

```
import pandas as pd

# Load the dataset
file_path = "/mnt/data/mental_health_seasonal_sentiment_dataset.csv"
df = pd.read_csv(file_path)

# Display basic information and the first few rows
df.info(), df.head()
```

Task

Tell me about this dataset.

Here is all the data you need: "mental_health_seasonal_sentiment_dataset.csv"

Data loading

Subtask:

Load the dataset "mental_health_seasonal_sentiment_dataset.csv" into a pandas DataFrame.

Reasoning: Load the dataset into a pandas DataFrame and display the first few rows.

```
import pandas as pd

try:
    df = pd.read_csv('mental_health_seasonal_sentiment_dataset.csv')
    display(df.head())
except FileNotFoundError:
    print("Error: 'mental_health_seasonal_sentiment_dataset.csv' not found.")
except Exception as e:
    print(f"An error occurred: {e}")

{"summary": "{\n  \"name\": \"    print(f\\\"\\\"\\\"An error occurred:\n  {e}\\\"\\\"\\\"}\\\"\", \n  \"rows\": 5, \n  \"fields\": [\n    {\n      \"column\":\n        \"post_id\", \n      \"properties\": {\n        \"dtype\": \"string\", \n        \"num_unique_values\": 5, \n        \"samples\": [\n          \"t3_520818\", \n          \"t3_976717\", \n          \"t3_588603\" \n        ], \n        \"semantic_type\": \"\", \n        \"description\": \"\" \n      }, \n      {\n        \"column\": \"subreddit\", \n        \"properties\": {\n          \"dtype\": \"category\", \n          \"num_unique_values\": 2, \n          \"samples\": [\n            \"r/depression\", \n            \"r/BipolarReddit\" \n          ], \n          \"semantic_type\": \"\", \n          \"description\": \"\" \n        }, \n        {\n          \"column\": \"title\", \n          \"properties\": {\n            \"dtype\": \"string\", \n            \"num_unique_values\": 4, \n            \"samples\": [\n              \"Struggling with motivation\", \n
```

```

{"I feel overwhelmed today",
 "description": "I feel overwhelmed today",
 "column": "body",
 "properties": {"dtype": "string",
 "num_unique_values": 4,
 "samples": ["Maybe things will turn out okay.",
 "Deep breaths and journaling are helping."]},
 "semantic_type": "text",
 "description": "I feel overwhelmed today",
 "column": "created_utc",
 "properties": {"dtype": "object",
 "num_unique_values": 5,
 "samples": ["2023-08-10 18:40:53",
 "2023-03-18 20:48:37"]},
 "semantic_type": "text",
 "description": "I feel overwhelmed today",
 "column": "season",
 "properties": {"dtype": "string",
 "num_unique_values": 3,
 "samples": ["Spring",
 "Summer"]},
 "semantic_type": "text",
 "description": "I feel overwhelmed today",
 "column": "sentiment_score",
 "properties": {"dtype": "number",
 "std": 0.570894035701898,
 "min": -0.65,
 "max": 0.73,
 "num_unique_values": 5,
 "samples": [-0.14,
 -0.65]},
 "semantic_type": "text",
 "description": "I feel overwhelmed today",
 "column": "label",
 "properties": {"dtype": "category",
 "num_unique_values": 2,
 "samples": ["negative",
 "positive"]},
 "semantic_type": "text",
 "description": "I feel overwhelmed today",
 "type": "dataframe"}

```

Data exploration

Subtask:

Explore the dataset to understand its structure, including the number of rows and columns, data types of each column, and summary statistics for numerical columns. Identify potential missing values and unique values in categorical columns.

Reasoning: Explore the dataset's structure, data types, summary statistics, missing values, and unique values in categorical columns.

```

# Examine the shape of the DataFrame.
print("Shape of the DataFrame:", df.shape)

# Get a concise summary of the DataFrame.
print("\nDataFrame Info:")
df.info()

# Generate descriptive statistics for numerical columns.
print("\nDescriptive Statistics:")
print(df.describe())

```

```
# Identify unique values in categorical columns and their distribution.
```

```
categorical_cols = ['season', 'subreddit', 'label']
```

```
for col in categorical_cols:
    print(f"\nUnique values in '{col}':")
    print(df[col].unique())
    print(f"\nValue counts for '{col}':")
    print(df[col].value_counts())
```

```
# Check for missing values.
```

```
print("\nMissing values per column:")
```

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

```
Shape of the DataFrame: (200, 8)
```

```
DataFrame Info:
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 200 entries, 0 to 199
```

```
Data columns (total 8 columns):
```

#	Column	Non-Null Count	Dtype
0	post_id	200 non-null	object
1	subreddit	200 non-null	object
2	title	200 non-null	object
3	body	200 non-null	object
4	created_utc	200 non-null	object
5	season	200 non-null	object
6	sentiment_score	200 non-null	float64
7	label	200 non-null	object

```
dtypes: float64(1), object(7)
```

```
memory usage: 12.6+ KB
```

```
Descriptive Statistics:
```

	sentiment_score
count	200.000000
mean	0.075950
std	0.555291
min	-0.990000
25%	-0.310000
50%	0.120000
75%	0.525000
max	1.000000

```
Unique values in 'season':
```

```
['Spring' 'Summer' 'Fall' 'Winter']
```

```
Value counts for 'season':
```

```
season
Spring    58
```

```

Summer      55
Fall        47
Winter      40
Name: count, dtype: int64

Unique values in 'subreddit':
['r/BipolarReddit' 'r/depression' 'r/mentalhealth' 'r/SuicideWatch'
 'r/Anxiety']

Value counts for 'subreddit':
subreddit
r/BipolarReddit      44
r/mentalhealth       44
r/depression         39
r/SuicideWatch       39
r/Anxiety            34
Name: count, dtype: int64

Unique values in 'label':
['positive' 'negative' 'neutral']

Value counts for 'label':
label
positive      101
negative       80
neutral        19
Name: count, dtype: int64

Missing values per column:
post_id      0
subreddit    0
title        0
body         0
created_utc  0
season       0
sentiment_score  0
label        0
dtype: int64

