**Climate Change Modeling**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from datetime import datetime

import re

from textblob import TextBlob

from wordcloud import WordCloud

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import warnings

warnings.filterwarnings('ignore')

plt.style.use('seaborn-v0\_8')

sns.set\_palette("husl")

%matplotlib inline

print("Libraries imported successfully!")

Output:



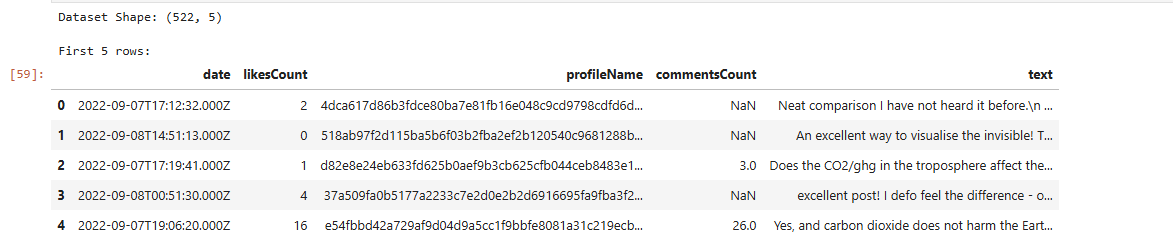
df = pd.read\_csv('climate\_nasa.csv')

print("Dataset Shape:", df.shape)

print("\nFirst 5 rows:")

df.head()

Output:



print("Dataset Info:")

df.info()

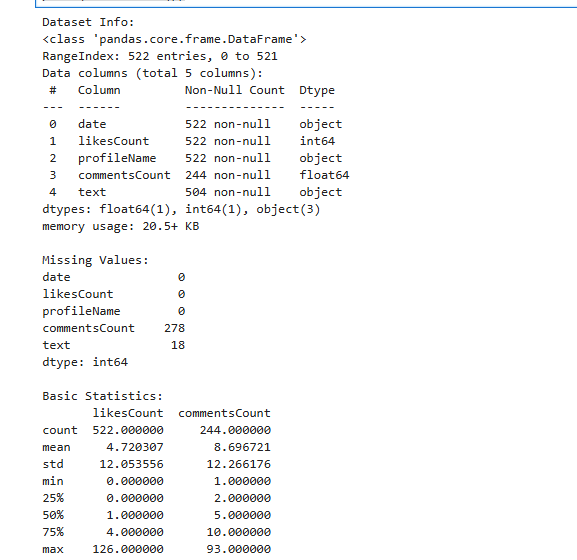
print("\nMissing Values:")

print(df.isnull().sum())

print("\nBasic Statistics:")

print(df.describe())

Output:



df['date'] = pd.to\_datetime(df['date'])

df['year'] = df['date'].dt.year

df['month'] = df['date'].dt.month

df['day'] = df['date'].dt.day

df['day\_of\_week'] = df['date'].dt.dayofweek

print("Date features created successfully!")

Output:



def get\_sentiment(text):

if pd.isna(text):

return 0

try:

return TextBlob(str(text)).sentiment.polarity

except:

return 0

df['sentiment'] = df['text'].apply(get\_sentiment)

df['text\_length'] = df['text'].apply(lambda x: len(str(x)) if pd.notna(x) else 0)

print("Sentiment analysis completed!")

