Install !pip install pingouin

```
!pip install pingouin
```

```
Collecting pingouin
  Downloading pingouin-0.5.3-py3-none-any.whl (198 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 198.6/198.6 kB 2.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.19 in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.23.5)
Requirement already satisfied: scipy>=1.7 in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.11.4)
Requirement already satisfied: pandas>=1.0 in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.5.3)
Requirement already satisfied: matplotlib>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from pingouin) (3.7.1)
Requirement already satisfied: seaborn>=0.11 in /usr/local/lib/python3.10/dist-packages (from pingouin) (0.12.2)
Requirement already satisfied: statsmodels>=0.13 in /usr/local/lib/python3.10/dist-packages (from pingouin) (0.14.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from pingouin) (1.2.2)
Collecting pandas-flavor>=0.2.0 (from pingouin)
  Downloading pandas_flavor-0.6.0-py3-none-any.whl (7.2 kB)
Collecting outdated (from pingouin)
  Downloading outdated-0.2.2-py2.py3-none-any.whl (7.5 kB)
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (from pingouin) (0.9.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (1.2.0
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (4.45
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (1.4.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (3.1.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.0.2->pingouin) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0->pingouin) (2023.3.post1)
Requirement already satisfied: xarray in /usr/local/lib/python3.10/dist-packages (from pandas-flavor>=0.2.0->pingouin) (2023.7.0)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13->pingouin) (0.5.3)
Requirement already satisfied: setuptools>=44 in /usr/local/lib/python3.10/dist-packages (from outdated->pingouin) (67.7.2)
Collecting littleutils (from outdated->pingouin)
  Downloading littleutils-0.2.2.tar.gz (6.6 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from outdated->pingouin) (2.31.0)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->pingouin) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->pingouin) (3.2.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels>=0.13->pingouin) (1.16
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->outdated->pingoui
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->outdated->pingouin) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->outdated->pingouin) (2.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->outdated->pingouin) (20
Building wheels for collected packages: littleutils
```

```
  Building wheel for littleutils (setup.py) ... done
  Created wheel for littleutils: filename=littleutils-0.2.2-py3-none-any.whl size=7026 sha256=a5fc2c20675a7105873cc542eb624c6734081e
  Stored in directory: /root/.cache/pip/wheels/3d/fe/b0/27a9892da57472e538c7452a721a9cf463cc03cf7379889266
Successfully built littleutils
Installing collected packages: littleutils, outdated, pandas-flavor, pingouin
Successfully installed littleutils-0.2.2 outdated-0.2.2 pandas-flavor-0.6.0 pingouin-0.5.3
```

```
import pandas as pd
import pingouin as pg
```

```
#Annotater Shubham
df_1 = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Training Sheet_Shubham')
```

Double-click (or enter) to edit

```
#Annotater Mustafa
df_2 = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Training Sheet_Mustafa')
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Annotation for Actionability

Based on 3 dimensions Clarity, Specificity and Explainability

```
#Annotater Gopika
df_3 = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Gopika -  Training Sheet Main')
```

```
#Annotater Ali
df_4 = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Ali - Training Sheet Main')
```

Clean Data

```
df_1 = df_1[['id','Warmth','Concern','Acknowledgement']]
df_2 = df_2[['id','Warmth','Concern','Acknowledgement']]
df_3 = df_3[['id','Clarity','Specificity','Explainability']]
df_4 = df_4[['id','Clarity','Specificity','Explainability']]
```

Analysis

Is empathy score higher for lengthier texts?

Annotator Shubham

```
#Annotater Shubham
df_1_full = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Training Sheet_Shubham')
```

```
import numpy as np
df_1_full["length"]=df_1_full["answerText"].str.len()
```

```
df_1_full=df_1_full[['id','Warmth','Concern','Acknowledgement',"length"]]
```

```
df_1_full['Mean Empathy'] = df_1_full[['Warmth', 'Concern', 'Acknowledgement']].mean(axis=1)
```