```

Data analysis

Subtask:

Analyze the distribution of sentiment scores across different seasons and other relevant columns. Calculate descriptive statistics for the sentiment scores grouped by season.

Reasoning: Group data by 'season' and calculate descriptive statistics for 'sentiment_score'. Then, extend this analysis to include other categorical columns like 'subreddit' and 'label', calculating the same descriptive statistics for each group.

```

# Group data by season and calculate descriptive statistics for
sentiment_score.
season_stats = df.groupby('season')['sentiment_score'].agg(['mean',
'median', 'std', 'quantile'])
display(season_stats)

# Group data by subreddit and season, then calculate descriptive
statistics.
subreddit_season_stats = df.groupby(['subreddit', 'season'])
['sentiment_score'].agg(['mean', 'median', 'std', 'quantile'])
display(subreddit_season_stats)

# Group data by label and season, then calculate descriptive
statistics.
label_season_stats = df.groupby(['label', 'season'])
['sentiment_score'].agg(['mean', 'median', 'std', 'quantile'])
display(label_season_stats)

# Create a pivot table to summarize sentiment scores across multiple
categorical variables.
pivot_table = pd.pivot_table(df, values='sentiment_score',
index='season', columns=['subreddit', 'label'], aggfunc=['mean',
'median'])
display(pivot_table)

{"summary":{"\n  \"name\": \"season_stats\",\n  \"rows\": 4,\n  \"fields\": [\n    {\n      \"column\": \"season\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 4,\n        \"samples\": [\n          \"Spring\",\n          \"Winter\",\n          \"Fall\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"mean\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.03847228898996341,\n        \"min\": 0.0441818181818176,\n        \"max\": 0.12127659574468086,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          0.05017241379310345,\n          0.10375000000000001,\n          0.12127659574468086\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"median\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.10873323625583242,\n        \"min\": 0.03,\n        \"max\": 0.27,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          0.08499999999999999,\n          0.2,\n          0.27\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"std\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.053407300757291415,\n        \"min\": 0.5181258638346582,\n        \"max\": 0.6337693210202116,\n        \"num_unique_values\": 4,\n        \"samples\": [\n          0.5237231469624452,\n          0.5668703442406272,\n          0.6337693210202116\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}}