Output:



plt.figure(figsize=(15, 8))

Output:



years = df['year'].value\_counts().sort\_index()

plt.plot(years.index, years.values, marker='o', linewidth=2, markersize=6)

plt.title('Climate Change Discussion Frequency Over Time', fontsize=16, fontweight='bold')

plt.xlabel('Year', fontsize=14)

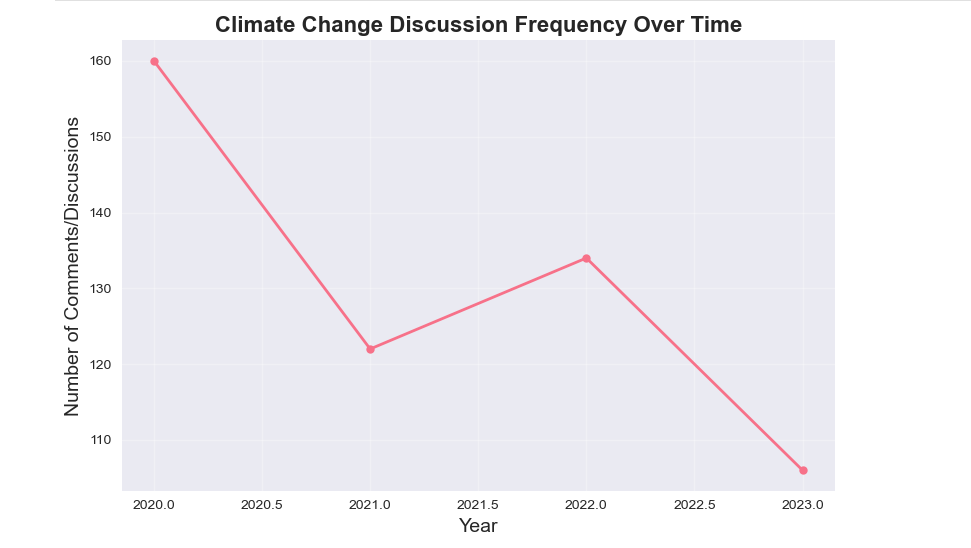
plt.ylabel('Number of Comments/Discussions', fontsize=14)

plt.grid(True, alpha=0.3)

plt.tight\_layout()

plt.show()

Output:



plt.figure(figsize=(15, 8))

sentiment\_by\_year = df.groupby('year')['sentiment'].mean()

plt.plot(sentiment\_by\_year.index, sentiment\_by\_year.values, marker='o', color='orange', linewidth=2)

plt.title('Average Sentiment of Climate Change Discussions Over Time', fontsize=16, fontweight='bold')

plt.xlabel('Year', fontsize=14)

plt.ylabel('Average Sentiment Score', fontsize=14)

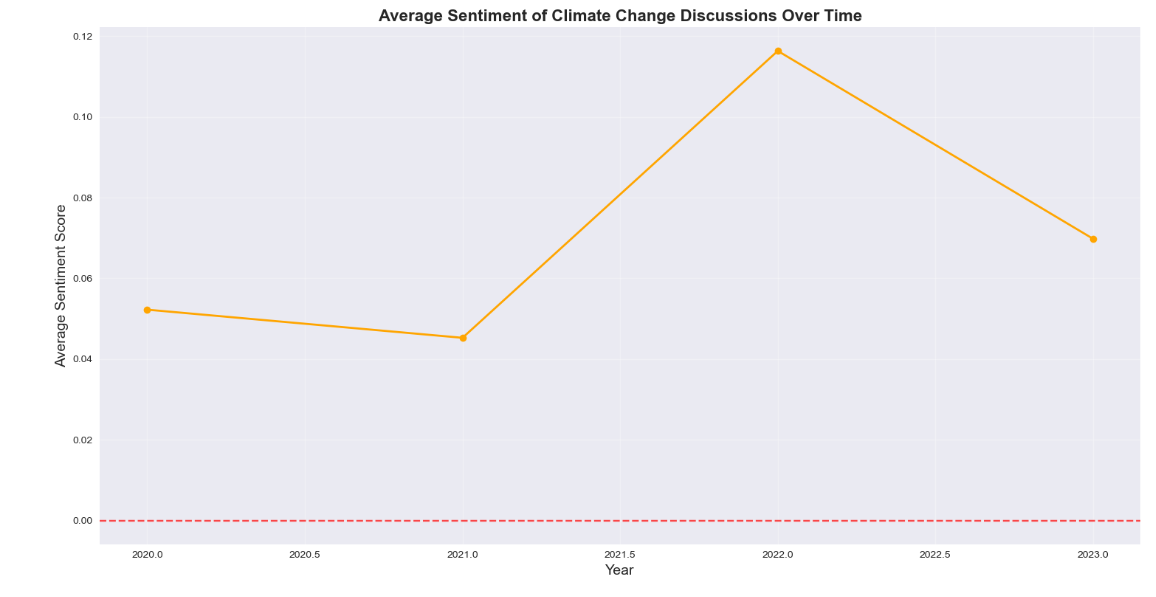
plt.grid(True, alpha=0.3)