```
correlation = df_1_full['length'].corr(df_1_full['Mean Empathy'])
```

```
print("Correlation between Length of reply and Overall Empathy for annotator Shubham:", correlation)
```

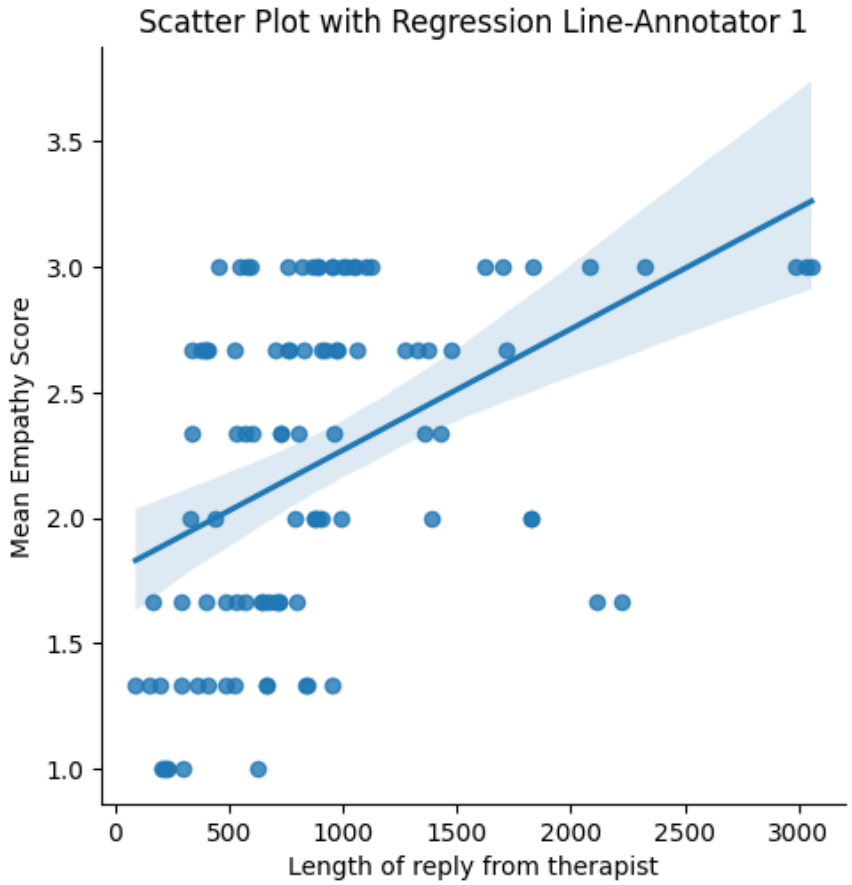
```
    Correlation between Length of reply and Overall Empathy for annotator Shubham: 0.443948408532576
```

```
import scipy.stats as stats
corr, p_value = stats.pearsonr(df_1_full['length'], df_1_full['Mean Empathy'])
p_value
```

```
3.7184019000076554e-06
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Create a scatter plot with a regression line
sns.lmplot(x='length', y='Mean Empathy', data=df_1_full)
plt.title('Scatter Plot with Regression Line-Annotator 1')
plt.xlabel('Length of reply from therapist')
plt.ylabel('Mean Empathy Score')
plt.show()
```



Is empathy score higher for lengthier texts?

Annotator Mustafa

```
#Annotater Mustafa
df_2_full = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Training Sheet_Mustafa')
```

```
import numpy as np
df_2_full["length"]=df_2_full["answerText"].str.len()
```

```
df_2_full=df_2_full[['id','Warmth','Concern','Acknowledgement',"length"]]
```

```
df_2_full['Mean Empathy'] = df_2_full[['Warmth', 'Concern', 'Acknowledgement']].mean(axis=1)
```

```
correlation = df_2_full['length'].corr(df_2_full['Mean Empathy'])
```

```
print("Correlation between Length of reply and Overall Empathy for annotator Mustafa:", correlation)
```

```
Correlation between Length of reply and Overall Empathy for annotator Mustafa: 0.4702036310903028
```

```
corr, p_value = stats.pearsonr(df_2_full['length'], df_2_full['Mean Empathy'])
p_value
```

```
7.972729680658673e-07
```

```
import matplotlib.pyplot as plt
import seaborn as sns


# Create a scatter plot with a regression line
sns.lmplot(x='length', y='Mean Empathy', data=df_2_full)
plt.title('Scatter Plot with Regression Line-Annotator 2')
plt.xlabel('Length of reply from therapist')
plt.ylabel('Mean Empathy Score')
plt.show()
```

Is actionability score higher for lengthier texts?

Annotator Gopika

```
#Annotater Gopika
df_3_full= pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Gopika -  Training Sheet Main')
```

```
df_3_full.rename(columns={'Specificty':'Specificity'},inplace=True)
```

```
import numpy as np
df_3_full["length"]=df_3_full["answerText"].str.len()
```

```
df_3_full=df_3_full[['id','Clarity','Specificity','Explainability',"length"]]
```

```
df_3_full['Mean Actionability'] = df_3_full[['Clarity','Specificity','Explainability']].mean(axis=1)
```