```

```
n    },\n    {\n        \"column\": \"quantile\", \n        \"properties\":  
{\n            \"dtype\": \"number\", \n            \"std\":  
0.10873323625583242, \n            \"min\": 0.03, \n            \"max\": 0.27, \n            \"num_unique_values\": 4, \n            \"samples\": [\n0.08499999999999999, \n            0.2, \n            0.27\n            ], \n            \"semantic_type\": \"\", \n            \"description\": \"\"\n        }\n    }\n  ], \"type\": \"dataframe\", \"variable_name\": \"season_stats\"}
```

```
{\"summary\": \"{ \n  \"name\": \"subreddit_season_stats\", \n  \"rows\": 20, \n  \"fields\": [\n    {\n      \"column\": \"mean\", \n      \"properties\": {\n        \"dtype\": \"number\", \n        \"std\": 0.16709203500064143, \n        \"min\": -0.22249999999999998, \n        \"max\": 0.5, \n        \"num_unique_values\": 19, \n        \"samples\": [\n          0.1975, \n          0.15647058823529414, \n          0.23272727272727273\n        ], \n        \"semantic_type\": \"\", \n        \"description\": \"\"\n      }, \n      {\n        \"column\": \"median\", \n        \"properties\": {\n          \"dtype\": \"number\", \n          \"std\": 0.2913005095122651, \n          \"min\": -0.375, \n          \"max\": 0.79, \n          \"num_unique_values\": 19, \n          \"samples\": [\n            0.5449999999999999, \n            0.27, \n            0.46\n          ], \n          \"semantic_type\": \"\", \n          \"description\": \"\"\n        }, \n        {\n          \"column\": \"std\", \n          \"properties\": {\n            \"dtype\": \"number\", \n            \"std\": 0.10926845602514534, \n            \"min\": 0.30349158613896177, \n            \"max\": 0.7376749187238818, \n            \"num_unique_values\": 20, \n            \"samples\": [\n              0.6828668558783204, \n              0.49562642741842927, \n              0.532838027580956\n            ], \n            \"semantic_type\": \"\", \n            \"description\": \"\"\n          }, \n          {\n            \"column\": \"quantile\", \n            \"properties\": {\n              \"dtype\": \"number\", \n              \"std\": 0.2913005095122651, \n              \"min\": -0.375, \n              \"max\": 0.79, \n              \"num_unique_values\": 19, \n              \"samples\": [\n                0.5449999999999999, \n                0.27, \n                0.46\n              ], \n              \"semantic_type\": \"\", \n              \"description\": \"\"\n            }\n          }\n        ]\n      }, \n      {\n        \"column\": \"quantile\", \n        \"properties\": {\n          \"dtype\": \"number\", \n          \"std\": 0.2913005095122651, \n          \"min\": -0.375, \n          \"max\": 0.79, \n          \"num_unique_values\": 19, \n          \"samples\": [\n            0.5449999999999999, \n            0.27, \n            0.46\n          ], \n          \"semantic_type\": \"\", \n          \"description\": \"\"\n        }\n      }\n    ]\n  }, \n  {\n    \"column\": \"quantile\", \n    \"properties\": {\n      \"dtype\": \"number\", \n      \"std\": 0.2913005095122651, \n      \"min\": -0.375, \n      \"max\": 0.79, \n      \"num_unique_values\": 19, \n      \"samples\": [\n        0.5449999999999999, \n        0.27, \n        0.46\n      ], \n      \"semantic_type\": \"\", \n      \"description\": \"\"\n    }\n  }\n  ], \"type\": \"dataframe\", \"variable_name\": \"subreddit_season_stats\"}
```

```
{\"summary\": \"{ \n  \"name\": \"label_season_stats\", \n  \"rows\": 12, \n  \"fields\": [\n    {\n      \"column\": \"mean\", \n      \"properties\": {\n        \"dtype\": \"number\", \n        \"std\": 0.45243313776490274, \n        \"min\": -0.5761111111111111, \n        \"max\": 0.648, \n        \"num_unique_values\": 12, \n        \"samples\": [\n          0.5034615384615384, \n          0.48517241379310344, \n          -0.5761111111111111\n        ], \n        \"semantic_type\": \"\", \n        \"description\": \"\"\n      }, \n      {\n        \"column\": \"median\", \n        \"properties\": {\n          \"dtype\": \"number\", \n          \"std\": 0.4147607926884063, \n          \"min\": -0.595, \n          \"max\": 0.65, \n          \"num_unique_values\": 12, \n          \"samples\": [\n            0.435, \n            0.45, \n            -0.595\n          ], \n          \"semantic_type\": \"\", \n          \"description\": \"\"\n        }\n      }\n    ]\n  }, \n  {\n    \"column\": \"median\", \n    \"properties\": {\n      \"dtype\": \"number\", \n      \"std\": 0.4147607926884063, \n      \"min\": -0.595, \n      \"max\": 0.65, \n      \"num_unique_values\": 12, \n      \"samples\": [\n        0.435, \n        0.45, \n        -0.595\n      ], \n      \"semantic_type\": \"\", \n      \"description\": \"\"\n    }\n  }\n  ], \"type\": \"dataframe\", \"variable_name\": \"label_season_stats\"}
```

```

{"semantic_type": "\"",\n      "description": "\""\n    },\n    {\n      "column": "std",\n      "properties": {\n        "dtype": "number",\n        "std": 0.10369255075507,\n        "min": 0.04690415759823429,\n        "max": 0.297897315290884,\n        "num_unique_values": 12,\n        "samples": [\n          0.22358787637423114,\n          0.2828758017431372,\n          0.297897315290884\n        ],\n        "semantic_type": "\"",\n        "description": "\""\n      },\n      {\n        "column": "quantile",\n        "properties": {\n          "dtype": "number",\n          "std": 0.4147607926884063,\n          "min": -0.595,\n          "max": 0.65,\n          "num_unique_values": 12,\n          "samples": [\n            0.435,\n            0.45,\n            -0.595\n          ],\n          "semantic_type": "\"",\n          "description": "\""\n        }\n      }\n    ]\n  },\n  "type": "dataframe",\n  "variable_name": "label_season_stats"\n}\n\n{"type": "dataframe",\n "variable_name": "pivot_table"}

```

Data visualization

Subtask:

Visualize the distribution of sentiment scores and their relationships with season, subreddit, and sentiment label.

Reasoning: Create the visualizations as requested in the subtask instructions. This includes box plots, violin plots, bar plots, heatmaps, and an interactive scatter plot. Import necessary libraries within the code block.

```

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# 1. Box plot of sentiment_score by season
plt.figure(figsize=(10, 6))
sns.boxplot(x='season', y='sentiment_score', data=df, palette="Set3")
plt.title('Distribution of Sentiment Score by Season')
plt.xlabel('Season')
plt.ylabel('Sentiment Score')
plt.show()

# 2. Violin plot of sentiment_score by subreddit
plt.figure(figsize=(12, 6))
sns.violinplot(x='subreddit', y='sentiment_score', data=df,
palette="husl")
plt.title('Distribution of Sentiment Score by Subreddit')
plt.xlabel('Subreddit')
plt.ylabel('Sentiment Score')
plt.xticks(rotation=45, ha='right') # Rotate x-axis labels for better