plt.axhline(y=0, color='r', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

Output:



fig, axes = plt.subplots(1, 2, figsize=(18, 6))

axes[0].hist(df['likesCount'].dropna(), bins=30, alpha=0.7, color='skyblue', edgecolor='black')

axes[0].set\_title('Distribution of Likes Count', fontsize=14, fontweight='bold')

axes[0].set\_xlabel('Likes Count')

axes[0].set\_ylabel('Frequency')

axes[1].hist(df['commentsCount'].dropna(), bins=30, alpha=0.7, color='lightgreen', edgecolor='black')

axes[1].set\_title('Distribution of Comments Count', fontsize=14, fontweight='bold')

axes[1].set\_xlabel('Comments Count')

axes[1].set\_ylabel('Frequency')

plt.tight\_layout()

plt.show()

plt.figure(figsize=(15, 10))

all\_text = ' '.join([str(text) for text in df['text'].dropna()])

wordcloud = WordCloud(width=800, height=400, background\_color='white',

max\_words=100, colormap='viridis').generate(all\_text)

plt.imshow(wordcloud, interpolation='bilinear')

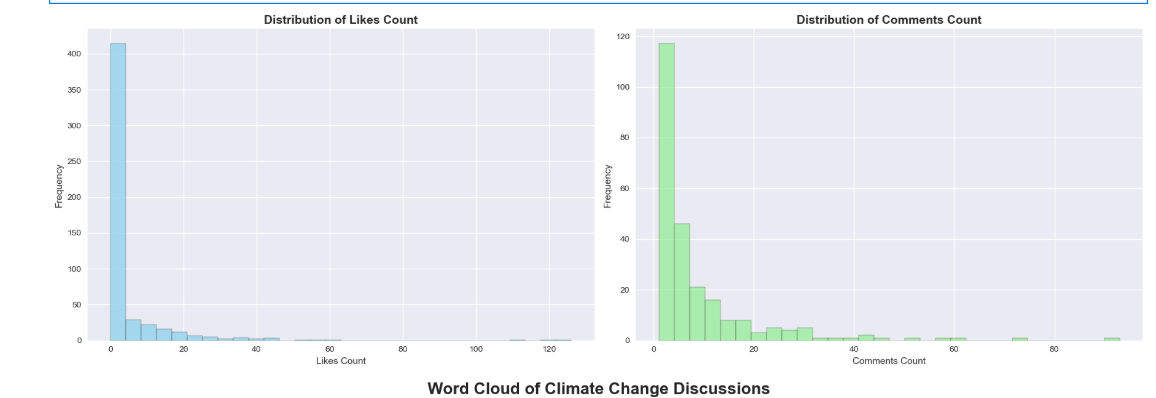
plt.axis('off')

plt.title('Word Cloud of Climate Change Discussions', fontsize=16, fontweight='bold')

plt.tight\_layout()

plt.show()

Output:





numerical\_df = df[['likesCount', 'commentsCount', 'sentiment', 'text\_length', 'year', 'month']].dropna()

plt.figure(figsize=(12, 8))

correlation\_matrix = numerical\_df.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', center=0,