```
correlation = df_3_full['length'].corr(df_3_full['Mean Actionability'])
```

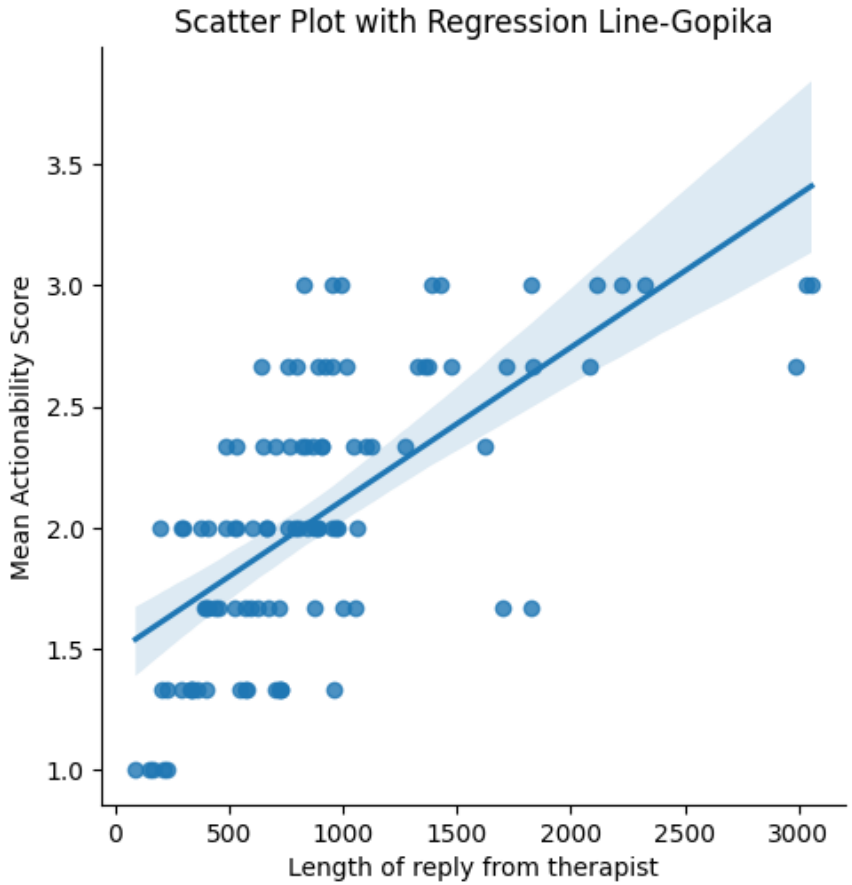
```
print("Correlation between Length of reply and Overall Actionability for annotator Gopika:", correlation)
```

```
Correlation between Length of reply and Overall Actionability for annotator Gopika: 0.6684060540235198
```

```
corr, p_value = stats.pearsonr(df_3_full['length'], df_3_full['Mean Actionability'])
p_value
```

```
3.0023672147530743e-14
```

```
import matplotlib.pyplot as plt
import seaborn as sns


# Create a scatter plot with a regression line
sns.lmplot(x='length', y='Mean Actionability', data=df_3_full)
plt.title('Scatter Plot with Regression Line-Gopika')
plt.xlabel('Length of reply from therapist')
plt.ylabel('Mean Actionability Score')
plt.show()
```



Is actionability score higher for lengthier texts?

Annotator Ali

```
#Annotater Ali
df_4_full= pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Ali - Training Sheet Main')
```

```
import numpy as np
df_4_full["length"]=df_4_full["answerText"].str.len()
df_4_full=df_4_full[['id','Clarity','Specificity','Explainability',"length","topic"]]
df_4_full['Mean Actionability'] = df_4_full[['Clarity','Specificity','Explainability']].mean(axis=1)
```

```
<ipython-input-42-850229ef8b6c>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
  df_4_full['Mean Actionability'] = df_4_full[['Clarity','Specificity','Explainability']].mean(axis=1)
```

```
import scipy.stats as stats
correlation = df_4_full['length'].corr(df_4_full['Mean Actionability'])
```

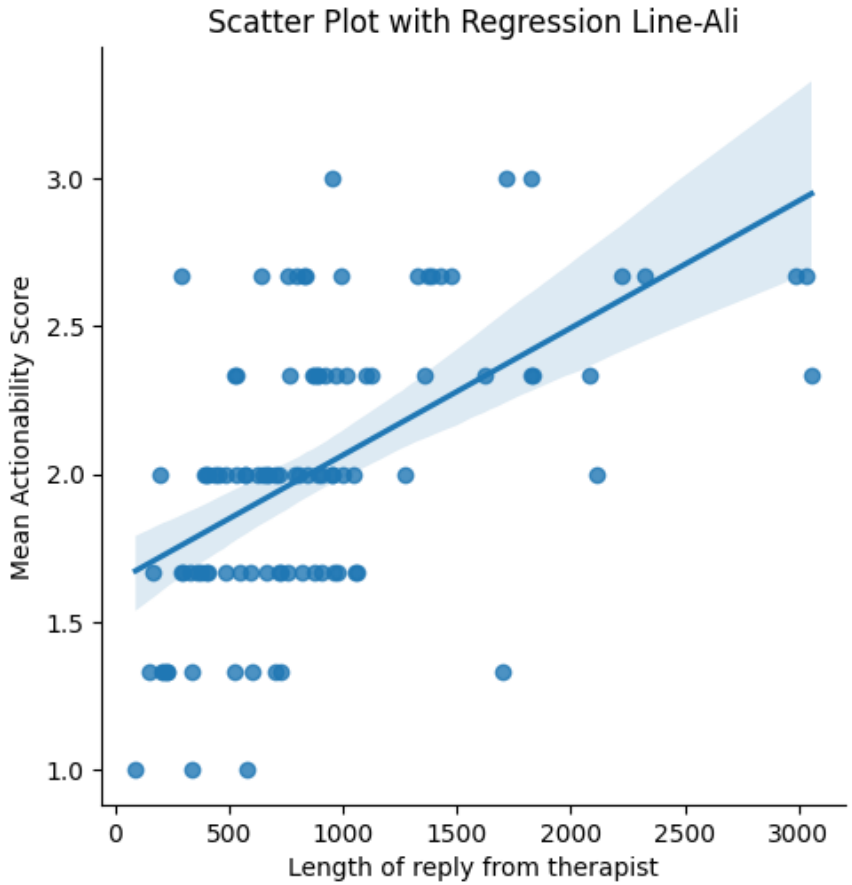
```
print("Correlation between Length of reply and Overall Actionability for annotator Ali:", correlation)
```