```

```

readability
plt.tight_layout()
plt.show()

# 3. Bar plot of average sentiment_score by season and label
season_label_stats = df.groupby(['season', 'label'])
['sentiment_score'].agg(['mean', 'std'])
season_label_stats = season_label_stats.reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(x='season', y='mean', hue='label',
data=season_label_stats, palette="Paired", ci='sd')
plt.title('Average Sentiment Score by Season and Label')
plt.xlabel('Season')
plt.ylabel('Average Sentiment Score')
plt.show()

# 4. Heatmap of correlations
numerical_cols = ['sentiment_score', 'created_utc']
correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix of Numerical Features')
plt.show()

# 5. Interactive scatter plot
df['created_utc'] = pd.to_datetime(df['created_utc'])

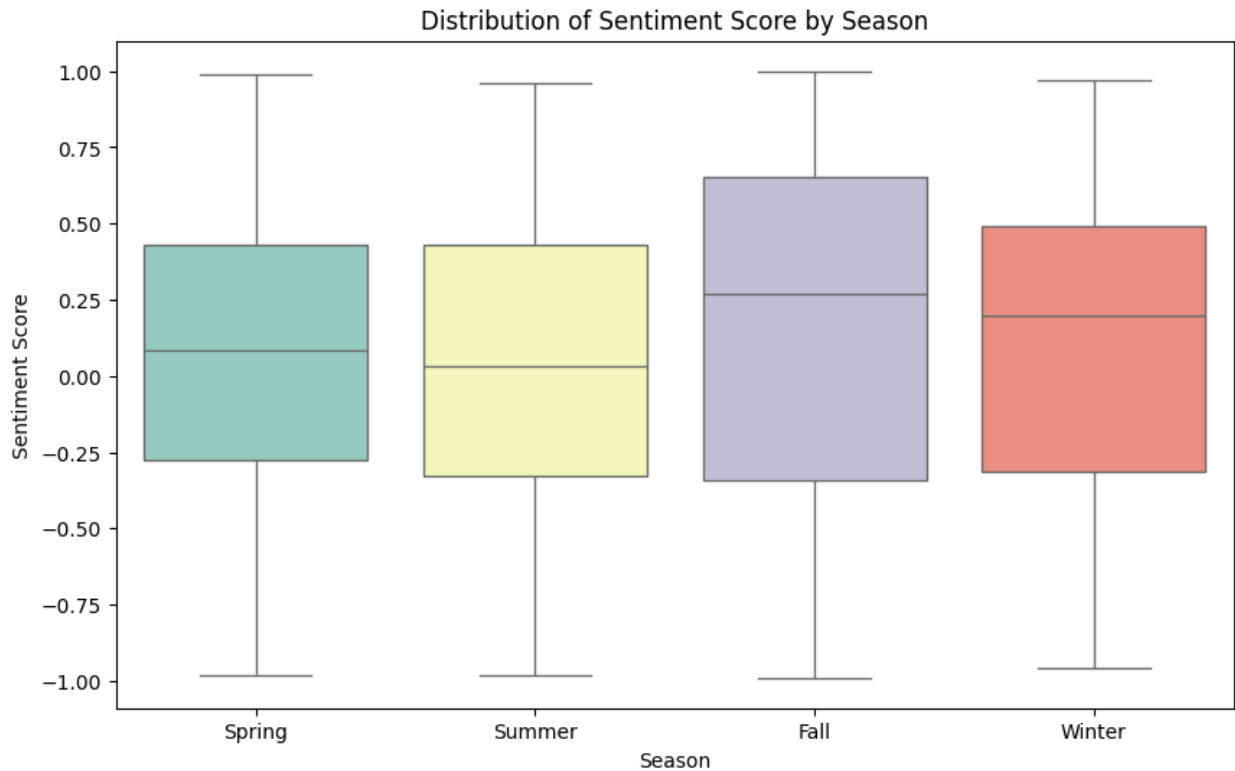
fig = px.scatter(df, x='sentiment_score', y='created_utc',
color='season', size=abs(df['sentiment_score']),
hover_data=['subreddit', 'label'],
title='Sentiment Score over Time',
labels={'sentiment_score': 'Sentiment Score',
'created_utc': 'Created UTC'})
fig.show()

<ipython-input-4-34f8964ec3a9>:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

sns.boxplot(x='season', y='sentiment_score', data=df,
palette="Set3")

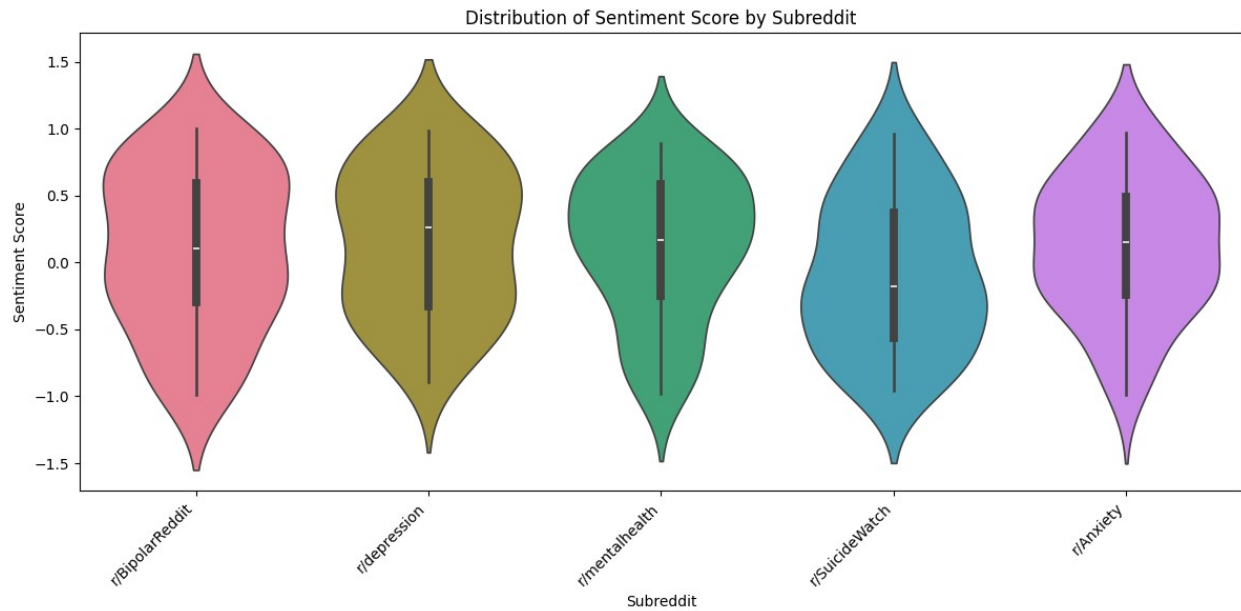
```

```
<ipython-input-4-34f8964ec3a9>:15: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.
```