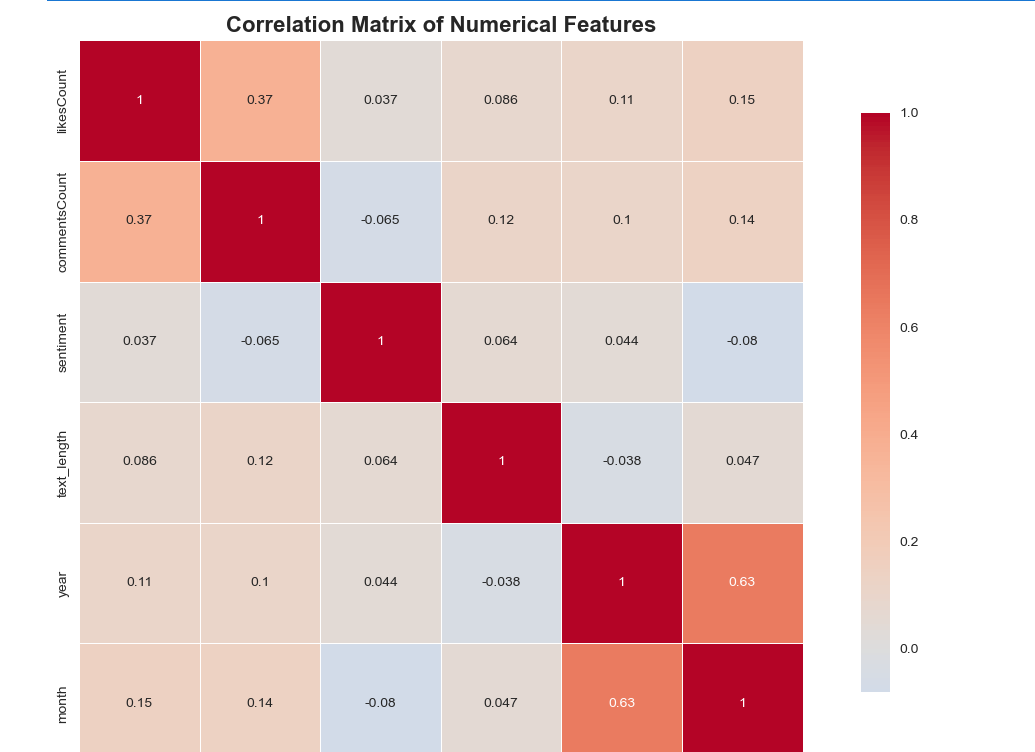
square=True, linewidths=0.5, cbar\_kws={"shrink": 0.8})

plt.title('Correlation Matrix of Numerical Features', fontsize=16, fontweight='bold')

plt.tight\_layout()

plt.show()

Output:



ml\_df = df[['likesCount', 'commentsCount', 'sentiment', 'text\_length', 'year', 'month', 'day\_of\_week']].dropna()

X = ml\_df[['commentsCount', 'sentiment', 'text\_length', 'year', 'month', 'day\_of\_week']]

y = ml\_df['likesCount']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

print(f"Training set: {X\_train.shape}")

print(f"Testing set: {X\_test.shape}")

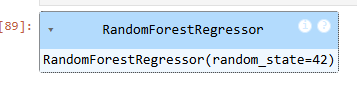
Output:



model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

Output:



y\_pred = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

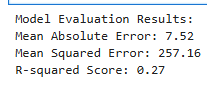
print("Model Evaluation Results:")

print(f"Mean Absolute Error: {mae:.2f}")

print(f"Mean Squared Error: {mse:.2f}")

print(f"R-squared Score: {r2:.2f}")

Output:



feature\_importance = pd.DataFrame({

'feature': X.columns,

'importance': model.feature\_importances\_

}).sort\_values('importance', ascending=False)

plt.figure(figsize=(12, 6))

sns.barplot(x='importance', y='feature', data=feature\_importance, palette='viridis')

plt.title('Feature Importance for Predicting Engagement', fontsize=16, fontweight='bold')