```
Correlation between Length of reply and Overall Actionability for annotator Ali: 0.5521024980834964
```

```
corr, p_value = stats.pearsonr(df_4_full['length'], df_4_full['Mean Actionability'])
p_value
```

```
2.6100405102308717e-09
```

```
import matplotlib.pyplot as plt
import seaborn as sns


# Create a scatter plot with a regression line
sns.lmplot(x='length', y='Mean Actionability', data=df_4_full)
plt.title('Scatter Plot with Regression Line-Ali')
plt.xlabel('Length of reply from therapist')
plt.ylabel('Mean Actionability Score')
plt.show()
```



Cohen's Kappa Analysis for Empathy

```
from sklearn.metrics import cohen_kappa_score
kappa_warmth = cohen_kappa_score(df_1_full["Warmth"], df_2_full["Warmth"],weights='quadratic')

print("Quadratic Weighted Kappa:", kappa_warmth)
```

```
Quadratic Weighted Kappa: 0.6532156368221942
```

```
from sklearn.metrics import cohen_kappa_score
kappa_acknowledgement = cohen_kappa_score(df_1_full["Acknowledgement"], df_2_full["Acknowledgement"], weights='quadratic')

print("Quadratic Weighted Kappa:", kappa_acknowledgement)
```

```
Quadratic Weighted Kappa: 0.5792833309154214
```

```
from sklearn.metrics import cohen_kappa_score
kappa_concern = cohen_kappa_score(df_1_full["Concern"], df_2_full["Concern"], weights='quadratic')

print("Quadratic Weighted Kappa:", kappa_concern)
```

```
Quadratic Weighted Kappa: 0.49141767323585506
```

```
import statistics
data_empathy=[kappa_warmth,kappa_acknowledgement,kappa_concern]
# Calculate the mean
mean_value = statistics.mean(data_empathy)

# Calculate the standard deviation
std_value = statistics.stdev(data_empathy)

print("Mean:", mean_value)
print("Standard Deviation:", std_value)
```

```
Mean: 0.5746388803244902
Standard Deviation: 0.08099891015402788
```

Cohen's Kappa Analysis for Actionability

```
from sklearn.metrics import cohen_kappa_score
kappa_specificity = cohen_kappa_score(df_3_full["Specificity"], df_4_full["Specificity"])

print("Quadratic Weighted Kappa:", kappa_specificity)
```

```
Quadratic Weighted Kappa: 0.37339331619537275
```

```
from sklearn.metrics import cohen_kappa_score
kappa_clarity = cohen_kappa_score(df_3_full["Clarity"], df_4_full["Clarity"], weights='quadratic')

print("Quadratic Weighted Kappa:", kappa_clarity)
```

```
Quadratic Weighted Kappa: 0.751727115716753
```

```
from sklearn.metrics import cohen_kappa_score
kappa_explainability = cohen_kappa_score(df_3_full["Explainability"], df_4_full["Explainability"], weights='quadratic')

print("Quadratic Weighted Kappa:", kappa_explainability)
```

```
Quadratic Weighted Kappa: 0.6366939146230699
```

```
import statistics
data_actionability=[kappa_specificity,kappa_clarity,kappa_explainability]
# Calculate the mean
mean_value = statistics.mean(data_actionability)

# Calculate the standard deviation
std_value = statistics.stdev(data_actionability)

print("Mean:", mean_value)
print("Standard Deviation:", std_value)
```

```
Mean: 0.5872714488450652
Standard Deviation: 0.19394857838549506
```

Mean Empathy across topics

```
df_1_full=pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Training Sheet_Shubham')
df_2_full = pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Training Sheet_Mustafa')
```