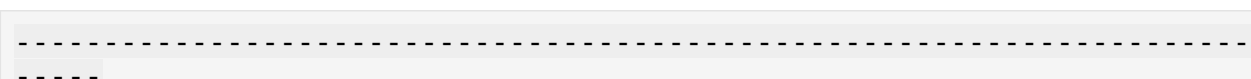
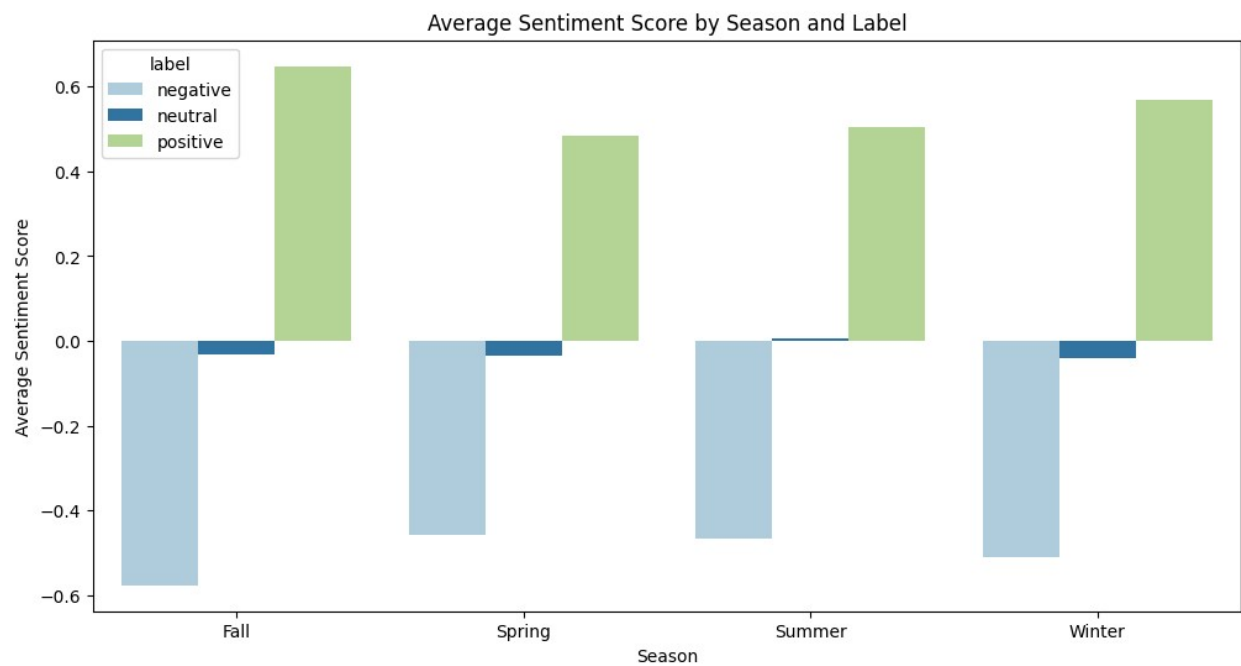
```
sns.violinplot(x='subreddit', y='sentiment_score', data=df, palette="husl")
```



```
<ipython-input-4-34f8964ec3a9>:28: FutureWarning:
```

The `ci` parameter is deprecated. Use `errorbar='sd'` for the same effect.

```
sns.barplot(x='season', y='mean', hue='label',
data=season_label_stats, palette="Paired", ci='sd')
```



```

ValueError                                Traceback (most recent call
last)
<ipython-input-4-34f8964ec3a9> in <cell line: 0>()
    35 # 4. Heatmap of correlations
    36 numerical_cols = ['sentiment_score', 'created_utc']
--> 37 correlation_matrix = df[numerical_cols].corr()
    38
    39 plt.figure(figsize=(8, 6))

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
corr(self, method, min_periods, numeric_only)
   11047         cols = data.columns
   11048         idx = cols.copy()
> 11049         mat = data.to_numpy(dtype=float, na_value=np.nan,
copy=False)
   11050
   11051         if method == "pearson":

/usr/local/lib/python3.11/dist-packages/pandas/core/frame.py in
to_numpy(self, dtype, copy, na_value)
   1991         if dtype is not None:
   1992             dtype = np.dtype(dtype)
-> 1993         result = self._mgr.as_array(dtype=dtype, copy=copy,
na_value=na_value)
   1994         if result.dtype is not dtype:
   1995             result = np.asarray(result, dtype=dtype)

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in as_array(self, dtype, copy, na_value)
   1692             arr.flags.writeable = False
   1693         else:
-> 1694             arr = self._interleave(dtype=dtype,
na_value=na_value)
   1695             # The underlying data was copied within
_interleave, so no need
   1696             # to further copy if copy=True or setting na_value

/usr/local/lib/python3.11/dist-packages/pandas/core/internals/managers
.py in _interleave(self, dtype, na_value)
   1751         else:
   1752             arr = blk.get_values(dtype)
-> 1753             result[rl.indexer] = arr
   1754             itemmask[rl.indexer] = 1
   1755

ValueError: could not convert string to float: '2023-05-01 22:25:36'