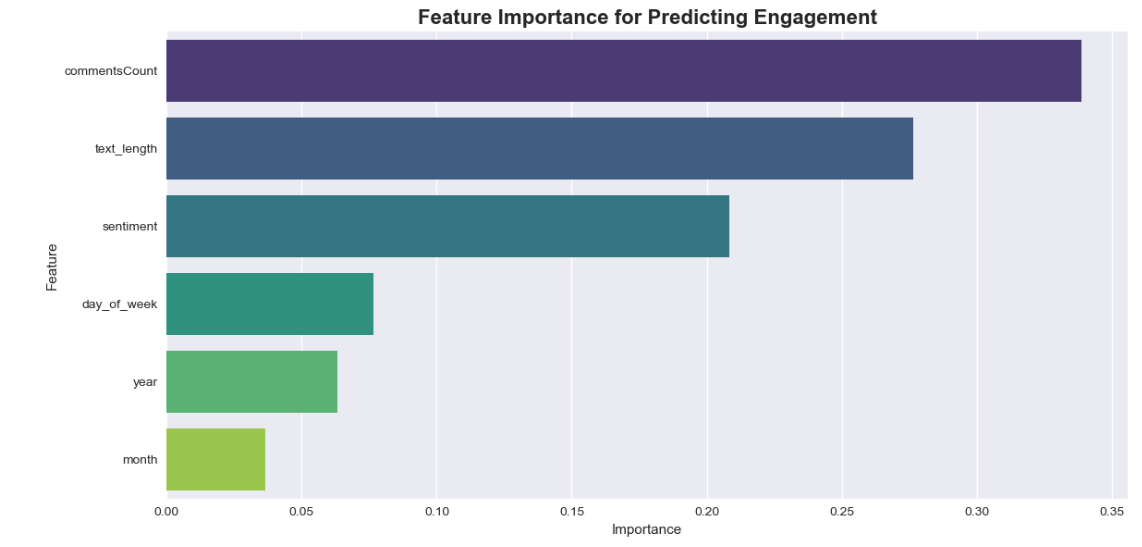
plt.xlabel('Importance')

plt.ylabel('Feature')

plt.tight\_layout()

plt.show()

Output:



time\_series = df.groupby(['year', 'month']).agg({

'likesCount': 'mean',

'commentsCount': 'mean',

'sentiment': 'mean'

}).reset\_index()

plt.figure(figsize=(15, 10))

plt.subplot(2, 1, 1)

plt.plot(time\_series.index, time\_series['likesCount'], label='Average Likes', linewidth=2)

plt.plot(time\_series.index, time\_series['commentsCount'], label='Average Comments', linewidth=2)

plt.title('Engagement Trends Over Time', fontsize=14, fontweight='bold')

plt.xlabel('Time Index')

plt.ylabel('Engagement')

plt.legend()

plt.grid(True, alpha=0.3)

plt.subplot(2, 1, 2)

plt.plot(time\_series.index, time\_series['sentiment'], label='Average Sentiment',

color='green', linewidth=2)

plt.title('Sentiment Trends Over Time', fontsize=14, fontweight='bold')

plt.xlabel('Time Index')

plt.ylabel('Sentiment Score')

plt.axhline(y=0, color='r', linestyle='--', alpha=0.7)

