choose two column names for ratings


    # Extract the ratings from the chosen columns
    rating1 = source_data[6]
    rating2 = source_data[3]

    # Handle missing values
    rating1 = rating1.replace(' ', np.nan)
    rating2 = rating2.replace(' ', np.nan)

    # Get unique ratings and fill missing values with zeros
    unique_ratings = sorted(set(rating1.dropna().unique()) | set(rating2.dropna().unique()))
    rating1 = rating1.fillna(0)
    rating2 = rating2.fillna(0)

    # Check if unique_ratings is empty
    if not unique_ratings:
        continue

    # Calculate Cohen's kappa score
    kappa = cohen_kappa_score(rating1, rating2, labels=unique_ratings)

    # Store the score for the current essay source
    kappa_scores[source] = kappa

# Print the Cohen's kappa scores for each essay source
for source, kappa in kappa_scores.items():
    print(f"Cohen's kappa score for essay source {source}: {kappa}")
```

```
Cohen's kappa score for essay source G: -0.15189873417721533
Cohen's kappa score for essay source F: -0.14285714285714285
Cohen's kappa score for essay source E: -0.13513513513513487
```

In [ ]:
```

```