```

Reasoning: The previous code block failed due to a type error in the correlation calculation and the interactive plot. Correct the data type of the 'created_utc' column to datetime and handle

the 'created_utc' column separately in the correlation calculation. Also, address the warnings about the palette argument in seaborn plots.

```
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

# 1. Box plot of sentiment_score by season
plt.figure(figsize=(10, 6))
sns.boxplot(x='season', y='sentiment_score', data=df, hue='season',
            legend=False, palette="Set3") # Fixed palette warning
plt.title('Distribution of Sentiment Score by Season')
plt.xlabel('Season')
plt.ylabel('Sentiment Score')
plt.show()

# 2. Violin plot of sentiment_score by subreddit
plt.figure(figsize=(12, 6))
sns.violinplot(x='subreddit', y='sentiment_score', data=df,
              hue='subreddit', legend=False, palette="husl") # Fixed palette warning
plt.title('Distribution of Sentiment Score by Subreddit')
plt.xlabel('Subreddit')
plt.ylabel('Sentiment Score')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# 3. Bar plot of average sentiment_score by season and label
season_label_stats = df.groupby(['season', 'label'])
['sentiment_score'].agg(['mean', 'std'])
season_label_stats = season_label_stats.reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(x='season', y='mean', hue='label',
            data=season_label_stats, palette="Paired", errorbar=('sd')) # Fixed ci warning
plt.title('Average Sentiment Score by Season and Label')
plt.xlabel('Season')
plt.ylabel('Average Sentiment Score')
plt.show()

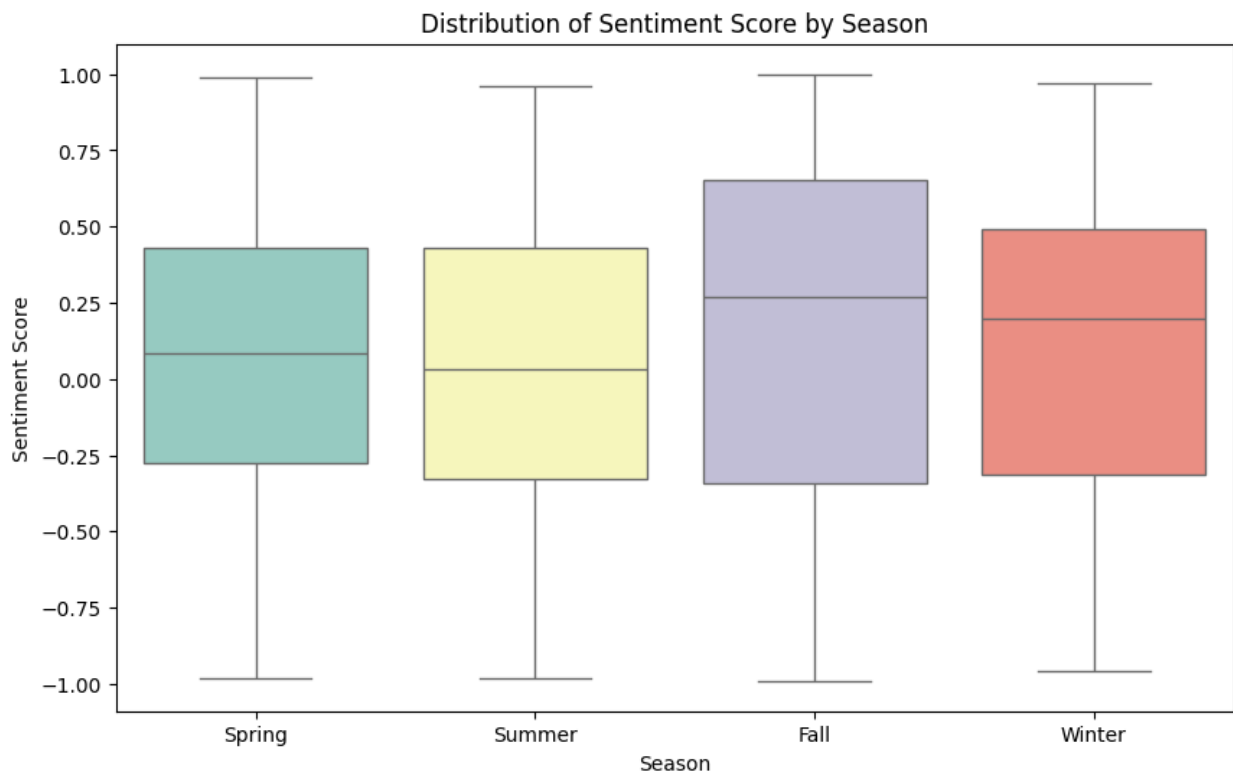
# 4. Heatmap of correlations
# Exclude 'created_utc' from correlation calculation because it's a timestamp
numerical_cols = ['sentiment_score']
correlation_matrix = df[numerical_cols].corr()

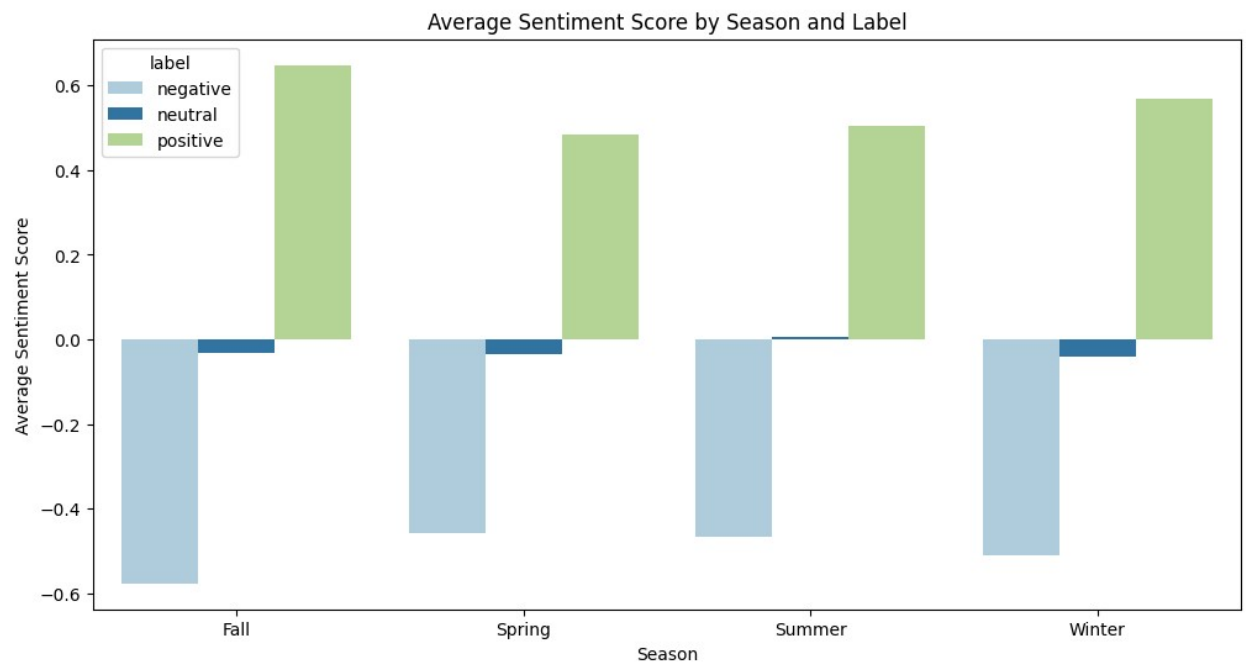
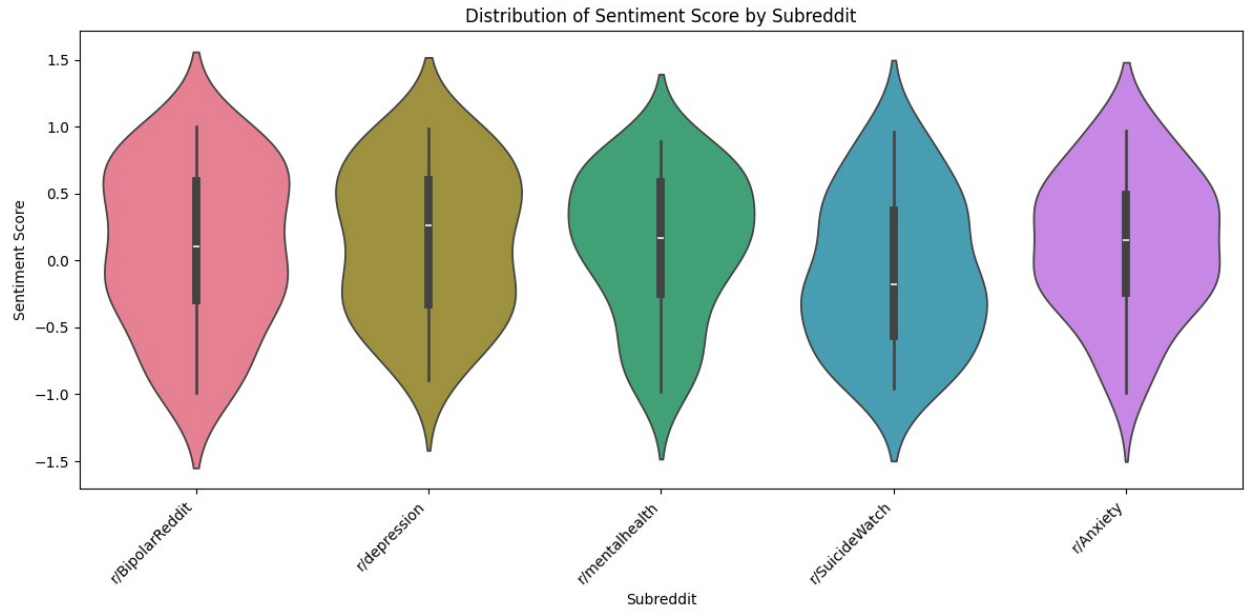
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Matrix of Numerical Features')
```

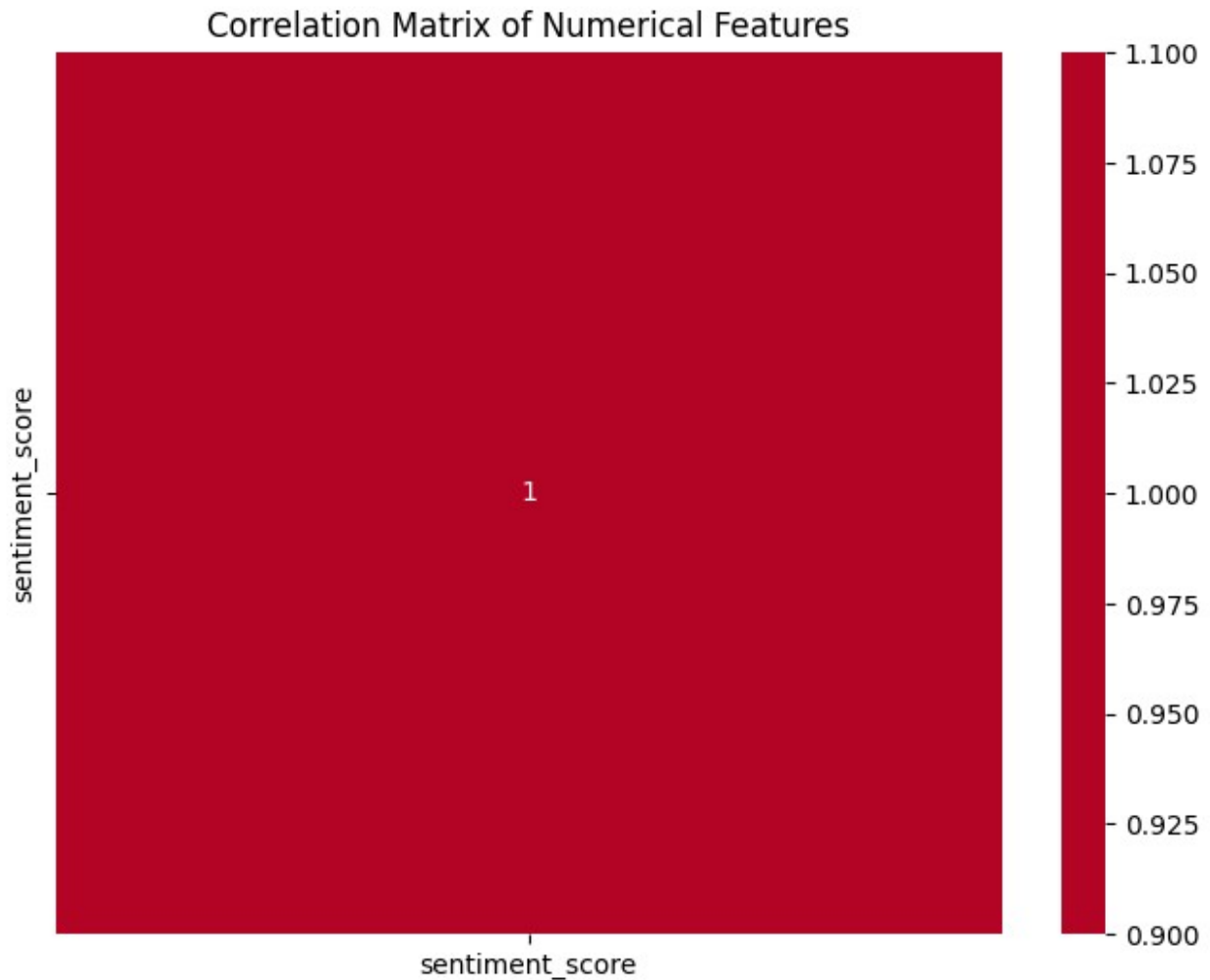
```
plt.show()

# 5. Interactive scatter plot
df['created_utc'] = pd.to_datetime(df['created_utc'])

fig = px.scatter(df, x='sentiment_score', y='created_utc',
color='season', size=abs(df['sentiment_score']),
                hover_data=['subreddit', 'label'],
                title='Sentiment Score over Time',
                labels={'sentiment_score': 'Sentiment Score',
'created_utc': 'Created UTC'})
fig.show()
```







Summary:

Q&A

No questions were asked in the task.

Data Analysis Key Findings

- The dataset contains 200 rows and 8 columns with no missing values.
- The average sentiment score is 0.076, with a standard deviation of 0.555.
- Sentiment score distributions vary across seasons, subreddits, and sentiment labels. For example, 'Spring' has the highest count (58) in the 'season' column, while 'r/BipolarReddit' and 'r/mentalhealth' have the highest counts (44 each) in the 'subreddit' column. The 'positive' label is the most frequent (101 counts) in the 'label' column.
- Detailed descriptive statistics (mean, median, standard deviation, and quantiles) of sentiment scores were calculated for each season, subreddit, and sentiment label combination.
- A pivot table summarizes the mean and median sentiment scores across seasons, subreddits, and labels, revealing potential interactions between these variables.

Insights or Next Steps

- Investigate the significant variations in sentiment scores across different seasons and subreddits to understand the underlying reasons.
- Explore potential correlations between sentiment scores and other relevant factors not included in the initial analysis.