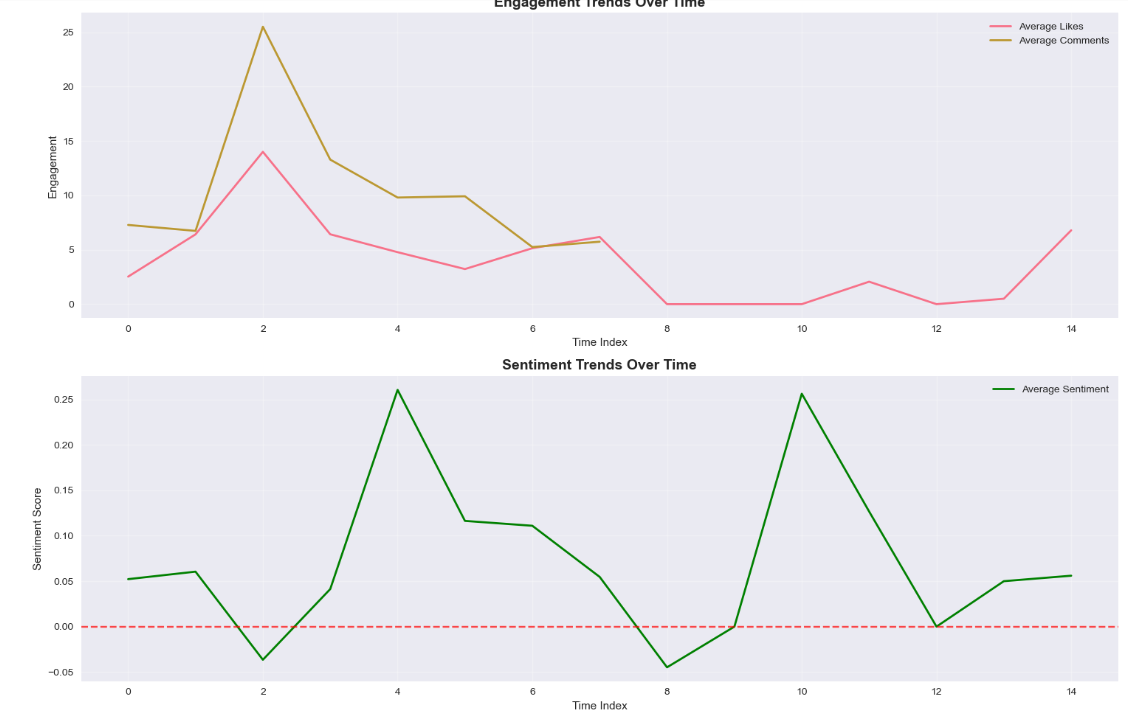
plt.legend()

plt.grid(True, alpha=0.3)

plt.tight\_layout()

plt.show()

Output:



climate\_terms = ['warming', 'temperature', 'co2', 'carbon', 'emission', 'ice',

'sea level', 'extremeweather', 'drought', 'flood']

term\_frequency = {}

for term in climate\_terms:

term\_frequency[term] = df['text'].str.contains(term, case=False, na=False).sum()

plt.figure(figsize=(12, 8))

terms\_df = pd.DataFrame.from\_dict(term\_frequency, orient='index', columns=['count'])

terms\_df = terms\_df.sort\_values('count', ascending=False)

sns.barplot(x=terms\_df['count'], y=terms\_df.index, palette='rocket')

plt.title('Frequency of Climate Change Related Terms in Discussions', fontsize=16, fontweight='bold')