```
df_combined_1= pd.merge(df_1_full, df_2_full, on='id', how='left')
```

```
df_combined_1.shape
```

```
(100, 42)
```

```
df_combined_1.columns
```

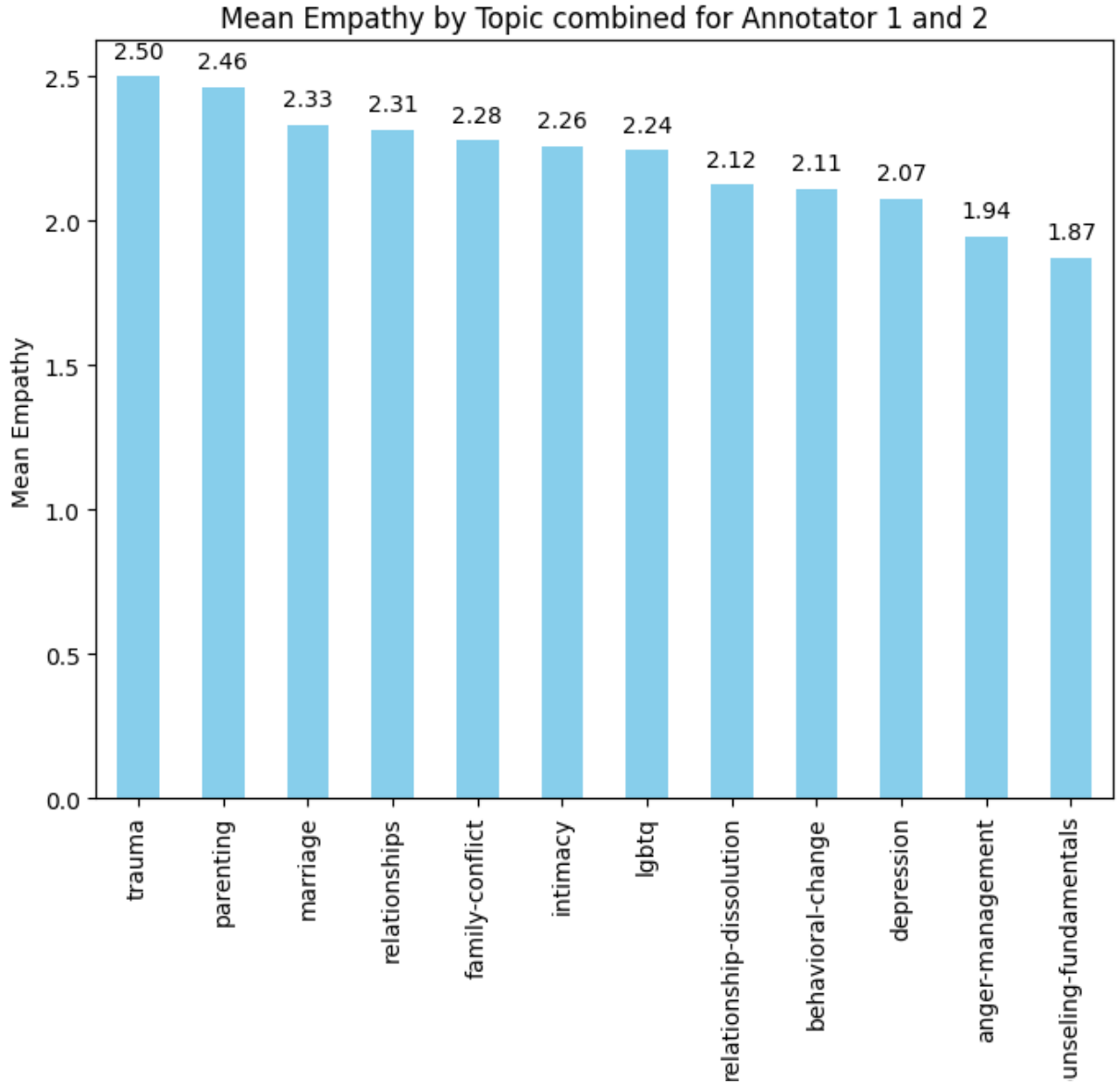
```
Index(['id', 'og_id_x', 'questionID_x', 'questionTitle_x', 'questionText_x',
       'questionLink_x', 'topic_x', 'therapistInfo_x', 'therapistURL_x',
       'answerText_x', 'Warmth_x', 'Concern_x', 'Acknowledgement_x',
       'Specificity_x', 'Clarity_x', 'Practicality (feasible)', 'Unnamed: 16',
       'Unnamed: 17', 'Unnamed: 18', 'Concern.1_x',
       'The therapist conveys concern by seeming to show regard for, and interest in, the patient. The therapist uses vocabulary
and syntax which give the impression that they are involved with the patient and attentive to what the patient has said._x',
       'og_id_y', 'questionID_y', 'questionTitle_y', 'questionText_y',
       'questionLink_y', 'topic_y', 'therapistInfo_y', 'therapistURL_y',
       'answerText_y', 'upvotes', 'views', 'split', 'Warmth_y', 'Concern_y',
       'Acknowledgement_y', 'Specificity_y', 'Clarity_y', 'Practicality',
       'Unnamed: 19', 'Concern.1_y',
       'The therapist conveys concern by seeming to show regard for, and interest in, the patient. The therapist uses vocabulary
and syntax which give the impression that they are involved with the patient and attentive to what the patient has said._y'],
      dtype='object')
```

```
df_combined_1=df_combined_1[['id','topic_x', 'answerText_x','Warmth_x', 'Concern_x', 'Acknowledgement_x','Warmth_y', 'Concern_y',
       'Acknowledgement_y']]
```

```
df_combined_1["Mean Empathy_1_2"]=df_combined_1[['Warmth_x', 'Concern_x', 'Acknowledgement_x','Warmth_y', 'Concern_y',
       'Acknowledgement_y']].mean(axis=1)
df_combined_1["Mean Empathy_1"]=df_combined_1[['Warmth_x', 'Concern_x', 'Acknowledgement_x']].mean(axis=1)
df_combined_1["Mean Empathy_2"]=df_combined_1[['Warmth_y', 'Concern_y', 'Acknowledgement_y']].mean(axis=1)


mean_empathy_by_topic = df_combined_1.groupby('topic_x')['Mean Empathy_1_2'].mean()



mean_empathy_by_topic
```

```
topic_x
anger-management            1.944444
behavioral-change           2.111111
counseling-fundamentals     1.871795
depression                  2.074074
family-conflict             2.277778
intimacy                    2.257576
lgbtq                       2.244444
marriage                    2.333333
parenting                   2.462963
relationship-dissolution    2.125000
relationships               2.314815
trauma                      2.500000
Name: Mean Empathy_1_2, dtype: float64
```

```
mean_empathy_by_topic = mean_empathy_by_topic.sort_values(ascending=False)
plt.figure(figsize=(8, 6))
ax=mean_empathy_by_topic.plot(kind='bar', color='skyblue')
for i, v in enumerate(mean_empathy_by_topic):
    ax.text(i, v + 0.05, f'{v:.2f}', ha='center', va='bottom')
# Add labels and a title
plt.xlabel('Topic')
plt.ylabel('Mean Empathy')
plt.title('Mean Empathy by Topic combined for Annotator 1 and 2')

# Display the chart
plt.show()
```



Mean Actionability across topics

```
df_3_full= pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Gopika -  Training Sheet Main')


df_4_full= pd.read_excel('/content/CounselChat-EDA (1).xlsx',sheet_name='Ali - Training Sheet Main')


df_combined_2= pd.merge(df_3_full, df_4_full, on='id', how='left')


df_combined_2.columns
```

```
    Index(['id', 'og_id_x', 'questionID_x', 'questionTitle_x', 'questionText_x',
           'questionLink_x', 'topic_x', 'therapistInfo_x', 'therapistURL_x',
           'answerText_x', 'upvotes_x', 'views_x', 'split_x', 'Warmth_x',
           'Concern_x', 'Acknowledgement_x', 'Clarity_x', 'Specificity_x',
           'Explainability_x', 'Actionability_Avg_x', 'Unnamed: 20_x',
           'Unnamed: 21_x', 'Unnamed: 22_x', 'og_id_y', 'questionID_y',
           'questionTitle_y', 'questionText_y', 'questionLink_y', 'topic_y',
           'therapistInfo_y', 'therapistURL_y', 'answerText_y', 'upvotes_y',
           'views_y', 'split_y', 'Warmth_y', 'Concern_y', 'Acknowledgement_y',
           'Clarity_y', 'Specificity_y', 'Explainability_y', 'Actionability_Avg_y',
           'Unnamed: 20_y', 'Unnamed: 21_y', 'Unnamed: 22_y'],
          dtype='object')
```

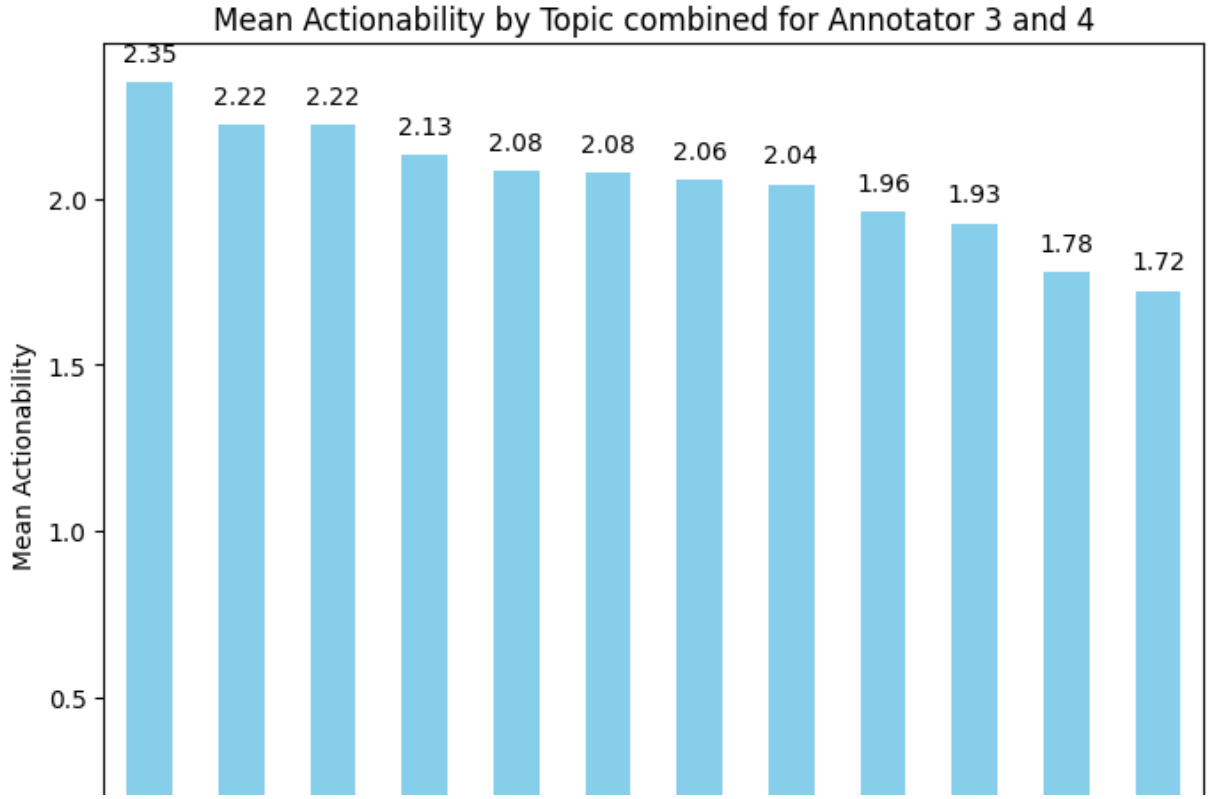
```
df_combined_2=df_combined_2[['id','topic_x', 'answerText_x','Clarity_x', 'Specificity_x',
       'Explainability_x','Clarity_y', 'Specificity_y', 'Explainability_y']]


df_combined_2["Mean Actionability_3_4"]=df_combined_2[['Clarity_x', 'Specificity_x',
       'Explainability_x','Clarity_y', 'Specificity_y', 'Explainability_y']].mean(axis=1)
df_combined_2["Mean Actionability_3"]=df_combined_2[['Clarity_x', 'Specificity_x',
       'Explainability_x']].mean(axis=1)
df_combined_2["Mean Actionability_4"]=df_combined_2[['Clarity_y', 'Specificity_y', 'Explainability_y']].mean(axis=1)
```

```
    <ipython-input-79-550ba755b28c>:1: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
      df_combined_2["Mean Actionability_3_4"]=df_combined_2[['Clarity_x', 'Specificity_x',
    <ipython-input-79-550ba755b28c>:3: SettingWithCopyWarning:
    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
  df_combined_2["Mean Actionability_3"]=df_combined_2[['Clarity_x', 'Specificity_x',
<ipython-input-79-550ba755b28c>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
  df_combined_2["Mean Actionability_4"]=df_combined_2[['Clarity_y', 'Specificity_y', 'Explainability_y']].mean(axis=1)
```

```
mean_actionability_by_topic = df_combined_2.groupby('topic_x')['Mean Actionability_3_4'].mean()
```

```
mean_actionability_by_topic = mean_actionability_by_topic.sort_values(ascending=False)
plt.figure(figsize=(8, 6))
ax=mean_actionability_by_topic.plot(kind='bar', color='skyblue')
for i, v in enumerate(mean_actionability_by_topic):
    ax.text(i, v + 0.05, f'{v:.2f}', ha='center', va='bottom')
# Add labels and a title
plt.xlabel('Topic')
plt.ylabel('Mean Actionability')
plt.title('Mean Actionability by Topic combined for Annotator 3 and 4')

# Display the chart
plt.show()
```



Regression Analysis- Empathy

Using TFIDF

```
pip install pandas scikit-learn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

```
df_empathy=df_combined_1[['id','answerText_x','Mean Empathy_1_2']]
df_empathy["length"]=df_empathy['answerText_x'].str.len()
df_empathy
```

```
<ipython-input-83-73109c4fb138>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
  df_empathy["length"]=df_empathy['answerText_x'].str.len()
```

	id	answerText_x	Mean Empathy_1_2	length
0	1	If everyone thinks you're worthless, then mayb...	2.166667	961
1	2	Hello, and thank you for your question and see...	2.833333	2082
2	3	First thing I'd suggest is getting the sleep y...	1.000000	299
3	4	Therapy is essential for those that are feelin...	1.333333	844
4	5	I first want to let you know that you are not ...	2.500000	337
...	...	...	...	...
95	96	There is some great advice here that can reall...	2.333333	375
96	97	It's a good question man, and it must be terri...	2.666667	576
97	98	Make sure that you continue to treat your Mom ...	1.333333	208
98	99	You have the answer already. It is not your mo...	1.833333	401
99	100	It is very difficult to move from being the ch...	2.500000	905

100 rows × 4 columns

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LinearRegression
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import TfidfVectorizer


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df_empathy[['answerText_x','length']], df_empathy['Mean Empathy_1_2'], test_size=0.2



# Create a pipeline with TF-IDF for text and StandardScaler for text_length, followed by Linear Regression
model = Pipeline([
    ('features', ColumnTransformer([
        ('answerText_x', TfidfVectorizer(), 'answerText_x'),        # Convert text to TF-IDF features
        ('length', StandardScaler(), ['length'])  # Standardize text length
    ])),
    ('regressor', LinearRegression())  # Linear Regression model
])

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

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

# Evaluate the model
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.21670950539886774
```

```
from sklearn.metrics import r2_score

# Calculate R-squared
r2 = r2_score(y_test, predictions)
print(f'R-squared: {r2}')
```

```
R-squared: 0.3184938026329558
```

Regression Analysis- Actionability

Using TFIDF

```
df_actionability=df_combined_2[['id','answerText_x','Mean Actionability_3_4']]
df_actionability["length"]=df_actionability['answerText_x'].str.len()
```

```
<ipython-input-89-18c505b0de23>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
  df_actionability["length"]=df_actionability['answerText_x'].str.len()
```

```
X_train, X_test, y_train, y_test = train_test_split(df_actionability[['answerText_x','length']], df_actionability['Mean Actionability_3_
```

```
# Create a pipeline with TF-IDF for text and StandardScaler for text_length, followed by Linear Regression
model = Pipeline([
    ('features', ColumnTransformer([
        ('answerText_x', TfidfVectorizer(), 'answerText_x'),         # Convert text to TF-IDF features
        ('length', StandardScaler(), ['length'])  # Standardize text length
    ])),
    ('regressor', LinearRegression())  # Linear Regression model
])

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

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

# Evaluate the model
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.1725684592320577
```

```
from sklearn.metrics import r2_score

# Calculate R-squared
r2 = r2_score(y_test, predictions)
print(f'R-squared: {r2}')
```

```
R-squared: 0.28158837440253504
```

BERT based analysis- Empathy

```
df_empathy=df_combined_1[['id','answerText_x','Mean Empathy_1_2']]
df_empathy["length"]=df_empathy['answerText_x'].str.len()
df_empathy
```

```
<ipython-input-99-73109c4fb138>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
  df_empathy["length"]=df_empathy['answerText_x'].str.len()
```

	id	answerText_x	Mean Empathy_1_2	length
0	1	If everyone thinks you're worthless, then mayb...	2.166667	961
1	2	Hello, and thank you for your question and see...	2.833333	2082
2	3	First thing I'd suggest is getting the sleep y...	1.000000	299
3	4	Therapy is essential for those that are feelin...	1.333333	844
4	5	I first want to let you know that you are not ...	2.500000	337
...	...	...	...	...
95	96	There is some great advice here that can reall...	2.333333	375
96	97	It's a good question man, and it must be terri...	2.666667	576
97	98	Make sure that you continue to treat your Mom ...	1.333333	208
98	99	You have the answer already. It is not your mo...	1.833333	401

```
!pip install torch
!pip install transformers
!pip install scikit-learn
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.1.0+cu118)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packages (from torch) (4.5.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.35.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.19.4)
```

```
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.15.0)
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->trans
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2023.11.
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

```
import torch
from transformers import BertTokenizer, BertModel
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pandas as pd


# Split the data
train_data, test_data = train_test_split(df_empathy, test_size=0.2, random_state=42)

# Tokenization using BERT
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
train_tokens = tokenizer(train_data['answerText_x'].tolist(), padding=True, truncation=True, max_length=128, return_tensors='pt')
test_tokens = tokenizer(test_data['answerText_x'].tolist(), padding=True, truncation=True, max_length=128, return_tensors='pt')

# Use BERT embeddings for regression
bert_model = BertModel.from_pretrained('bert-base-uncased')
with torch.no_grad():
    train_embeddings = bert_model(**train_tokens).last_hidden_state.mean(dim=1)
    test_embeddings = bert_model(**test_tokens).last_hidden_state.mean(dim=1)

# Include the length of the reply as another parameter
train_lengths = torch.tensor(train_data['length'].tolist(), dtype=torch.float32).unsqueeze(1)
test_lengths = torch.tensor(test_data['length'].tolist(), dtype=torch.float32).unsqueeze(1)
```

BERT Analysis- Actionability

```
import torch
from transformers import BertTokenizer, BertModel
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pandas as pd
```