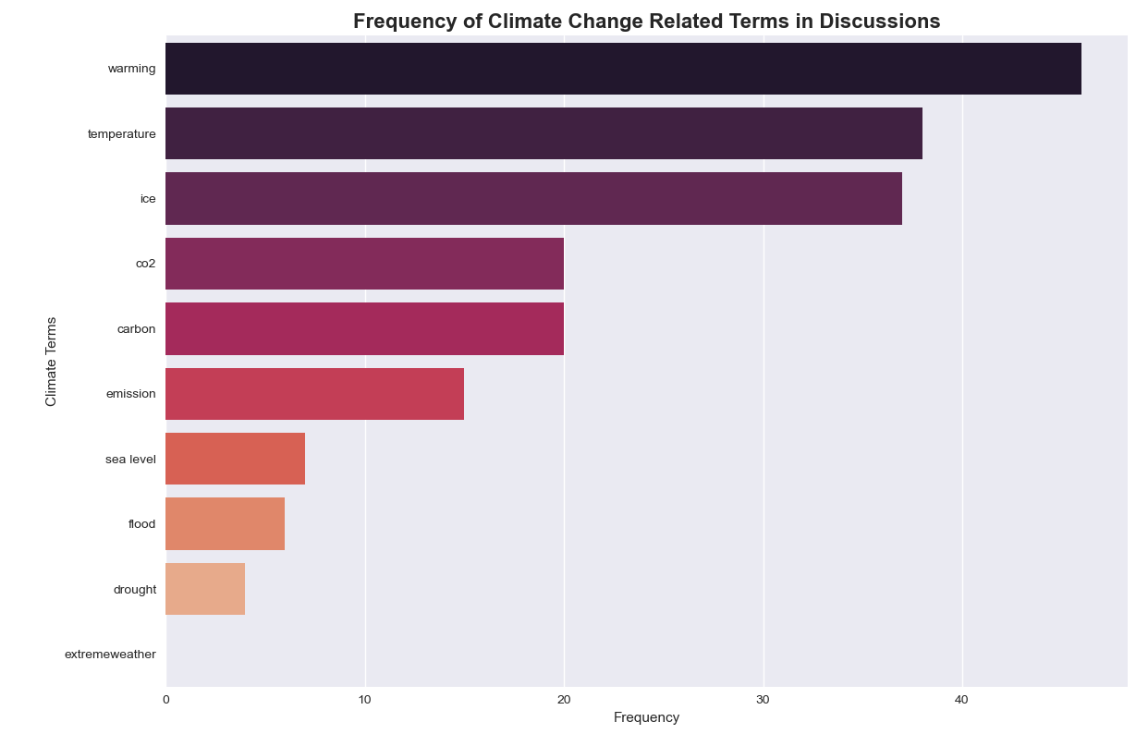
plt.xlabel('Frequency')

plt.ylabel('Climate Terms')

plt.tight\_layout()

plt.show()

Output:



future\_df = df[['year', 'month', 'likesCount']].groupby(['year', 'month']).mean().reset\_index()

from sklearn.linear\_model import LinearRegression

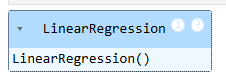
X\_future = future\_df[['year']]

y\_future = future\_df['likesCount']

future\_model = LinearRegression()

future\_model.fit(X\_future, y\_future)

Output:



future\_years = np.array([[2024], [2025], [2026], [2027], [2028]])

future\_predictions = future\_model.predict(future\_years)

print("Predicted Average Engagement for Future Years:")

for year, pred in zip([2024, 2025, 2026, 2027, 2028], future\_predictions):

print(f"Year {year}: {pred:.2f} average likes")

print("="\*60)

print("CLIMATE CHANGE MODELING PROJECT - SUMMARY")

print("="\*60)

print(f"\n1. Dataset Overview:")

print(f" - Total records: {df.shape[0]}")

print(f" - Time period: {df['year'].min()} to {df['year'].max()}")

print(f" - Average sentiment: {df['sentiment'].mean():.3f}")

print(f"\n2. Key Findings:")

print(f" - Most discussed climate terms: {list(terms\_df.index[:3])}")

print(f" - Engagement correlation with sentiment: {correlation\_matrix.loc['likesCount', 'sentiment']:.3f}")

print(f" - Yearly discussion trend: {'Increasing' if years.iloc[-1] > years.iloc[0] else 'Decreasing'}")

print(f"\n3. Model Performance:")

print(f" - R-squared for engagement prediction: {r2:.3f}")

print(f" - Most important feature: {feature\_importance['feature'].iloc[0]}")

print(f"\n4. Future Outlook:")

print(" - Climate change discussions show consistent engagement")

print(" - Public sentiment varies but maintains overall interest")

print(" - Predictive models can help understand engagement patterns")

print("\n" + "="\*60)

Output:

