Appendix

Social Media Usage and Emotional Wellbeing

Social media has become deeply embedded into society and our personal lives, impacting our everyday activities and emotional wellbeing. This analysis explores a dataset that has captured social media engagement and the users' prevailing emotional state. The objective of this research is to understand the relationship between social media habits and emotional well-being.

This notebook will include the following sections:

- Data Cleaning / Preparation
- Exploratory Data Analysis
- Model Selection
- Model Analysis

Section 1: Data Cleaning / Preparation

- Load libraries and data
- Understand the data with descriptive statistics
- Locate and address any missing values
-

In [5]:
```
# Import libraries for data analysis, visualization, math calculation

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import scipy.stats as stats
from itertools import combinations
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from sklearn.metrics import classification_report

from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
```

In [6]:
```
social_df = pd.read_csv("https://raw.githubusercontent.com/gurlv/SocialMediaDataset
```

In [7]:
```
print(social_df.head(n=20))
```

```
    User_ID  Age     Gender   Platform  Daily_Usage_Minutes  Posts_Per_Day  \
0         1   25     Female  Instagram                  120              3
1         2   30       Male    Twitter                   90              5
2         3   22  Non-binary   Facebook                  60              2
3         4   28     Female  Instagram                  200              8
4         5   33       Male   LinkedIn                   45              1
5         6   21       Male  Instagram                  150              4
6         7   27     Female    Twitter                   85              3
7         8   24  Non-binary   Facebook                 110              6
8         9   29     Female   LinkedIn                   55              2
9        10   31       Male  Instagram                  170              5
10       11   23     Female    Twitter                   75              4
11       12   26  Non-binary   Facebook                  95              3
12       13   34       Male   LinkedIn                   65              1
13       14   22     Female  Instagram                  180              7
14       15   28       Male    Twitter                  100              6
15       16   21  Non-binary   Facebook                  40              1
16       17   35     Female  Instagram                  125              4
17       18   27       Male    Twitter                   90              3
18       19   23  Non-binary   LinkedIn                  50              1
19       20   32     Female  Instagram                  140              5

    Likes_Received_Per_Day  Comments_Received_Per_Day  Messages_Sent_Per_Day  \
0                       45                         10                     12
1                       20                         25                     30
2                       15                          5                     20
3                      100                         30                     50
4                        5                          2                     10
5                       60                         15                     25
6                       30                         10                     18
7                       25                         12                     22
8                       10                          3                      8
9                       80                         20                     35
10                      35                          7                     20
11                      20                         10                     18
12                      12                          4                     15
13                      90                         25                     40
14                      40                         23                     28
15                       5                          2                     10
16                      55                         18                     30
17                      33                         15                     25
18                       8                          3                     12
19                      70                         22                     33

   Dominant_Emotion
0         Happiness
1             Anger
2           Neutral
3           Anxiety
4           Boredom
5         Happiness
6             Anger
7           Sadness
8           Neutral
9         Happiness
10          Anxiety
```

```
11         Sadness
12         Boredom
13       Happiness
14           Anger
15         Neutral
16         Anxiety
17         Sadness
18         Neutral
19       Happiness
```

In [8]:
```
print(social_df.count())
```

```
User_ID                       1000
Age                           1000
Gender                        1000
Platform                      1000
Daily_Usage_Minutes           1000
Posts_Per_Day                 1000
Likes_Received_Per_Day        1000
Comments_Received_Per_Day     1000
Messages_Sent_Per_Day         1000
Dominant_Emotion              1000
dtype: int64
```

In [9]:
```
print(social_df.columns)
```

```
Index(['User_ID', 'Age', 'Gender', 'Platform', 'Daily_Usage_Minutes',
       'Posts_Per_Day', 'Likes_Received_Per_Day', 'Comments_Received_Per_Day',
       'Messages_Sent_Per_Day', 'Dominant_Emotion'],
      dtype='object')
```

In [10]:
```
print(social_df.dtypes)
```

```
User_ID                        int64
Age                            int64
Gender                        object
Platform                      object
Daily_Usage_Minutes            int64
Posts_Per_Day                  int64
Likes_Received_Per_Day         int64
Comments_Received_Per_Day      int64
Messages_Sent_Per_Day          int64
Dominant_Emotion              object
dtype: object
```

In [11]:
```
# Identify missing values
missing_values = social_df.isnull().sum()

# Print the missing values count for each column
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
User_ID                     0
Age                         0
Gender                      0
Platform                    0
Daily_Usage_Minutes         0
Posts_Per_Day               0
Likes_Received_Per_Day      0
Comments_Received_Per_Day   0
Messages_Sent_Per_Day       0
Dominant_Emotion            0
dtype: int64
```

In [12]:
```
# Get a list of the categorical columns and all unique values
emotion_list = social_df['Dominant_Emotion'].unique()
print("Emotions:", emotion_list)

gender_list = social_df['Gender'].unique()
print("Gender:", gender_list)

platform_list = social_df['Platform'].unique()
print("Platform:", platform_list)
```

```
Emotions: ['Happiness' 'Anger' 'Neutral' 'Anxiety' 'Boredom' 'Sadness']
Gender: ['Female' 'Male' 'Non-binary']
Platform: ['Instagram' 'Twitter' 'Facebook' 'LinkedIn' 'Whatsapp' 'Telegram'
 'Snapchat']
```

In [13]:
```
# Group the data by 'Platform' column
grouped_data = social_df.groupby('Platform')['Gender']

# Calculate descriptive statistics for each group
group_stats = grouped_data.describe()
print(group_stats)
```

```
          count unique         top freq
Platform
Facebook    190      3  Non-binary  140
Instagram   250      3      Female  160
LinkedIn    120      3        Male   50
Snapchat     80      2  Non-binary   50
Telegram     80      2        Male   60
Twitter     200      3        Male  110
Whatsapp     80      2      Female   60
```

In [14]:
```
# Group the data by 'Platform' column
grouped_data = social_df.groupby('Platform')
# Calculate descriptive statistics for each group
group_stats = grouped_data.describe()
print(group_stats)
```

```
          User_ID                                                          \
            count        mean         std   min     25%    50%     75%
Platform
Facebook    190.0  496.684211  289.397852   3.0  245.00  500.0  743.00
Instagram   250.0  492.040000  289.064198   1.0  239.75  498.0  738.50
LinkedIn    120.0  479.500000  288.860264   5.0  230.00  480.0  728.00
Snapchat     80.0  529.000000  289.371208  58.0  280.50  529.0  777.50
Telegram     80.0  528.000000  289.371208  57.0  279.50  528.0  776.50
Twitter     200.0  494.300000  289.347229   2.0  241.75  499.0  739.25
Whatsapp     80.0  527.000000  289.371208  56.0  278.50  527.0  775.50

                      Age             ... Comments_Received_Per_Day        \
             max count       mean     ...                       75%   max
Platform                              ...
Facebook    997.0  190.0  26.263158  ...                      12.0  16.0
Instagram   995.0  250.0  28.080000  ...                      30.0  40.0
LinkedIn    955.0  120.0  29.833333  ...                       6.0   8.0
Snapchat   1000.0   80.0  25.750000  ...                      18.0  20.0
Telegram    999.0   80.0  28.125000  ...                      16.0  20.0
Twitter     996.0  200.0  26.700000  ...                      20.0  30.0
Whatsapp    998.0   80.0  28.375000  ...                      18.0  25.0

          Messages_Sent_Per_Day                                                \
                          count       mean       std   min   25%   50%    75%
Platform
Facebook                  190.0  16.694737  4.606353  10.0  12.0  18.0  20.00
Instagram                 250.0  33.508000  6.368372  12.0  30.0  31.0  38.00
LinkedIn                  120.0  12.558333  2.336398   8.0  10.0  12.0  14.25
Snapchat                   80.0  21.875000  5.746490  12.0  18.0  21.0  27.25
Telegram                   80.0  21.875000  5.042541  12.0  19.5  22.0  25.75
Twitter                   200.0  21.170000  4.198720  10.0  18.0  22.0  24.00
Whatsapp                   80.0  22.125000  2.729933  18.0  20.0  21.5  25.00


            max
Platform
Facebook   25.0
Instagram  50.0
LinkedIn   17.0
Snapchat   30.0
Telegram   28.0
Twitter    30.0
Whatsapp   26.0

[7 rows x 56 columns]
```

In [15]:
```
print(social_df.dtypes)
```

```
User_ID                         int64
Age                             int64
Gender                         object
Platform                       object
Daily_Usage_Minutes             int64
Posts_Per_Day                   int64
Likes_Received_Per_Day          int64
Comments_Received_Per_Day       int64
Messages_Sent_Per_Day           int64
Dominant_Emotion               object
dtype: object
```

Section 2: Exploratory Data Analysis

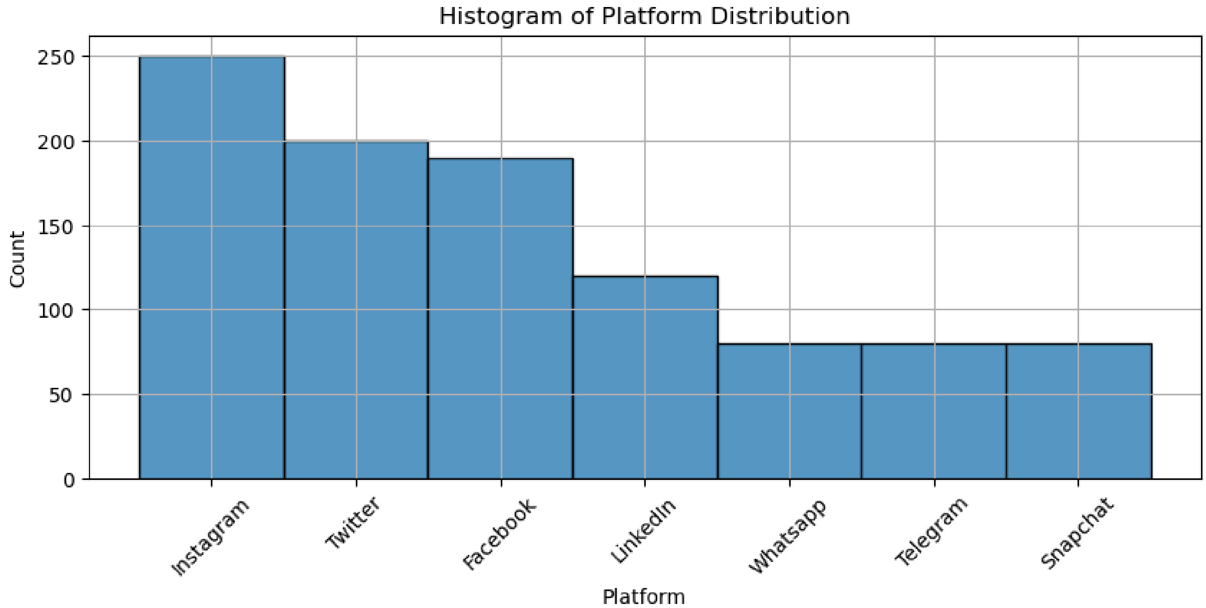
Dominant Emotion Per Platform

In [16]:

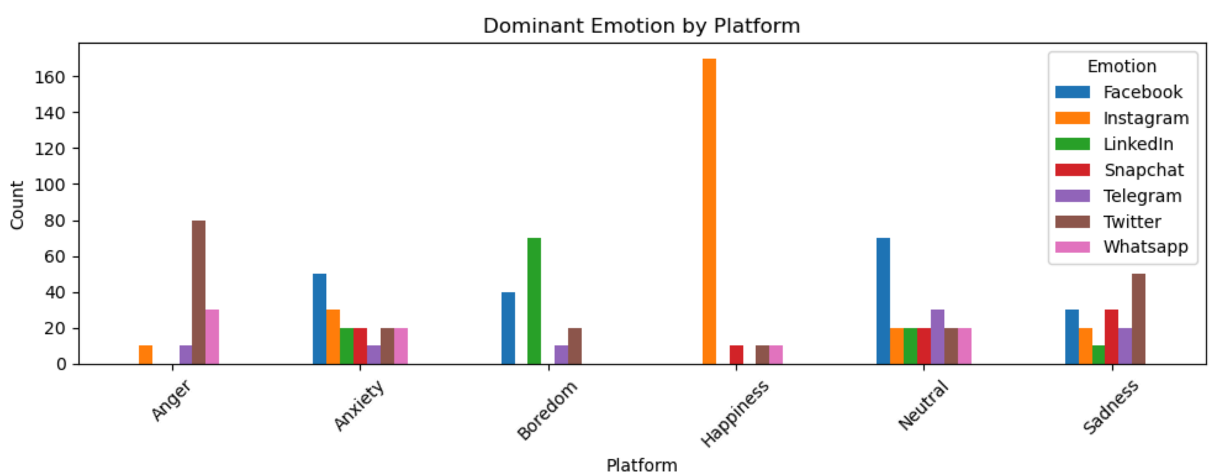
```
# Histogram for the distribution of platforms
plt.figure(figsize=(10, 4))
sns.histplot(data=social_df, x='Platform', discrete=True, kde=False)
plt.title('Histogram of Platform Distribution')
plt.xlabel('Platform')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()


# Group the data by race
grouped_data = social_df.groupby('Platform')['Dominant_Emotion']

# Calculate descriptive statistics, I will be using this grouping method for my fin
descriptive_stats = grouped_data.describe()

# Contingency table for Dominant Emotion by Platform
contingency_table_platform_emotion = pd.crosstab(social_df['Dominant_Emotion'], soc

# Perform the Chi-Square test - chi tests are used to determine whether there is a
chi2, p, dof, ex = stats.chi2_contingency(contingency_table_platform_emotion)

# Bar chart Dominant Emotion by Platform
contingency_table_platform_emotion.plot(kind='bar', figsize=(10, 4))

plt.title('Dominant Emotion by Platform')
plt.xlabel('Platform')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Display results
print("\n", descriptive_stats)
print("\nContingency Table:\n")
print(contingency_table_platform_emotion, "\n")
```

```
# Print results
print(f"Chi-Square Test: chi2 = {chi2}, p-value = {p}, degrees of freedom = {dof}",
# Currently Chi-Square is incorrect
```

```
C:\Users\infin\anaconda3\envs\Pandas\Lib\site-packages\seaborn\_oldcore.py:1119: Fut
ureWarning: use_inf_as_na option is deprecated and will be removed in a future versi
on. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



```
           count unique       top freq
Platform
Facebook    190      4    Neutral   70
Instagram   250      5  Happiness  170
LinkedIn    120      4    Boredom   70
Snapchat     80      4    Sadness   30
Telegram     80      5    Neutral   30
Twitter     200      6      Anger   80
Whatsapp     80      4      Anger   30

Contingency Table:

Platform          Facebook  Instagram  LinkedIn  Snapchat  Telegram  Twitter  \
Dominant_Emotion
Anger                    0         10         0         0        10       80
Anxiety                 50         30        20        20        10       20
Boredom                 40          0        70         0        10       20
Happiness                0        170         0        10         0       10
Neutral                 70         20        20        20        30       20
Sadness                 30         20        10        30        20       50

Platform          Whatsapp
Dominant_Emotion
Anger                   30
Anxiety                 20
Boredom                  0
Happiness               10
Neutral                 20
Sadness                  0

Chi-Square Test: chi2 = 1003.9059462853968, p-value = 7.68647651533505e-192, degrees
of freedom = 30
```



Conduct T-Tests for Dominant Emotion Per Platform

In [17]:
```python
# Get unique dominant emotions and platforms
emotions = social_df['Dominant_Emotion'].unique()
platforms = social_df['Platform'].unique()

# Function to perform pairwise t-tests - need to do this form of testing to
```

```python
def pairwise_t_tests(data, group_col, value_col):
    results = {}
    groups = data[group_col].unique()
    for group1, group2 in combinations(groups, 2):
        group1_data = data[data[group_col] == group1][value_col].dropna()
        group2_data = data[data[group_col] == group2][value_col].dropna()
        t_stat, p_val = stats.ttest_ind(group1_data, group2_data)
        results[(group1, group2)] = (t_stat, p_val)
    return results

# Perform t-tests for each emotion
t_test_results = {}
for emotion in emotions:
    emotion_data = social_df[social_df['Dominant_Emotion'] == emotion]
    t_test_results[emotion] = pairwise_t_tests(emotion_data, 'Platform', 'Age')

# Display the results
for emotion, results in t_test_results.items():
    print(f"Results for {emotion}:")
    for platforms, (t_stat, p_val) in results.items():
        print(f"  {platforms}: t-statistic = {t_stat:.4f}, p-value = {p_val:.4f}")
    print()
```

```
Results for Happiness:
  ('Instagram', 'Twitter'): t-statistic = -0.2485, p-value = 0.8040
  ('Instagram', 'Snapchat'): t-statistic = 3.1309, p-value = 0.0020
  ('Instagram', 'Whatsapp'): t-statistic = -6.1624, p-value = 0.0000
  ('Twitter', 'Snapchat'): t-statistic = inf, p-value = 0.0000
  ('Twitter', 'Whatsapp'): t-statistic = -inf, p-value = 0.0000
  ('Snapchat', 'Whatsapp'): t-statistic = -inf, p-value = 0.0000

Results for Anger:
  ('Twitter', 'Whatsapp'): t-statistic = -1.0603, p-value = 0.2914
  ('Twitter', 'Instagram'): t-statistic = -9.2790, p-value = 0.0000
  ('Twitter', 'Telegram'): t-statistic = -0.8754, p-value = 0.3838
  ('Whatsapp', 'Instagram'): t-statistic = -4.9425, p-value = 0.0000
  ('Whatsapp', 'Telegram'): t-statistic = 0.0000, p-value = 1.0000
  ('Instagram', 'Telegram'): t-statistic = inf, p-value = 0.0000

Results for Neutral:
  ('Facebook', 'LinkedIn'): t-statistic = -1.9930, p-value = 0.0494
  ('Facebook', 'Twitter'): t-statistic = 1.9092, p-value = 0.0595
  ('Facebook', 'Instagram'): t-statistic = -2.7217, p-value = 0.0078
  ('Facebook', 'Telegram'): t-statistic = -3.9776, p-value = 0.0001
  ('Facebook', 'Whatsapp'): t-statistic = -5.4467, p-value = 0.0000
  ('Facebook', 'Snapchat'): t-statistic = 0.1533, p-value = 0.8785
  ('LinkedIn', 'Twitter'): t-statistic = 5.0162, p-value = 0.0000
  ('LinkedIn', 'Instagram'): t-statistic = -0.7475, p-value = 0.4593
  ('LinkedIn', 'Telegram'): t-statistic = -1.4261, p-value = 0.1603
  ('LinkedIn', 'Whatsapp'): t-statistic = -3.6268, p-value = 0.0008
  ('LinkedIn', 'Snapchat'): t-statistic = 2.0548, p-value = 0.0468
  ('Twitter', 'Instagram'): t-statistic = -3.9035, p-value = 0.0004
  ('Twitter', 'Telegram'): t-statistic = -5.0138, p-value = 0.0000
  ('Twitter', 'Whatsapp'): t-statistic = -13.7434, p-value = 0.0000
  ('Twitter', 'Snapchat'): t-statistic = -2.1498, p-value = 0.0380
  ('Instagram', 'Telegram'): t-statistic = -0.4810, p-value = 0.6327
  ('Instagram', 'Whatsapp'): t-statistic = -1.6189, p-value = 0.1137
  ('Instagram', 'Snapchat'): t-statistic = 2.2426, p-value = 0.0308
  ('Telegram', 'Whatsapp'): t-statistic = -1.2212, p-value = 0.2280
  ('Telegram', 'Snapchat'): t-statistic = 3.1375, p-value = 0.0029
  ('Whatsapp', 'Snapchat'): t-statistic = 6.0447, p-value = 0.0000

Results for Anxiety:
  ('Instagram', 'Twitter'): t-statistic = 2.8898, p-value = 0.0058
  ('Instagram', 'Facebook'): t-statistic = 1.7359, p-value = 0.0865
  ('Instagram', 'LinkedIn'): t-statistic = 0.6972, p-value = 0.4890
  ('Instagram', 'Whatsapp'): t-statistic = 4.4740, p-value = 0.0000
  ('Instagram', 'Snapchat'): t-statistic = 2.9626, p-value = 0.0047
  ('Instagram', 'Telegram'): t-statistic = 2.1301, p-value = 0.0397
  ('Twitter', 'Facebook'): t-statistic = -2.4653, p-value = 0.0162
  ('Twitter', 'LinkedIn'): t-statistic = -2.7033, p-value = 0.0102
  ('Twitter', 'Whatsapp'): t-statistic = 1.4184, p-value = 0.1642
  ('Twitter', 'Snapchat'): t-statistic = -0.5987, p-value = 0.5529
  ('Twitter', 'Telegram'): t-statistic = -0.4364, p-value = 0.6659
  ('Facebook', 'LinkedIn'): t-statistic = -0.9213, p-value = 0.3601
  ('Facebook', 'Whatsapp'): t-statistic = 4.6833, p-value = 0.0000
  ('Facebook', 'Snapchat'): t-statistic = 2.3257, p-value = 0.0230
  ('Facebook', 'Telegram'): t-statistic = 1.6892, p-value = 0.0965
  ('LinkedIn', 'Whatsapp'): t-statistic = 4.8358, p-value = 0.0000
```

```
  ('LinkedIn', 'Snapchat'): t-statistic = 3.8987, p-value = 0.0004
  ('LinkedIn', 'Telegram'): t-statistic = 3.0551, p-value = 0.0049
  ('Whatsapp', 'Snapchat'): t-statistic = -2.7568, p-value = 0.0089
  ('Whatsapp', 'Telegram'): t-statistic = -2.0367, p-value = 0.0512
  ('Snapchat', 'Telegram'): t-statistic = 0.0000, p-value = 1.0000

Results for Boredom:
  ('LinkedIn', 'Telegram'): t-statistic = 4.9568, p-value = 0.0000
  ('LinkedIn', 'Facebook'): t-statistic = 4.8775, p-value = 0.0000
  ('LinkedIn', 'Twitter'): t-statistic = -3.4754, p-value = 0.0008
  ('Telegram', 'Facebook'): t-statistic = -9.2952, p-value = 0.0000
  ('Telegram', 'Twitter'): t-statistic = -27.4955, p-value = 0.0000
  ('Facebook', 'Twitter'): t-statistic = -21.5407, p-value = 0.0000

Results for Sadness:
  ('Facebook', 'Twitter'): t-statistic = 2.5249, p-value = 0.0136
  ('Facebook', 'Instagram'): t-statistic = -1.4639, p-value = 0.1498
  ('Facebook', 'LinkedIn'): t-statistic = -18.5253, p-value = 0.0000
  ('Facebook', 'Snapchat'): t-statistic = -2.0922, p-value = 0.0408
  ('Facebook', 'Telegram'): t-statistic = -26.3363, p-value = 0.0000
  ('Twitter', 'Instagram'): t-statistic = -2.7764, p-value = 0.0071
  ('Twitter', 'LinkedIn'): t-statistic = -9.1320, p-value = 0.0000
  ('Twitter', 'Snapchat'): t-statistic = -3.8683, p-value = 0.0002
  ('Twitter', 'Telegram'): t-statistic = -12.9463, p-value = 0.0000
  ('Instagram', 'LinkedIn'): t-statistic = -2.0367, p-value = 0.0512
  ('Instagram', 'Snapchat'): t-statistic = 0.2473, p-value = 0.8057
  ('Instagram', 'Telegram'): t-statistic = -2.9059, p-value = 0.0061
  ('LinkedIn', 'Snapchat'): t-statistic = 4.0475, p-value = 0.0002
  ('LinkedIn', 'Telegram'): t-statistic = nan, p-value = nan
  ('Snapchat', 'Telegram'): t-statistic = -5.7541, p-value = 0.0000
```

```
C:\Users\infin\anaconda3\envs\Pandas\Lib\site-packages\scipy\stats\_axis_nan_policy.
py:531: RuntimeWarning: Precision loss occurred in moment calculation due to catastr
ophic cancellation. This occurs when the data are nearly identical. Results may be u
nreliable.
  res = hypotest_fun_out(*samples, **kwds)
```

Facebook Data

- Age Hypotheses:
  - Null Hypothesis (H0): There is no significant relationship between age and dominant emotion on the Facebook platform.
  - Alternative Hypothesis (H1): There is a significant relationship between age and dominant emotion on the Facebook platform.
- Usage Hypotheses:
  - Null Hypothesis (H0): There is no significant relationship between daily usage and dominant emotion on the Facebook platform.
  - Alternative Hypothesis (H1): There is a significant relationship between daily usage and dominant emotion on the Facebook platform.

Filter to Facebook Platform

In [18]:
```
# Filter the data for the Facebook platform
facebook_subset = social_df[social_df['Platform'] == 'Facebook']

facebook_subset.describe()
```

Out[18]:

	User_ID	Age	Daily_Usage_Minutes	Posts_Per_Day	Likes_Received_Per_Day
count	190.000000	190.000000	190.000000	190.000000	190.000000
mean	496.684211	26.263158	72.105263	1.947368	19.726316
std	289.397852	3.330826	19.471577	1.148898	7.997939
min	3.000000	21.000000	40.000000	1.000000	5.000000
25%	245.000000	23.000000	60.000000	1.000000	12.000000
50%	500.000000	26.000000	70.000000	2.000000	20.000000
75%	743.000000	29.000000	85.000000	2.000000	27.000000
max	997.000000	33.000000	110.000000	6.000000	35.000000

Describe Targeted/Grouped Data - Dominant Emotion by Age

In [19]:
```
# Descriptive Statistics
cross_reference_age_emotion = facebook_subset.groupby('Dominant_Emotion')['Age'].de
print("Cross-reference of Age with Dominant Emotion on Facebook:")
print(cross_reference_age_emotion)
```

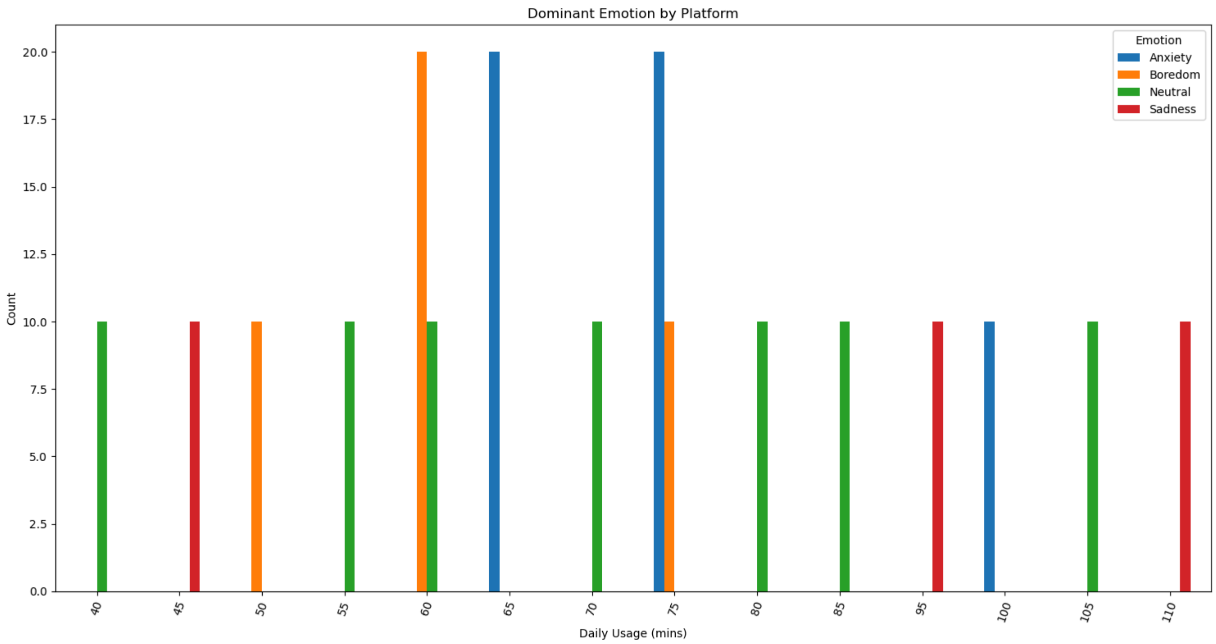
```
Cross-reference of Age with Dominant Emotion on Facebook:
                  count       mean       std   min   25%   50%   75%   max
Dominant_Emotion
Anxiety            50.0  28.400000  2.602981  26.0  26.0  28.0  29.0  33.0
Boredom            40.0  28.000000  1.012739  27.0  27.0  28.0  29.0  29.0
Neutral            70.0  24.142857  3.823216  21.0  21.0  22.0  29.0  31.0
Sadness            30.0  25.333333  0.958927  24.0  24.0  26.0  26.0  26.0
```

Describe Targeted/Grouped Data - Dominant Emotion by Daily Usage

In [20]:
```
# Descriptive Statistics
contingency_emotion_usage = pd.crosstab(facebook_subset['Daily_Usage_Minutes'], fac


cross_reference_emotion_usage = facebook_subset.groupby('Dominant_Emotion')['Daily_
print("Cross-reference of Daily Usage with Dominant Emotion on Twitter:\n")
print(cross_reference_emotion_usage)



# Plot the bar chart
contingency_emotion_usage.plot(kind='bar', figsize=(15, 8))
```

```
plt.title('Dominant Emotion by Platform')
plt.xlabel('Daily Usage (mins)')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=70)
plt.tight_layout()
```

```
Cross-reference of Daily Usage with Dominant Emotion on Twitter:

                   count       mean        std   min   25%   50%     75%    max
Dominant_Emotion
Anxiety             50.0  76.000000  12.936264  65.0  65.0  75.0   75.00  100.0
Boredom             40.0  61.250000   9.040507  50.0  57.5  60.0   63.75   75.0
Neutral             70.0  70.714286  20.041365  40.0  55.0  70.0   85.00  105.0
Sadness             30.0  83.333333  28.263945  45.0  45.0  95.0  110.00  110.0
```

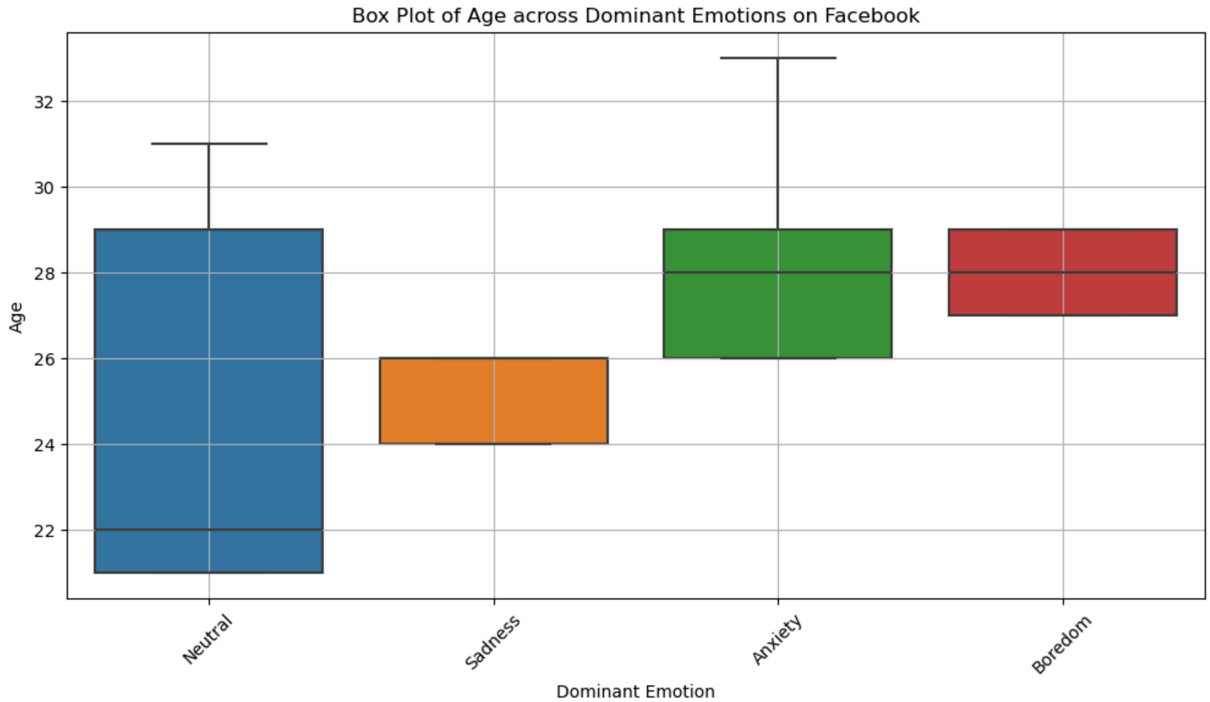


In [ ]:

Box Plot

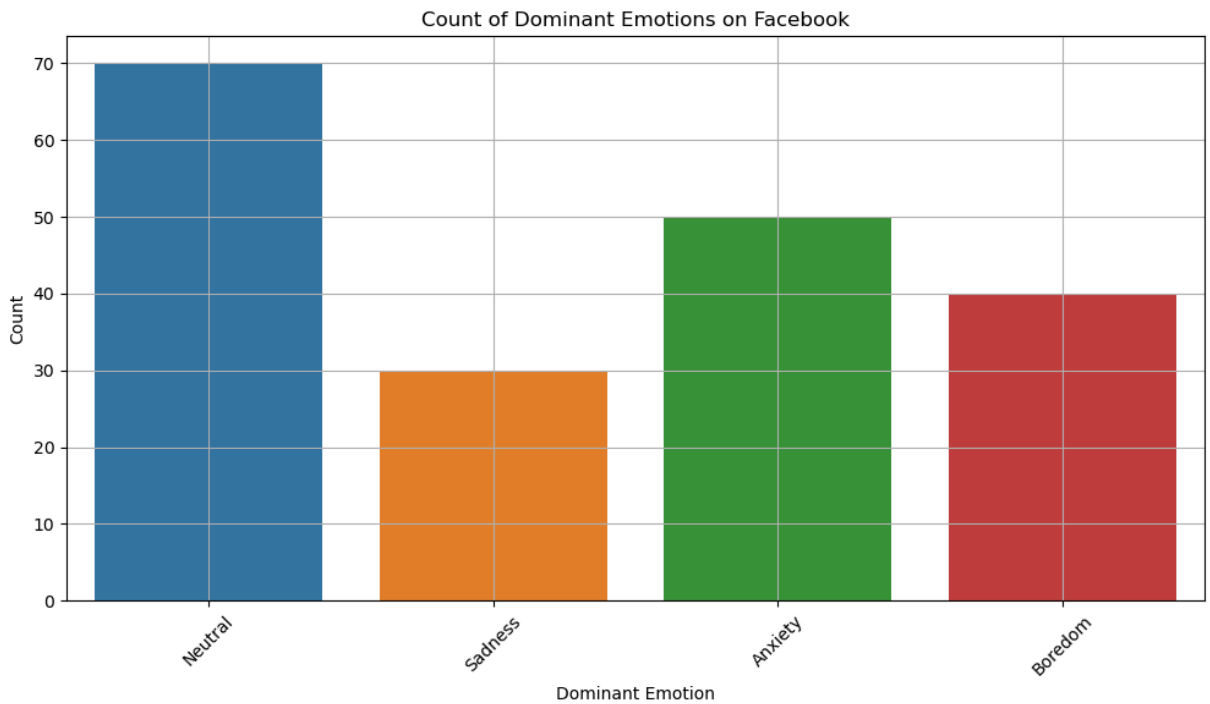
In [21]:

```
# Box Plot of Age for each Dominant Emotion
plt.figure(figsize=(12, 6))
sns.boxplot(data=facebook_subset, x='Dominant_Emotion', y='Age')
plt.title('Box Plot of Age across Dominant Emotions on Facebook')
plt.xlabel('Dominant Emotion')
plt.ylabel('Age')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Bar Plot

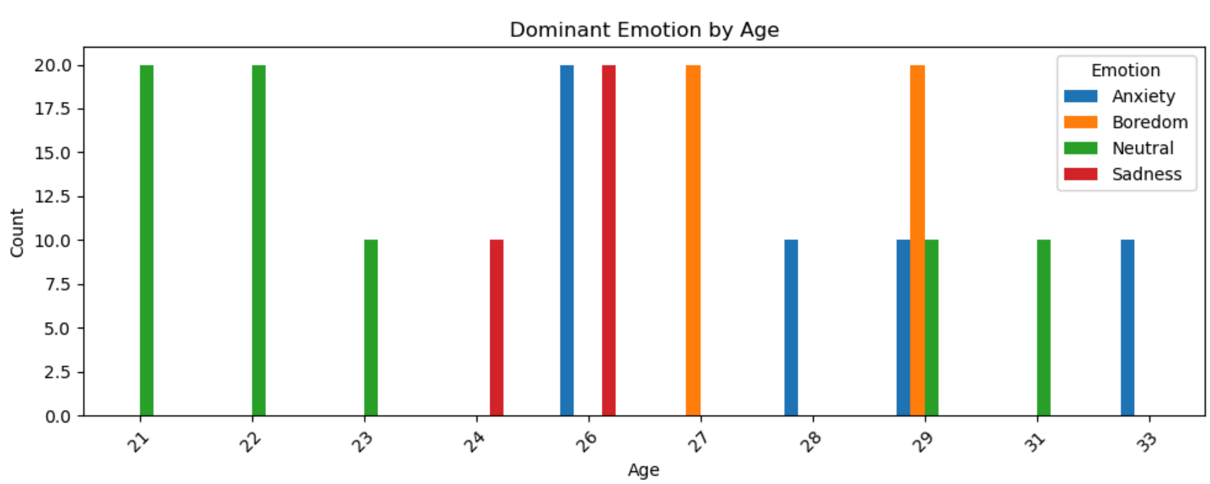
In [22]:
```python
# Bar Plot of the Count of Each Dominant Emotion
plt.figure(figsize=(12, 6))
sns.countplot(data=facebook_subset, x='Dominant_Emotion')
plt.title('Count of Dominant Emotions on Facebook')
plt.xlabel('Dominant Emotion')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Chi-Square Test

In [23]:
```
# Chi-Square Test
# Create a contingency table
contingency_table = pd.crosstab(facebook_subset['Dominant_Emotion'], facebook_subse
contingency_table_flipped = pd.crosstab(facebook_subset['Age'], facebook_subset['Do

# Plot the bar chart
contingency_table_flipped.plot(kind='bar', figsize=(10, 4))

plt.title('Dominant Emotion by Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the results
print("\nContingency Table:")
print(contingency_table_flipped)

# Perform the Chi-Square test
chi2, p, dof, ex = stats.chi2_contingency(contingency_table)
print(f"Chi-Square Test: chi2 = {chi2}, p-value = {p}" , "\n")
```

```
Contingency Table:
Dominant_Emotion  Anxiety  Boredom  Neutral  Sadness
Age
21                      0        0       20        0
22                      0        0       20        0
23                      0        0       10        0
24                      0        0        0       10
26                     20        0        0       20
27                      0       20        0        0
28                     10        0        0        0
29                     10       20       10        0
31                      0        0       10        0
33                     10        0        0        0
Chi-Square Test: chi2 = 372.30952380952385, p-value = 2.1101673144632784e-62
```



T-Tests

In [24]:
```
# T-Test
# Perform pairwise T-tests for age across different dominant emotions
emotions = facebook_subset['Dominant_Emotion'].unique()

# Conduct pairwise T-tests
t_test_results = {}
for emotion1, emotion2 in combinations(emotions, 2):
    group1 = facebook_subset[facebook_subset['Dominant_Emotion'] == emotion1]['Age'
    group2 = facebook_subset[facebook_subset['Dominant_Emotion'] == emotion2]['Age'
    t_stat, p_val = stats.ttest_ind(group1, group2)
    t_test_results[(emotion1, emotion2)] = (t_stat, p_val)

print("Pairwise T-test results:")
for key, value in t_test_results.items():
    print(f"{key}: t-statistic = {value[0]}, p-value = {value[1]}")
```

```
Pairwise T-test results:
('Neutral', 'Sadness'): t-statistic = -1.6785074633458787, p-value = 0.0964342154411
0535
('Neutral', 'Anxiety'): t-statistic = -6.821150743890093, p-value = 4.10648539633776
26e-10
('Neutral', 'Boredom'): t-statistic = -6.24541446843891, p-value = 8.529373399628306
e-09
('Sadness', 'Anxiety'): t-statistic = -6.192542583068983, p-value = 2.58364149082292
15e-08
('Sadness', 'Boredom'): t-statistic = -11.150912838994719, p-value = 5.2747565152123
88e-17
('Anxiety', 'Boredom'): t-statistic = 0.9171145859050562, p-value = 0.36158924216595
99
```

Instagram Data

- Age Hypotheses:
    - Null Hypothesis (H0): There is no significant relationship between age and dominant emotion on the Instagram platform.
    - Alternative Hypothesis (H1): There is a significant relationship between age and dominant emotion on the Instagram platform.
- Usage Hypotheses:
    - Null Hypothesis (H0): There is no significant relationship between daily usage and dominant emotion on the Instagram platform.
    - Alternative Hypothesis (H1): There is a significant relationship between daily usage and dominant emotion on the Instagram platform.

Filter to Instragram Platform

In [25]:
```
# Filter the data for the Facebook platform
instagram_subset = social_df[social_df['Platform'] == 'Instagram']
```

```
print("Instagram Descriptive Data:\n")
instagram_subset.describe()
# Formulate Hypotheses
# H0: There is no significant relationship between age and dominant emotion on the
# H1: There is a significant relationship between age and dominant emotion on the F
```

```
Instagram Descriptive Data:
```

Out[25]:

	User_ID	Age	Daily_Usage_Minutes	Posts_Per_Day	Likes_Received_Per_Day
count	250.000000	250.000000	250.000000	250.000000	250.000000
mean	492.040000	28.080000	153.400000	5.800000	79.272000
std	289.064198	4.259834	22.348104	1.270613	15.317458
min	1.000000	21.000000	115.000000	3.000000	45.000000
25%	239.750000	25.000000	140.000000	5.000000	65.000000
50%	498.000000	28.000000	150.000000	6.000000	80.000000
75%	738.500000	32.000000	170.000000	7.000000	90.000000
max	995.000000	35.000000	200.000000	8.000000	110.000000

Describe Targeted/Grouped Data - Dominant Emotion by Age

In [26]:

```
# Descriptive Statistics
cross_reference_age_emotion = instagram_subset.groupby('Dominant_Emotion')['Age'].d

print("Cross-reference of Age with Dominant Emotion on Instagram:\n")
print(cross_reference_age_emotion)
```

```
Cross-reference of Age with Dominant Emotion on Instagram:

                  count       mean       std   min   25%   50%   75%   max
Dominant_Emotion
Anger              10.0  34.000000  0.000000  34.0  34.0  34.0  34.0  34.0
Anxiety            30.0  29.666667  3.924576  26.0  26.0  28.0  35.0  35.0
Happiness         170.0  27.705882  3.733183  21.0  25.0  28.0  31.0  33.0
Neutral            20.0  27.000000  5.129892  22.0  22.0  27.0  32.0  32.0
Sadness            20.0  27.000000  6.155870  21.0  21.0  27.0  33.0  33.0
```

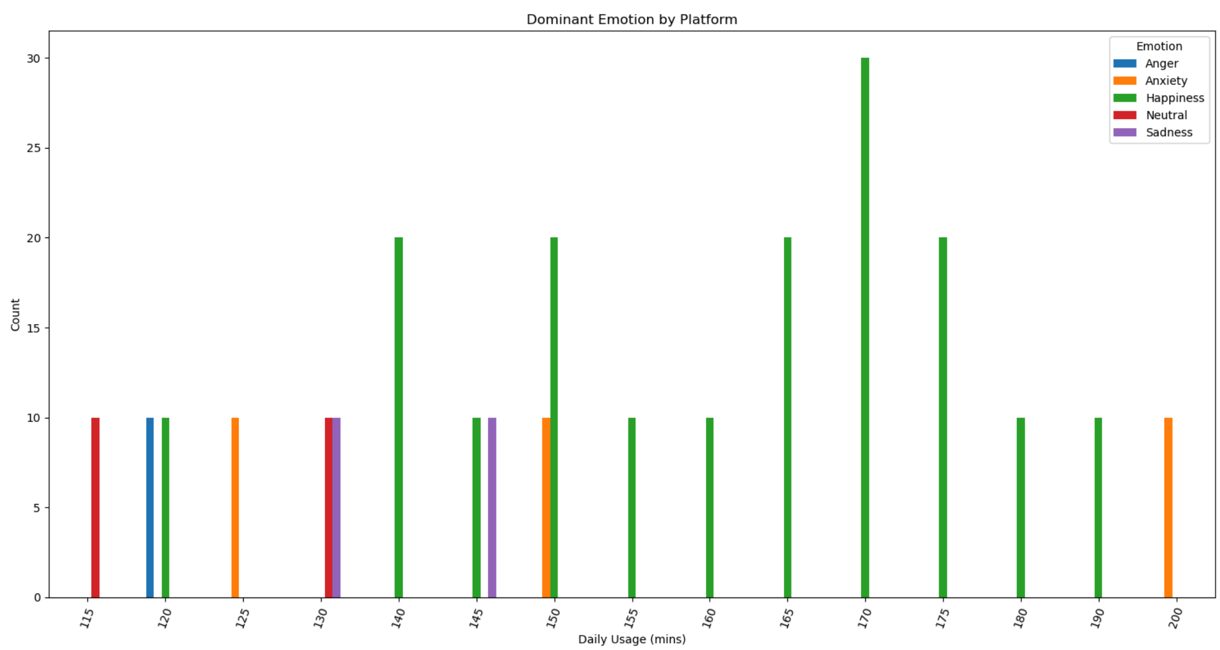
Describe Targeted/Grouped Data - Dominant Emotion by Daily Usage

In [27]:

```
# Descriptive Statistics
contingency_emotion_usage = pd.crosstab(instagram_subset['Daily_Usage_Minutes'], in


cross_reference_emotion_usage = instagram_subset.groupby('Dominant_Emotion')['Daily
print("Cross-reference of Daily Usage with Dominant Emotion on Instagram:\n")
print(cross_reference_emotion_usage)
```

```
# Plot the bar chart
contingency_emotion_usage.plot(kind='bar', figsize=(15, 8))

plt.title('Dominant Emotion by Platform')
plt.xlabel('Daily Usage (mins)')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=70)
plt.tight_layout()
```

```
Cross-reference of Daily Usage with Dominant Emotion on Instagram:

                  count        mean        std    min    25%    50%    75%  \
Dominant_Emotion
Anger              10.0  120.000000   0.000000  120.0  120.0  120.0  120.0
Anxiety            30.0  158.333333  31.713516  125.0  125.0  150.0  200.0
Happiness         170.0  160.000000  17.114304  120.0  150.0  165.0  170.0
Neutral            20.0  122.500000   7.694838  115.0  115.0  122.5  130.0
Sadness            20.0  137.500000   7.694838  130.0  130.0  137.5  145.0

                    max
Dominant_Emotion
Anger             120.0
Anxiety           200.0
Happiness         190.0
Neutral           130.0
Sadness           145.0
```

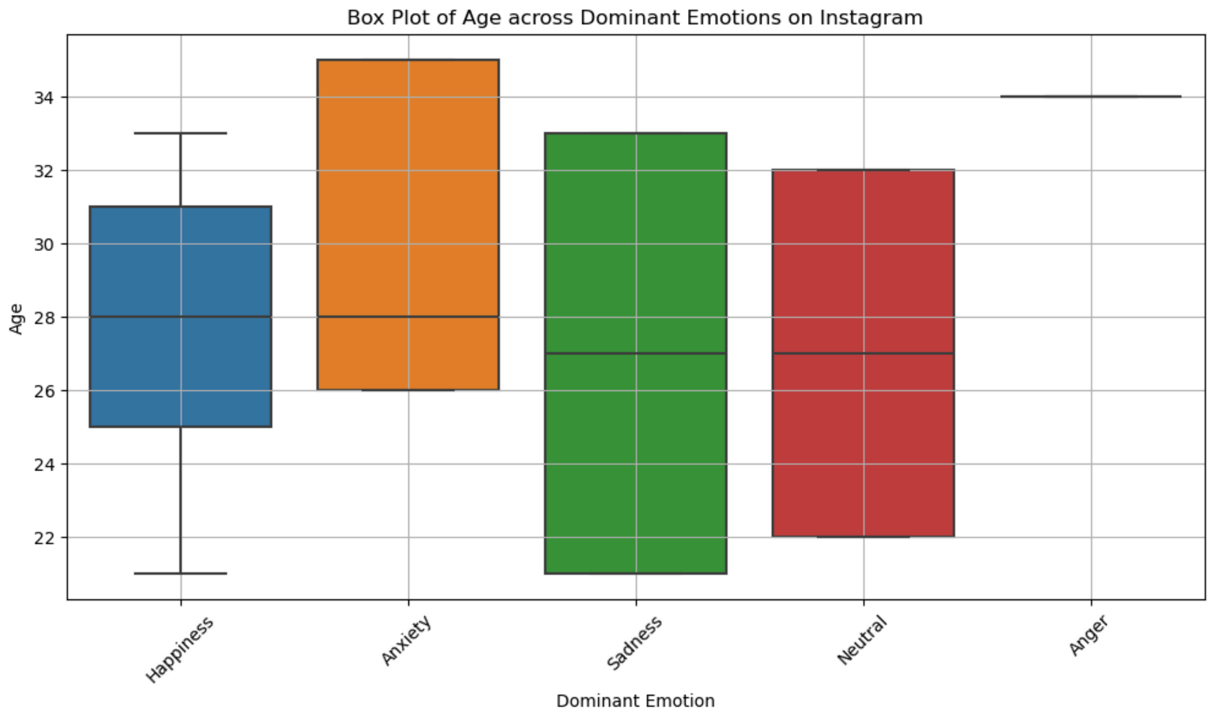


Box Plot

In [28]:

```
# Box Plot of Age for each Dominant Emotion
plt.figure(figsize=(12, 6))
sns.boxplot(data=instagram_subset, x='Dominant_Emotion', y='Age')
plt.title('Box Plot of Age across Dominant Emotions on Instagram')
plt.xlabel('Dominant Emotion')
```

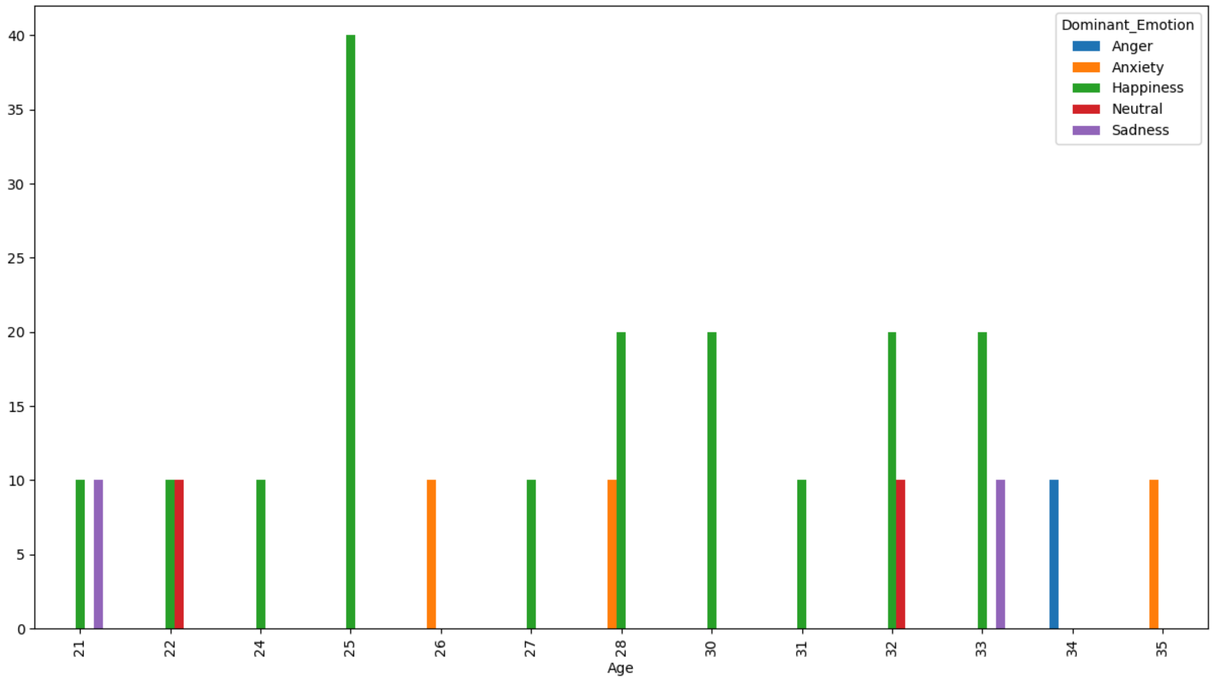
```
plt.ylabel('Age')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

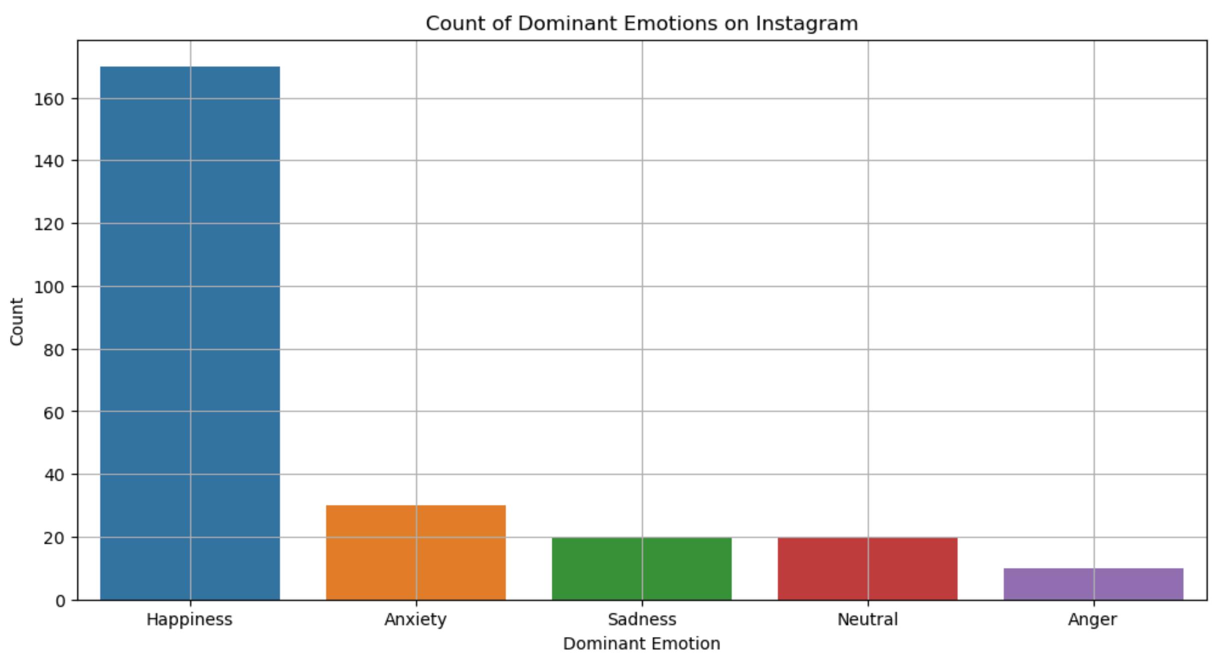


Bar Plot

In [29]:

```
# Bar Plot of the Count of Each Dominant Emotion
contingency_table_flipped = pd.crosstab(instagram_subset['Age'], instagram_subset['
contingency_table_flipped.plot(kind='bar', figsize=(15, 8))

plt.figure(figsize=(12, 6))
sns.countplot(data=instagram_subset, x='Dominant_Emotion')
plt.title('Count of Dominant Emotions on Instagram')
plt.xlabel('Dominant Emotion')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```





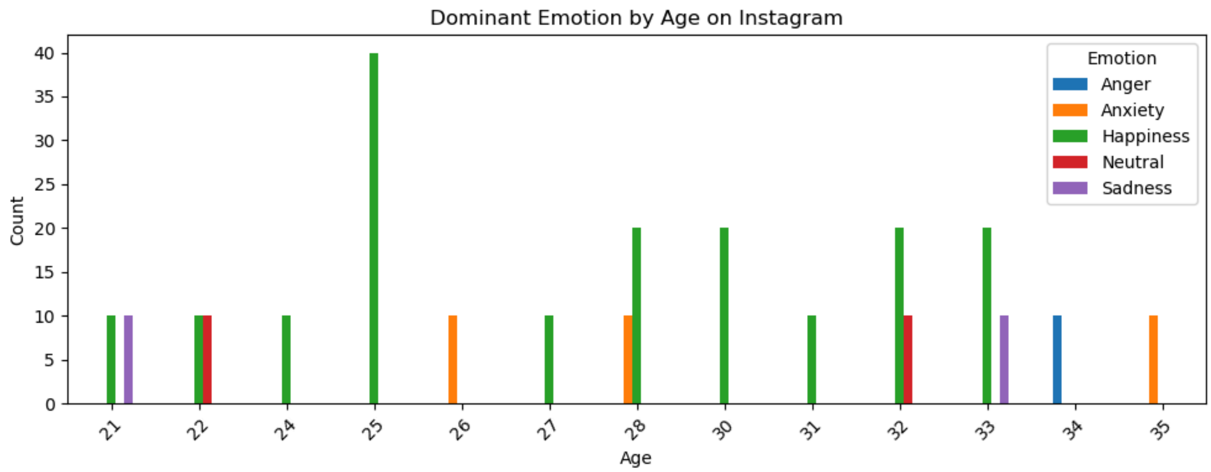
Chi-Square Tests

Age Chi-Square

In [30]:
```
# Create a contingency table
contingency_table_ig = pd.crosstab(instagram_subset['Dominant_Emotion'], instagram_
contingency_table_flipped_ig = pd.crosstab(instagram_subset['Age'], instagram_subse

# Plot the bar chart
contingency_table_flipped_ig.plot(kind='bar', figsize=(10, 4))

plt.title('Dominant Emotion by Age on Instagram')
plt.xlabel('Age')
plt.ylabel('Count')
```

```
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Perform the Chi-Square test
chi2, p, dof, ex = stats.chi2_contingency(contingency_table_ig)
print("\n", f"Chi-Square Test: chi2 = {chi2}, p-value = {p}" , "\n")
```

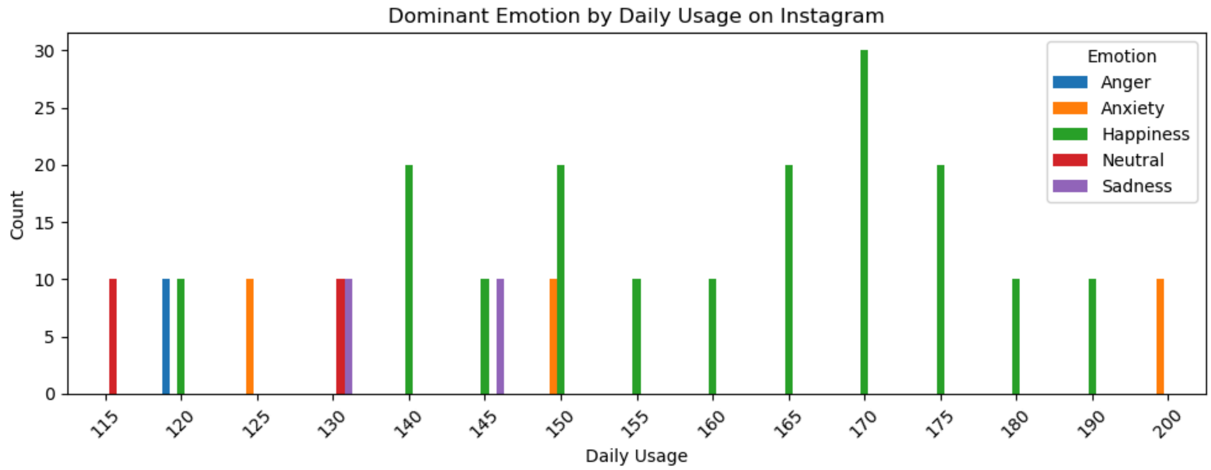
```
Chi-Square Test: chi2 = 608.6601307189542, p-value = 3.7116990458424385e-98
```



Daily Usage Chi-Square Test

In [31]:
```
# Create a contingency table
contingency_table_ig = pd.crosstab(instagram_subset['Dominant_Emotion'], instagram_
contingency_table_flipped_ig = pd.crosstab(instagram_subset['Daily_Usage_Minutes'],

# Plot the bar chart
contingency_table_flipped_ig.plot(kind='bar', figsize=(10, 4))

plt.title('Dominant Emotion by Daily Usage on Instagram')
plt.xlabel('Daily Usage')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Perform the Chi-Square test
chi2, p, dof, ex = stats.chi2_contingency(contingency_table_ig)
print("\n", f"Chi-Square Test: chi2 = {chi2}, p-value = {p}" , "\n")
```

```
Chi-Square Test: chi2 = 607.4346405228756, p-value = 1.3403566441834e-93
```



T-Tests

Age T-Tests

In [32]:
```
# T-Test
# Perform pairwise T-tests for age across different dominant emotions
emotions = instagram_subset['Dominant_Emotion'].unique()

# Conduct pairwise T-tests
t_test_results = {}
for emotion1, emotion2 in combinations(emotions, 2):
    group1 = instagram_subset[instagram_subset['Dominant_Emotion'] == emotion1]['Ag
    group2 = instagram_subset[instagram_subset['Dominant_Emotion'] == emotion2]['Ag
    t_stat, p_val = stats.ttest_ind(group1, group2)
    t_test_results[(emotion1, emotion2)] = (t_stat, p_val)

print("Pairwise T-test results:")
for key, value in t_test_results.items():
    print(f"{key}: t-statistic = {value[0]}, p-value = {value[1]}")
```

```
Pairwise T-test results:
('Happiness', 'Anxiety'): t-statistic = -2.632094531644513, p-value = 0.009154932662
108775
('Happiness', 'Sadness'): t-statistic = 0.7382961072826768, p-value = 0.461255158891
0969
('Happiness', 'Neutral'): t-statistic = 0.7662111754701796, p-value = 0.444511643314
51265
('Happiness', 'Anger'): t-statistic = -5.317536650197723, p-value = 3.12597729720596
1e-07
('Anxiety', 'Sadness'): t-statistic = 1.8737281400611845, p-value = 0.06706252871035
186
('Anxiety', 'Neutral'): t-statistic = 2.08008667208371, p-value = 0.0428802906169743
8
('Anxiety', 'Anger'): t-statistic = -3.461407702594723, p-value = 0.0013437611785476
168
('Sadness', 'Neutral'): t-statistic = 0.0, p-value = 1.0
('Sadness', 'Anger'): t-statistic = -3.5642255405212087, p-value = 0.001333429338403
0186
('Neutral', 'Anger'): t-statistic = -4.27707064862545, p-value = 0.00019909582395247
398
```

```
C:\Users\infin\anaconda3\envs\Pandas\Lib\site-packages\scipy\stats\_axis_nan_policy.
py:531: RuntimeWarning: Precision loss occurred in moment calculation due to catastr
ophic cancellation. This occurs when the data are nearly identical. Results may be u
nreliable.
  res = hypotest_fun_out(*samples, **kwds)
```

Usage T-Tests

In [33]:
```
# T-Test
# Perform pairwise T-tests for dominant emotion across daily usage
emotions = instagram_subset['Dominant_Emotion'].unique()

# Conduct pairwise T-tests
t_test_results = {}
for emotion1, emotion2 in combinations(emotions, 2):
    group1 = instagram_subset[instagram_subset['Dominant_Emotion'] == emotion1]['Da
    group2 = instagram_subset[instagram_subset['Dominant_Emotion'] == emotion2]['Da
    t_stat, p_val = stats.ttest_ind(group1, group2)
    t_test_results[(emotion1, emotion2)] = (t_stat, p_val)

print("Pairwise T-test results:")
for key, value in t_test_results.items():
    print(f"{key}: t-statistic = {value[0]}, p-value = {value[1]}")
```

```
Pairwise T-test results:
('Happiness', 'Anxiety'): t-statistic = 0.42223677990563857, p-value = 0.67331032679
80076
('Happiness', 'Sadness'): t-statistic = 5.800181485363875, p-value = 2.7620844832346
673e-08
('Happiness', 'Neutral'): t-statistic = 9.666969142273125, p-value = 3.3596020884561
065e-18
('Happiness', 'Anger'): t-statistic = 7.371495438892413, p-value = 6.117446221053582
e-12
('Anxiety', 'Sadness'): t-statistic = 2.8728200275324376, p-value = 0.00604171658303
9583
('Anxiety', 'Neutral'): t-statistic = 4.941250447355792, p-value = 9.844642961394271
e-06
('Anxiety', 'Anger'): t-statistic = 3.7892705668204045, p-value = 0.0005248612711937
475
('Sadness', 'Neutral'): t-statistic = 6.164414002968976, p-value = 3.387327223095145
6e-07
('Sadness', 'Anger'): t-statistic = 7.128451081042417, p-value = 9.328959725038259e-
08
('Neutral', 'Anger'): t-statistic = 1.0183501544346312, p-value = 0.3172305566444582
```

Twitter Data

- Age Hypotheses:
  - Null Hypothesis (H0): There is no significant relationship between age and dominant emotion on the Twitter platform.
  - Alternative Hypothesis (H1): There is a significant relationship between age and dominant emotion on the Twitter platform.
- Usage Hypotheses:

- Null Hypothesis (H0): There is no significant relationship between daily usage and dominant emotion on the Twitter platform.
- Alternative Hypothesis (H1): There is a significant relationship between daily usage and dominant emotion on the Twitterplatform.

Filter to Twitter Platform

In [34]:
```
# Filter the data for the Facebook platform
twitter_subset = social_df[social_df['Platform'] == 'Twitter']

print("Twitter Descriptive Data:\n")
twitter_subset.describe()
# Formulate Hypotheses
# H0: There is no significant relationship between age and dominant emotion on the
# H1: There is a significant relationship between age and dominant emotion on the T
```

```
Twitter Descriptive Data:
```

Out[34]:

	User_ID	Age	Daily_Usage_Minutes	Posts_Per_Day	Likes_Received_Per_Day
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	494.300000	26.700000	83.750000	3.405000	35.245000
std	289.347229	3.698431	10.852483	1.288127	9.888574
min	2.000000	21.000000	70.000000	1.000000	12.000000
25%	241.750000	23.750000	73.750000	3.000000	30.000000
50%	499.000000	27.000000	85.000000	3.500000	35.000000
75%	739.250000	29.250000	90.000000	4.000000	43.000000
max	996.000000	35.000000	105.000000	6.000000	50.000000

Describe Targeted/Grouped Data - Dominant Emotion by Age

In [35]:
```
# Descriptive Statistics
cross_reference_age_emotion = twitter_subset.groupby('Dominant_Emotion')['Age'].des

print("Cross-reference of Age with Dominant Emotion on Twitter:\n")
print(cross_reference_age_emotion)
```

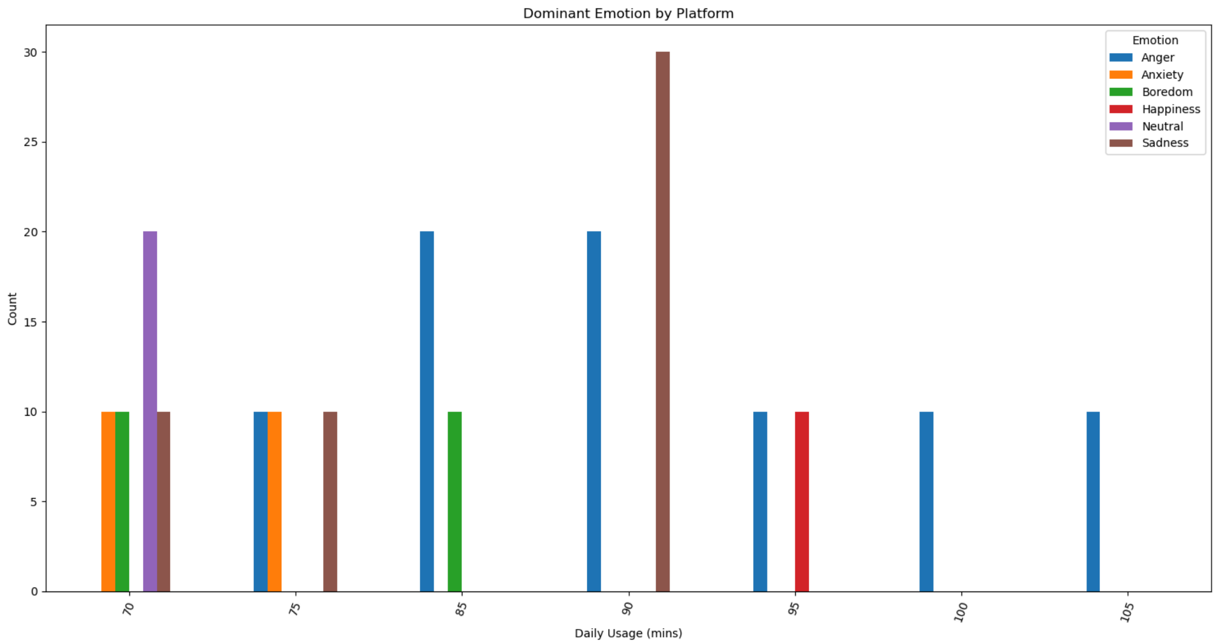
```
Cross-reference of Age with Dominant Emotion on Twitter:

                 count    mean       std   min    25%   50%    75%   max
Dominant_Emotion
Anger             80.0  27.375  2.246657  24.0  26.25  27.5  29.25  30.0
Anxiety           20.0  26.500  3.590924  23.0  23.00  26.5  30.00  30.0
Boredom           20.0  34.000  1.025978  33.0  33.00  34.0  35.00  35.0
Happiness         10.0  28.000  0.000000  28.0  28.00  28.0  28.00  28.0
Neutral           20.0  22.500  0.512989  22.0  22.00  22.5  23.00  23.0
Sadness           50.0  24.200  2.338672  21.0  22.00  25.0  26.00  27.0
```

Describe Targeted/Grouped Data - Dominant Emotion by Daily Usage

In [36]:

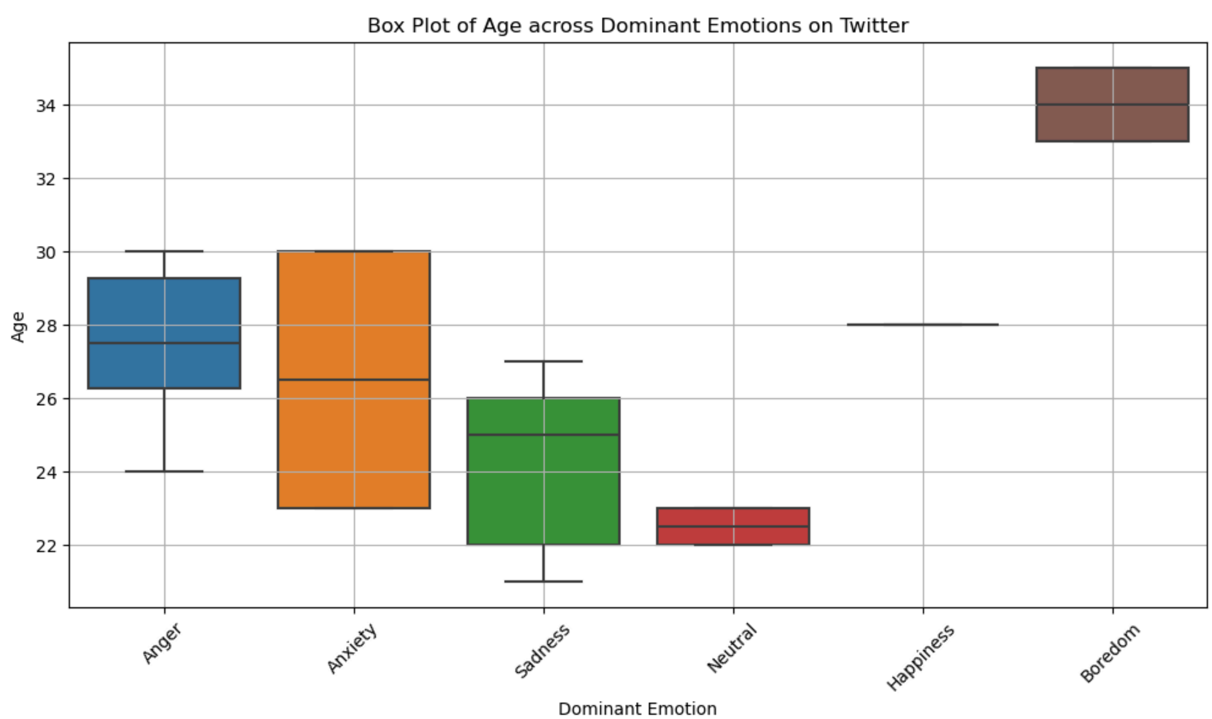
```python
# Descriptive Statistics
contingency_emotion_usage = pd.crosstab(twitter_subset['Daily_Usage_Minutes'], twit


cross_reference_emotion_usage = twitter_subset.groupby('Dominant_Emotion')['Daily_U
print("Cross-reference of Daily Usage with Dominant Emotion on Twitter:\n")
print(cross_reference_emotion_usage)




# Plot the bar chart
contingency_emotion_usage.plot(kind='bar', figsize=(15, 8))

plt.title('Dominant Emotion by Platform')
plt.xlabel('Daily Usage (mins)')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=70)
plt.tight_layout()
```

```
Cross-reference of Daily Usage with Dominant Emotion on Twitter:

                 count    mean       std   min   25%   50%    75%    max
Dominant_Emotion
Anger             80.0  90.625  8.872336  75.0  85.0  90.0  96.25  105.0
Anxiety           20.0  72.500  2.564946  70.0  70.0  72.5  75.00   75.0
Boredom           20.0  77.500  7.694838  70.0  70.0  77.5  85.00   85.0
Happiness         10.0  95.000  0.000000  95.0  95.0  95.0  95.00   95.0
Neutral           20.0  70.000  0.000000  70.0  70.0  70.0  70.00   70.0
Sadness           50.0  83.000  8.806306  70.0  75.0  90.0  90.00   90.0
```



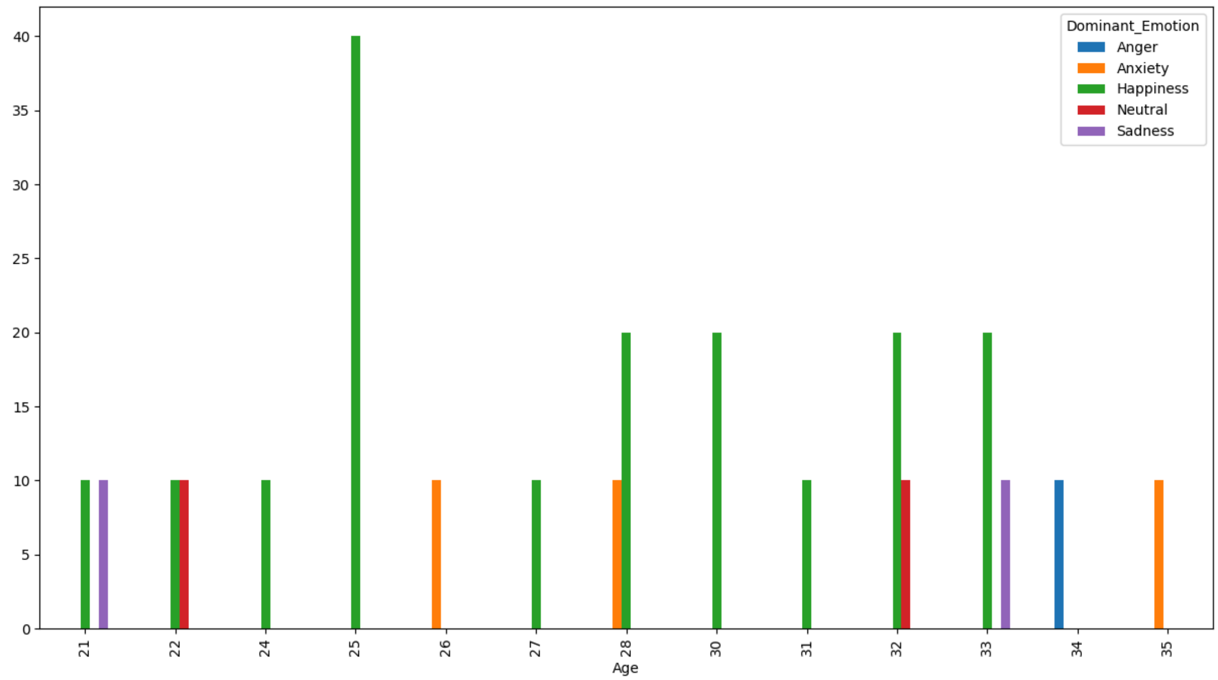
Box Plot

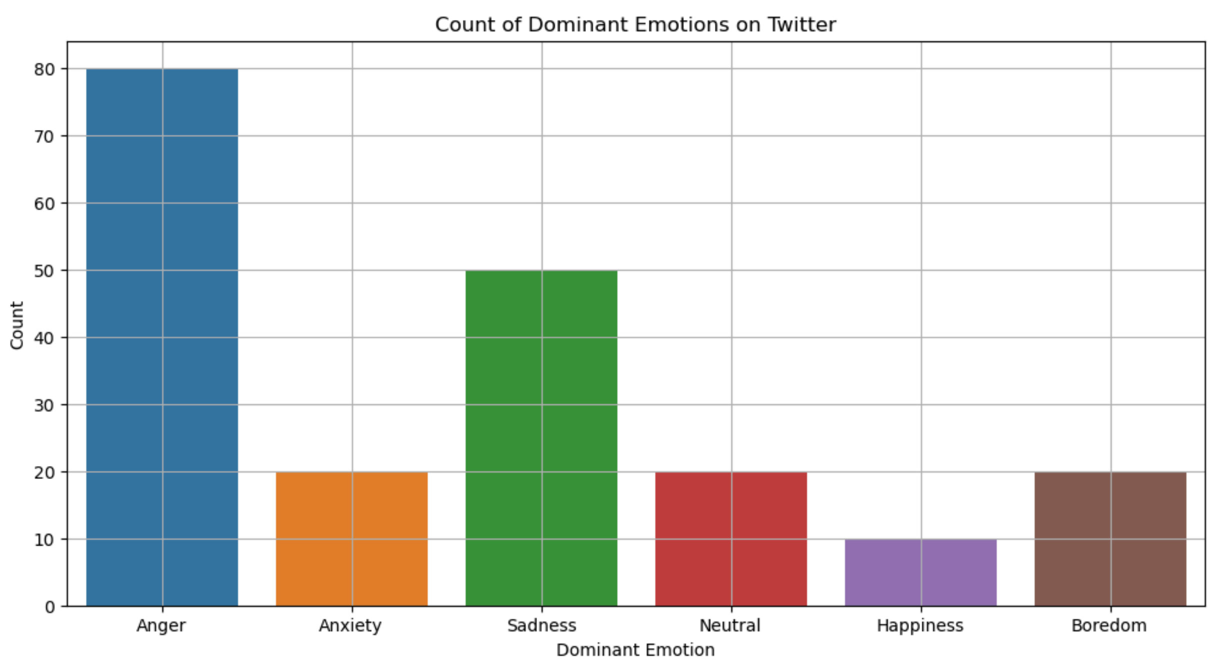
In [37]:
```python
# Box Plot of Age for each Dominant Emotion
plt.figure(figsize=(12, 6))
sns.boxplot(data=twitter_subset, x='Dominant_Emotion', y='Age')
plt.title('Box Plot of Age across Dominant Emotions on Twitter')
plt.xlabel('Dominant Emotion')
plt.ylabel('Age')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Bar Plot

In [38]:
```
# Bar Plot of the Count of Each Dominant Emotion

contingency_table_flipped.plot(kind='bar', figsize=(15, 8))

plt.figure(figsize=(12, 6))
sns.countplot(data=twitter_subset, x='Dominant_Emotion')
plt.title('Count of Dominant Emotions on Twitter')
plt.xlabel('Dominant Emotion')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



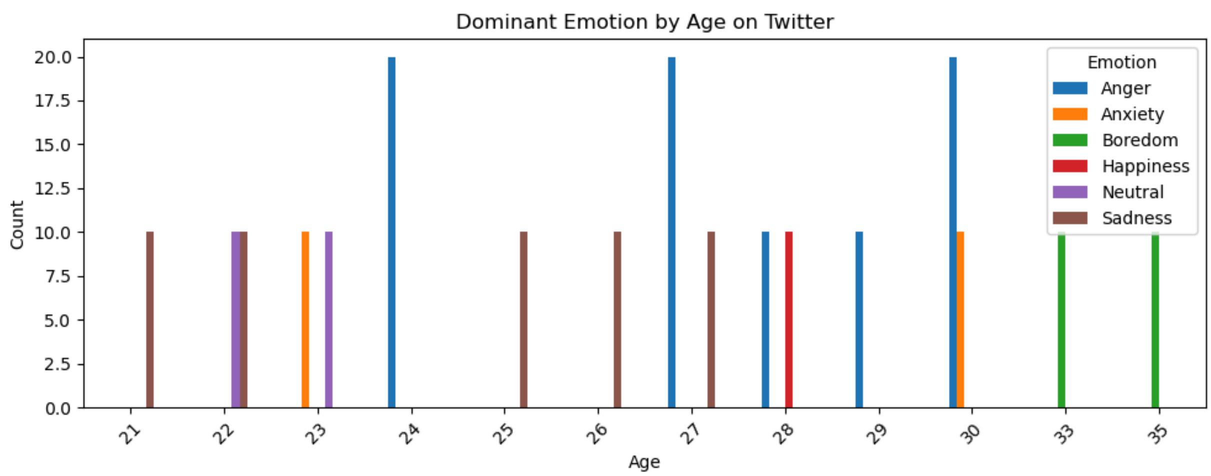


Chi-Square Tests

Age Chi-Square

In [39]:
```
# Create a contingency table
contingency_table_ig = pd.crosstab(twitter_subset['Dominant_Emotion'], twitter_subs
contingency_table_flipped_ig = pd.crosstab(twitter_subset['Age'], twitter_subset['D

# Plot the bar chart
contingency_table_flipped_ig.plot(kind='bar', figsize=(10, 4))

plt.title('Dominant Emotion by Age on Twitter')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Perform the Chi-Square test
chi2, p, dof, ex = stats.chi2_contingency(contingency_table_ig)
print("\n", f"Chi-Square Test: chi2 = {chi2}, p-value = {p}" , "\n")
```

```
Chi-Square Test: chi2 = 590.8333333333334, p-value = 7.763689734444644e-91
```

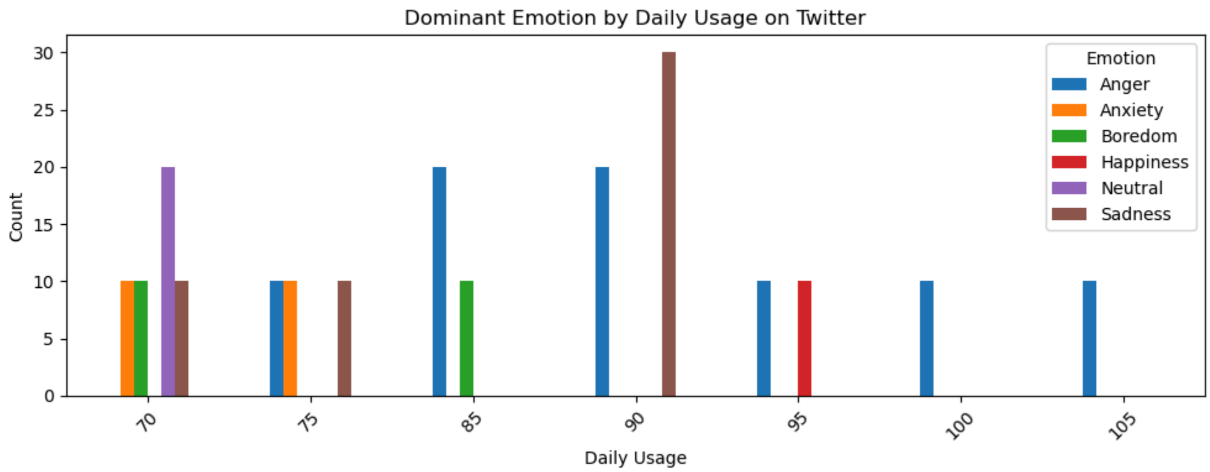


Usage Chi-Sqaure

In [40]:
```
# Create a contingency table
contingency_table_ig = pd.crosstab(twitter_subset['Dominant_Emotion'], twitter_subs
contingency_table_flipped_ig = pd.crosstab(twitter_subset['Daily_Usage_Minutes'], t

# Plot the bar chart
contingency_table_flipped_ig.plot(kind='bar', figsize=(10, 4))

plt.title('Dominant Emotion by Daily Usage on Twitter')
plt.xlabel('Daily Usage')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Perform the Chi-Square test
```

```
chi2, p, dof, ex = stats.chi2_contingency(contingency_table_ig)
print("\n", f"Chi-Square Test: chi2 = {chi2}, p-value = {p}" , "\n")
```

```
Chi-Square Test: chi2 = 304.1666666666667, p-value = 3.994003712776512e-47
```



T-Tests

Age T-Tests

In [41]:
```
# T-Test
# Perform pairwise T-tests for age across different dominant emotions
emotions = twitter_subset['Dominant_Emotion'].unique()

# Conduct pairwise T-tests
t_test_results = {}
for emotion1, emotion2 in combinations(emotions, 2):
    group1 = twitter_subset[twitter_subset['Dominant_Emotion'] == emotion1]['Age'].
    group2 = twitter_subset[twitter_subset['Dominant_Emotion'] == emotion2]['Age'].
    t_stat, p_val = stats.ttest_ind(group1, group2)
    t_test_results[(emotion1, emotion2)] = (t_stat, p_val)

print("Pairwise T-test results:")
for key, value in t_test_results.items():
    print(f"{key}: t-statistic = {value[0]}, p-value = {value[1]}")
```

```
Pairwise T-test results:
('Anger', 'Anxiety'): t-statistic = 1.3655967080115705, p-value = 0.1751917827154220
2
('Anger', 'Sadness'): t-statistic = 7.716591705266487, p-value = 2.9667056166942818e
-12
('Anger', 'Neutral'): t-statistic = 9.60707943301179, p-value = 8.627248536714974e-1
6
('Anger', 'Happiness'): t-statistic = -0.875376219064817, p-value = 0.38375229190327
63
('Anger', 'Boredom'): t-statistic = -12.819807435627384, p-value = 1.120299738064496
5e-22
('Anxiety', 'Sadness'): t-statistic = 3.1650063800388217, p-value = 0.00232042043852
41815
('Anxiety', 'Neutral'): t-statistic = 4.931531202375181, p-value = 1.643870033455229
7e-05
('Anxiety', 'Happiness'): t-statistic = -1.3093073414159544, p-value = 0.20107335807
489865
('Anxiety', 'Boredom'): t-statistic = -8.981112256201376, p-value = 6.19603323755009
2e-11
('Sadness', 'Neutral'): t-statistic = 3.206808560042833, p-value = 0.002046718271257
754
('Sadness', 'Happiness'): t-statistic = -5.103164563015182, p-value = 3.862380391313
066e-06
('Sadness', 'Boredom'): t-statistic = -17.9984567239651, p-value = 1.444915796524386
7e-27
('Neutral', 'Happiness'): t-statistic = -33.605555096342826, p-value = 3.55826372566
1098e-24
('Neutral', 'Boredom'): t-statistic = -44.83525398612123, p-value = 1.63254960126378
96e-34
('Happiness', 'Boredom'): t-statistic = -18.33030277982336, p-value = 3.939668183958
516e-17
```

```
C:\Users\infin\anaconda3\envs\Pandas\Lib\site-packages\scipy\stats\_axis_nan_policy.
py:531: RuntimeWarning: Precision loss occurred in moment calculation due to catastr
ophic cancellation. This occurs when the data are nearly identical. Results may be u
nreliable.
  res = hypotest_fun_out(*samples, **kwds)
```

Usage T-Tests

In [42]:

```python
# T-Test
# Perform pairwise T-tests for dominant emotion across daily usage
emotions = twitter_subset['Dominant_Emotion'].unique()

# Conduct pairwise T-tests
t_test_results = {}
for emotion1, emotion2 in combinations(emotions, 2):
    group1 = twitter_subset[twitter_subset['Dominant_Emotion'] == emotion1]['Daily_
    group2 = twitter_subset[twitter_subset['Dominant_Emotion'] == emotion2]['Daily_
    t_stat, p_val = stats.ttest_ind(group1, group2)
    t_test_results[(emotion1, emotion2)] = (t_stat, p_val)

print("Pairwise T-test results:")
for key, value in t_test_results.items():
    print(f"{key}: t-statistic = {value[0]}, p-value = {value[1]}")
```

```
Pairwise T-test results:
('Anger', 'Anxiety'): t-statistic = 9.011104260855047, p-value = 1.6925066017689062e
-14
('Anger', 'Sadness'): t-statistic = 4.780753839596978, p-value = 4.717659295640469e-
06
('Anger', 'Neutral'): t-statistic = 10.35655789619729, p-value = 2.0399232994694737e
-17
('Anger', 'Happiness'): t-statistic = -1.5516422020464036, p-value = 0.1243370002962
1696
('Anger', 'Boredom'): t-statistic = 6.064756928001706, p-value = 2.4805615927525357e
-08
('Anxiety', 'Sadness'): t-statistic = -5.223660039074294, p-value = 1.81775916226948
64e-06
('Anxiety', 'Neutral'): t-statistic = 4.358898943540674, p-value = 9.6033545892936e-
05
('Anxiety', 'Happiness'): t-statistic = -27.49545416973504, p-value = 8.348996731210
183e-22
('Anxiety', 'Boredom'): t-statistic = -2.7568097504180447, p-value = 0.0089186270404
53582
('Sadness', 'Neutral'): t-statistic = 6.572899475786694, p-value = 8.268908220956217
e-09
('Sadness', 'Happiness'): t-statistic = -4.279695021106551, p-value = 7.100602981173
518e-05
('Sadness', 'Boredom'): t-statistic = 2.442671180196237, p-value = 0.017182466702658
786
('Neutral', 'Happiness'): t-statistic = -inf, p-value = 0.0
('Neutral', 'Boredom'): t-statistic = -4.358898943540673, p-value = 9.6033545892936e
-05
('Happiness', 'Boredom'): t-statistic = 7.128451081042417, p-value = 9.3289597250382
59e-08
```

Comparative Data: Age, Usage & Dominant Emotion Across Top 3 Used Social Media Platforms

Filter & Describe Data

In [43]:
```
# Exclude rows where 'Age' is 'Female'
social_df_filtered = social_df[~social_df['Platform'].isin(['LinkedIn', 'Whatsapp',

# Group the data by
grouped_data_age = social_df_filtered.groupby('Platform')['Age']

# Group the data by
grouped_data_usage = social_df_filtered.groupby('Platform')['Daily_Usage_Minutes']

# Group the data by
grouped_data_emotion = social_df_filtered.groupby('Platform')['Dominant_Emotion']

# Display results
social_df_filtered.describe(), grouped_data_age.describe(), grouped_data_usage.desc
```

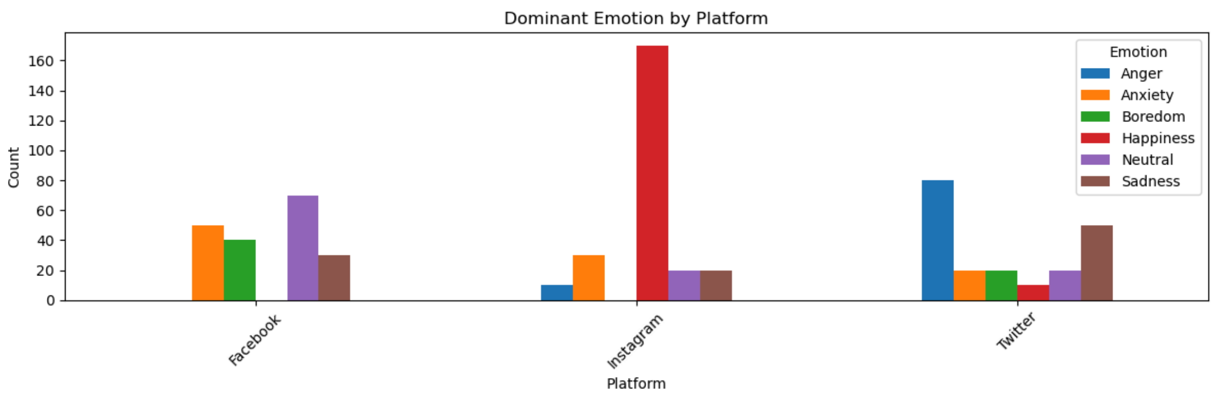
Out[43]:
```
(           User_ID         Age  Daily_Usage_Minutes  Posts_Per_Day  \
 count  640.000000  640.000000           640.000000     640.000000   
 mean   494.125000   27.109375           107.500000       3.907812   
 std    288.804969    3.904643            41.433621       2.039347   
 min      1.000000   21.000000            40.000000       1.000000   
 25%    241.500000   24.000000            75.000000       2.000000   
 50%    499.000000   27.000000            95.000000       4.000000   
 75%    740.500000   30.000000           145.000000       6.000000   
 max    997.000000   35.000000           200.000000       8.000000   

        Likes_Received_Per_Day  Comments_Received_Per_Day  \
 count              640.000000                 640.000000   
 mean                47.835938                  18.217188   
 std                 28.495443                   9.117286   
 min                  5.000000                   2.000000   
 25%                 25.000000                  11.000000   
 50%                 40.000000                  18.000000   
 75%                 72.000000                  25.000000   
 max                110.000000                  40.000000   

        Messages_Sent_Per_Day  
 count             640.000000  
 mean               24.660937  
 std                 8.993079  
 min                10.000000  
 25%                18.000000  
 50%                24.000000  
 75%                30.000000  
 max                50.000000  ,
            count       mean       std   min    25%   50%    75%   max
 Platform                                                              
 Facebook   190.0  26.263158  3.330826  21.0  23.00  26.0  29.00  33.0
 Instagram  250.0  28.080000  4.259834  21.0  25.00  28.0  32.00  35.0
 Twitter    200.0  26.700000  3.698431  21.0  23.75  27.0  29.25  35.0,
            count        mean        std    min     25%    50%    75%    max
 Platform                                                                   
 Facebook   190.0   72.105263  19.471577   40.0   60.00   70.0   85.0  110.0
 Instagram  250.0  153.400000  22.348104  115.0  140.00  150.0  170.0  200.0
 Twitter    200.0   83.750000  10.852483   70.0   73.75   85.0   90.0  105.0,
           count unique        top freq
 Platform                              
 Facebook    190      4    Neutral   70
 Instagram   250      5  Happiness  170
 Twitter     200      6      Anger   80)
```

Dominant Emotion by Platform

In [44]:
```python
# Create a contingency table that cross-classifies emotion with age
contingency_table_flipped = pd.crosstab(social_df_filtered['Platform'], social_df_f

# Plot the bar chart
contingency_table_flipped.plot(kind='bar', figsize=(12, 4))

plt.title('Dominant Emotion by Platform')
```

```
plt.xlabel('Platform')
plt.ylabel('Count')
plt.legend(title='Emotion')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the results
print(grouped_data_emotion.describe(), "\n")
```

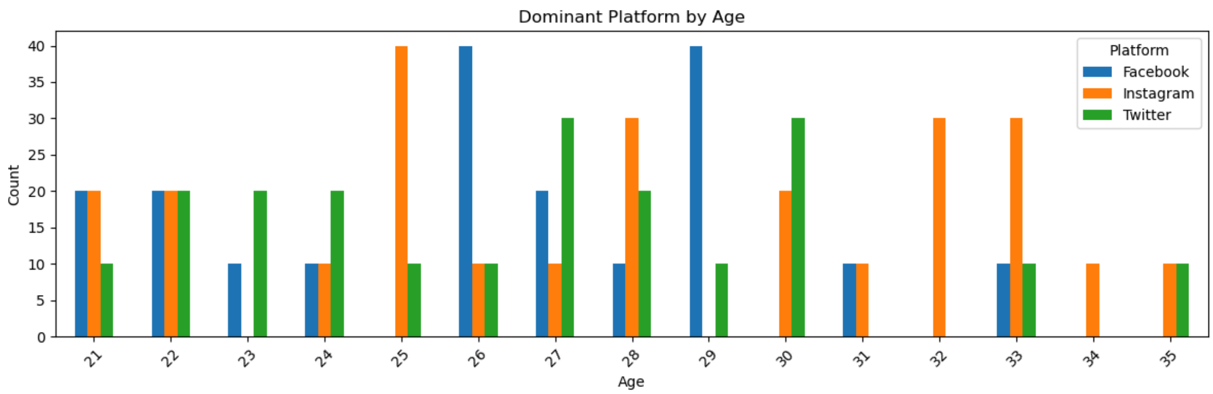
```
          count unique        top freq
Platform
Facebook    190      4    Neutral   70
Instagram   250      5  Happiness  170
Twitter     200      6      Anger   80
```



Age by Platform

In [45]:
```
# Create a contingency table that cross-classifies emotion with age
contingency_table_flipped = pd.crosstab(social_df_filtered['Age'], social_df_filter

# Plot the bar chart
contingency_table_flipped.plot(kind='bar', figsize=(12, 4))

plt.title('Dominant Platform by Age')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(title='Platform')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the results
print(grouped_data_age.describe(), "\n")
```

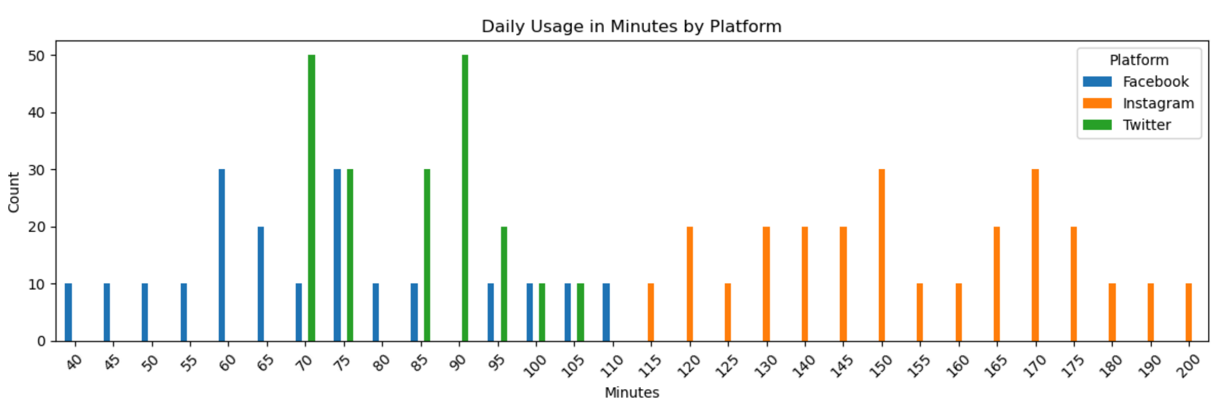
```
           count       mean       std   min    25%   50%    75%   max
Platform
Facebook   190.0  26.263158  3.330826  21.0  23.00  26.0  29.00  33.0
Instagram  250.0  28.080000  4.259834  21.0  25.00  28.0  32.00  35.0
Twitter    200.0  26.700000  3.698431  21.0  23.75  27.0  29.25  35.0
```



Daily Usage by Platform

In [46]:
```
# Create a contingency table that cross-classifies emotion with age
contingency_table_flipped = pd.crosstab(social_df_filtered['Daily_Usage_Minutes'],

# Plot the bar chart
contingency_table_flipped.plot(kind='bar', figsize=(12, 4))

plt.title('Daily Usage in Minutes by Platform')
plt.xlabel('Minutes')
plt.ylabel('Count')
plt.legend(title='Platform')
plt.xticks(rotation=45)
plt.tight_layout()

# Display the results
print(grouped_data_usage.describe(), "\n")
```

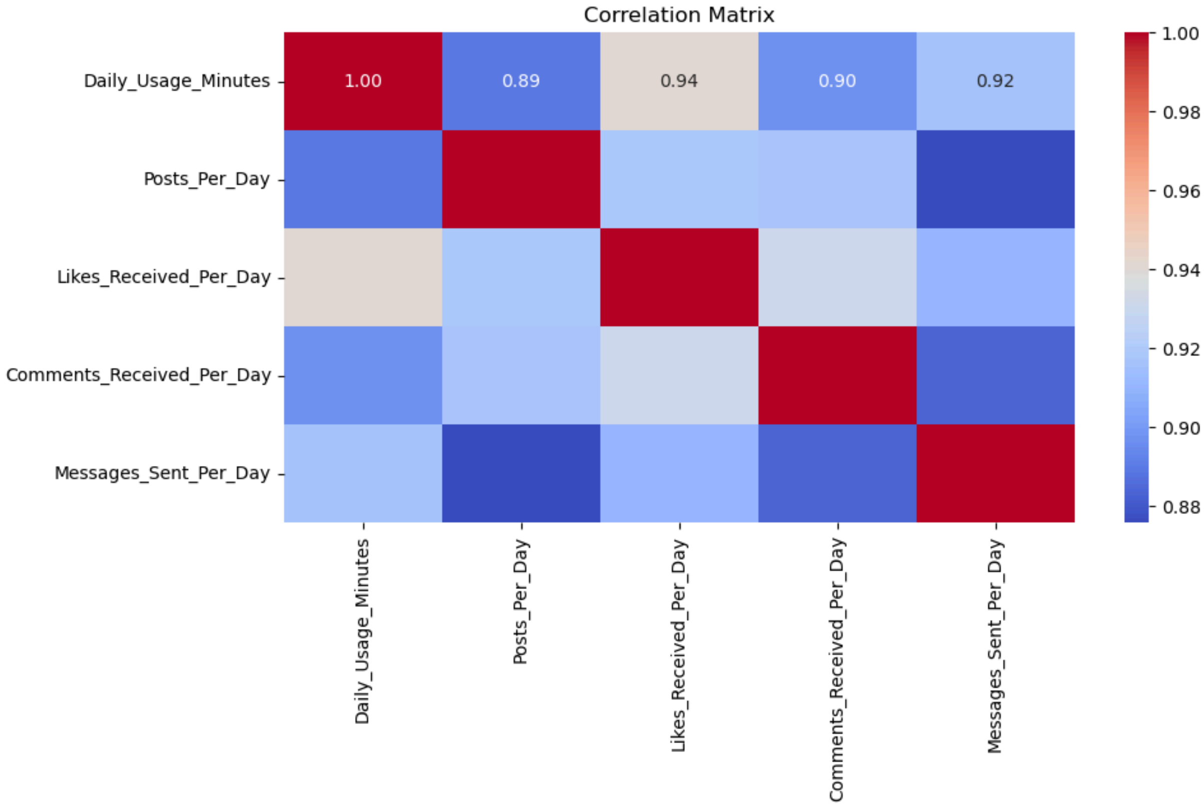
```
           count        mean        std    min     25%    50%    75%    max
Platform
Facebook   190.0   72.105263  19.471577   40.0   60.00   70.0   85.0  110.0
Instagram  250.0  153.400000  22.348104  115.0  140.00  150.0  170.0  200.0
Twitter    200.0   83.750000  10.852483   70.0   73.75   85.0   90.0  105.0
```



Section 3: Model Selection and Analysis

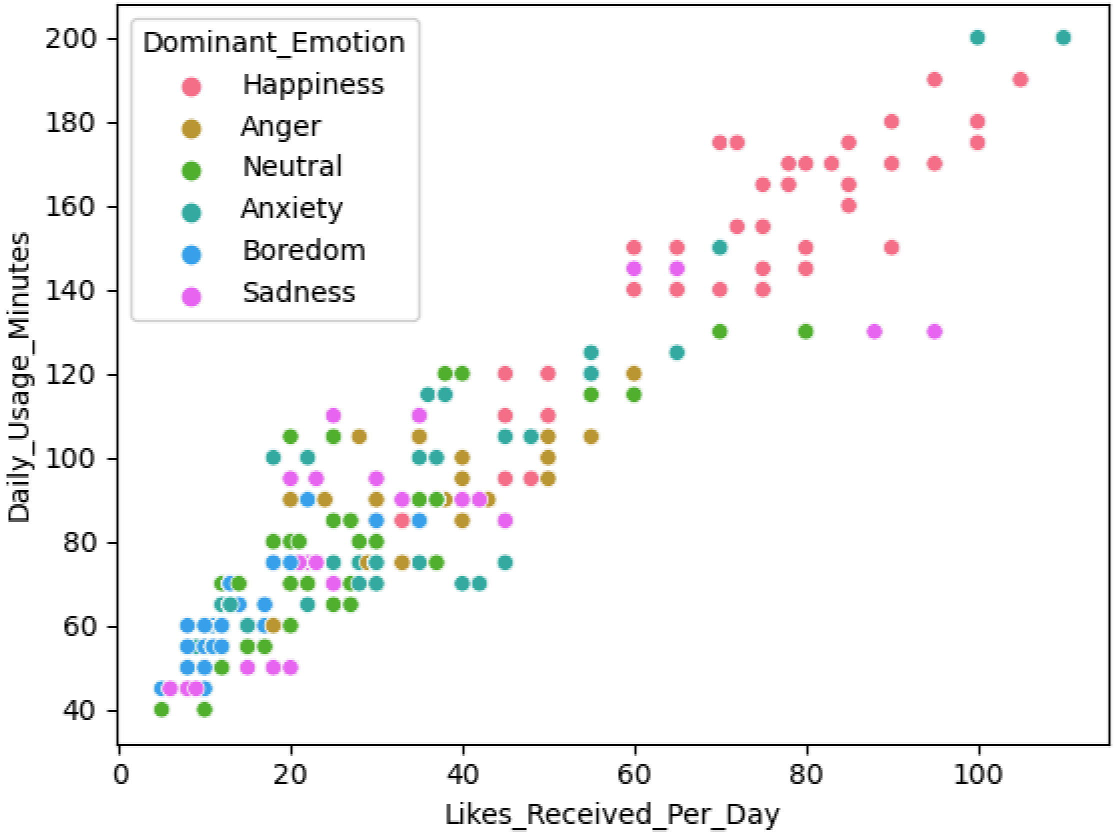
Model Selection

In [47]:
```
corr_matrix = social_df[['Daily_Usage_Minutes','Posts_Per_Day','Likes_Received_Per_
plt.figure(figsize=(10, 5))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

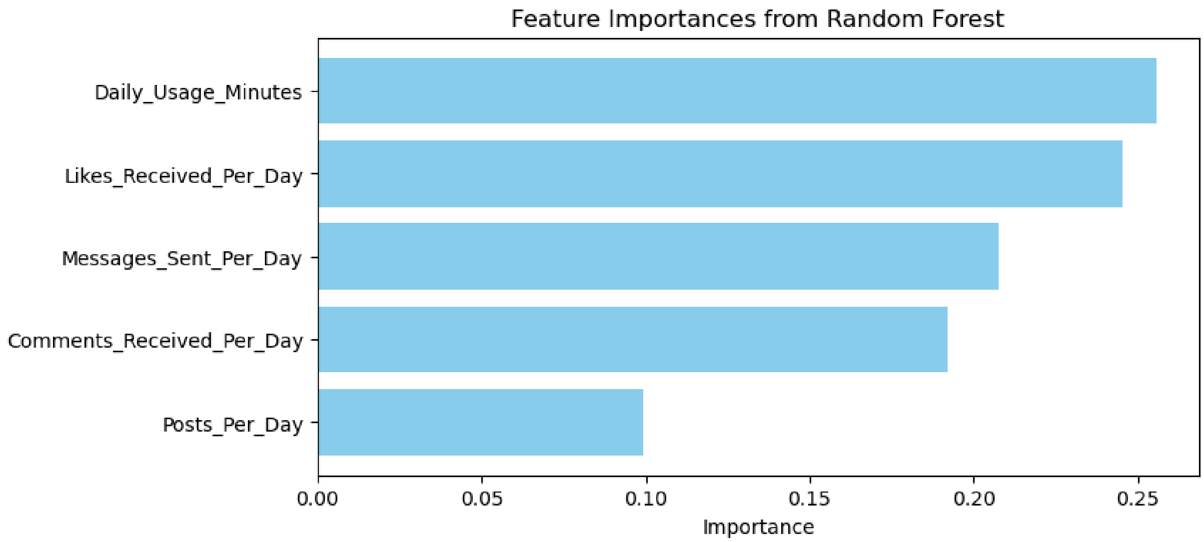


In [48]:
```
# visualization 'Likes_Received_Per_Day' vs 'Daily_Usage_Time (minutes)' depending

sns.scatterplot(social_df, x='Likes_Received_Per_Day', y='Daily_Usage_Minutes', hue
```

Out[48]: <Axes: xlabel='Likes_Received_Per_Day', ylabel='Daily_Usage_Minutes'>



In [49]:
```
#random forest feature importance to determine what to put into our overall engagem

X = social_df[['Daily_Usage_Minutes','Posts_Per_Day','Likes_Received_Per_Day','Comm
y = social_df["Age"].values

model_random_forest = RandomForestClassifier(n_estimators=100, random_state=42)

model_random_forest.fit(X, y)

importances = model_random_forest.feature_importances_
feature_names = X.columns

feature_importances = pd.DataFrame({'Feature': feature_names, 'Importance': importa
feature_importances = feature_importances.sort_values(by='Importance', ascending=Fa

plt.figure(figsize=(8, 4))
plt.barh(feature_importances['Feature'], feature_importances['Importance'], color='
plt.xlabel('Importance')
plt.title('Feature Importances from Random Forest')
plt.gca().invert_yaxis()
plt.show()

print('Based on this, we will use daily usage minutes, likes, messages, and comment
```



Based on this, we will use daily usage minutes, likes, messages, and comments received.

In [50]:
```
# Create dummy variables for 'dominant_emotion'
social_df = pd.read_csv("https://raw.githubusercontent.com/gurlv/SocialMediaDataset
dummies_df = social_df

df_with_dummies = pd.get_dummies(dummies_df, columns=['Dominant_Emotion'], drop_fir
for col in df_with_dummies.select_dtypes(include=['object']).columns:
    if col != 'Daily_Usage_Minutes':
        try:
            df_with_dummies[col] = pd.to_numeric(df_with_dummies[col])
        except ValueError:
            print(f"Warning: Column '{col}' cannot be converted to numeric and will
            df_with_dummies.drop(col, axis=1, inplace=True)

df_with_dummies.dropna(subset=df_with_dummies.columns.difference(['Daily_Usage_Minu
df_with_dummies['Daily_Usage_Minutes_Std'] = (df_with_dummies['Daily_Usage_Minutes'
df_with_dummies['Likes_Received_Per_Day_Std'] = (df_with_dummies['Likes_Received_Pe
df_with_dummies['Messages_Sent_Per_Day_Std'] = (df_with_dummies['Messages_Sent_Per_
df_with_dummies['Comments_Received_Per_Day_Std'] = (df_with_dummies['Comments_Recei

df_with_dummies['Overall_Engagement'] = df_with_dummies['Daily_Usage_Minutes_Std']

for col in df_with_dummies.select_dtypes(include=['int', 'float']).columns:
    if col in ('Age', 'User_ID', 'Posts_Per_Day','Daily_Usage_Minutes_Std', 'Likes_
               'Messages_Sent_Per_Day_Std', 'Comments_Received_Per_Day_Std', 'Likes
               'Messages_Sent_Per_Day', 'Comments_Received_Per_Day'):
        df_with_dummies.drop(col, axis=1, inplace=True) # not relevant for our stud

# Define the independent variables (X) and the dependent variable (y)
X = df_with_dummies.drop('Daily_Usage_Minutes', axis=1)
y = df_with_dummies['Daily_Usage_Minutes']

# Add a constant term to the independent variables matrix (required for the interce
X = sm.add_constant(X)
# Fit the linear regression model
model = sm.OLS(y, X)
results = model.fit()
```

```
# code ran to detect high multicollinearity
# # from statsmodels.stats.outliers_influence import variance_inflation_factor

# # Get the predictor variables (excluding the constant term)
# X_vif = X.drop('const', axis=1)

# # Calculate VIF for each variable
# vif_data = pd.DataFrame()
# vif_data["Variable"] = X_vif.columns
# vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in range(X_vi

# print(vif_data)

# print('VIF > 5 indicates high multicollinearity. It needs to be addressed.')

# Print the model summary
print("\n--- Daily Usage Minutes Linear Regression Summary ---\n")
print(results.summary())

print("\n---- Overall Engagement Linear Regression Summary ----\n")
X_for_likes = df_with_dummies.drop('Overall_Engagement', axis=1)
y_for_likes = df_with_dummies['Overall_Engagement']

# Add a constant term to the independent variables matrix (required for the interce
X_for_likes = sm.add_constant(X_for_likes)
# Fit the linear regression model
model_for_likes = sm.OLS(y_for_likes, X_for_likes)
results_for_likes = model_for_likes.fit()

# Print the model summary
print(results_for_likes.summary())
```

```
Warning: Column 'Gender' cannot be converted to numeric and will be dropped.
Warning: Column 'Platform' cannot be converted to numeric and will be dropped.

--- Daily Usage Minutes Linear Regression Summary ---

                            OLS Regression Results
==============================================================================
Dep. Variable:     Daily_Usage_Minutes   R-squared:                       0.951
Model:                             OLS   Adj. R-squared:                  0.950
Method:                  Least Squares   F-statistic:                     3188.
Date:                 Fri, 21 Jun 2024   Prob (F-statistic):               0.00
Time:                         03:02:58   Log-Likelihood:                -3573.7
No. Observations:                 1000   AIC:                             7161.
Df Residuals:                      993   BIC:                             7196.
Df Model:                            6
Covariance Type:             nonrobust
====================================================================================
==========
                                coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------------
----------
const                        89.1760      0.759    117.451      0.000      87.686
90.666
Dominant_Emotion_Anxiety      5.2713      1.009      5.227      0.000       3.292
7.250
Dominant_Emotion_Boredom     10.1208      1.129      8.962      0.000       7.905
12.337
Dominant_Emotion_Happiness   11.5431      1.118     10.323      0.000       9.349
13.737
Dominant_Emotion_Neutral      6.9591      0.999      6.969      0.000       5.000
8.919
Dominant_Emotion_Sadness      4.7535      1.029      4.617      0.000       2.733
6.774
Overall_Engagement            9.5240      0.106     89.488      0.000       9.315
9.733
==============================================================================
Omnibus:                        2.708   Durbin-Watson:                   1.973
Prob(Omnibus):                  0.258   Jarque-Bera (JB):                2.565
Skew:                           0.105   Prob(JB):                        0.277
Kurtosis:                       3.133   Cond. No.                         27.8
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.

---- Overall Engagement Linear Regression Summary ----

                            OLS Regression Results
==============================================================================
Dep. Variable:      Overall_Engagement   R-squared:                       0.951
Model:                             OLS   Adj. R-squared:                  0.951
Method:                  Least Squares   F-statistic:                     3223.
Date:                 Fri, 21 Jun 2024   Prob (F-statistic):               0.00
Time:                         03:02:58   Log-Likelihood:                -1261.5
```

```
No. Observations:                1000   AIC:                             2537.
Df Residuals:                     993   BIC:                             2571.
Df Model:                           6
Covariance Type:            nonrobust
=====================================================================================
==========
                                 coef    std err          t      P>|t|      [0.025
0.975]
-------------------------------------------------------------------------------------
----------
const                         -8.3208      0.120    -69.148      0.000      -8.557
-8.085
Daily_Usage_Minutes            0.0934      0.001     89.488      0.000       0.091
0.095
Dominant_Emotion_Anxiety      -0.4887      0.100     -4.885      0.000      -0.685
-0.292
Dominant_Emotion_Boredom      -1.3647      0.108    -12.647      0.000      -1.576
-1.153
Dominant_Emotion_Happiness    -0.5112      0.115     -4.430      0.000      -0.738
-0.285
Dominant_Emotion_Neutral      -0.8726      0.097     -8.956      0.000      -1.064
-0.681
Dominant_Emotion_Sadness      -0.5715      0.101     -5.634      0.000      -0.771
-0.372
==============================================================================
Omnibus:                        2.209   Durbin-Watson:                   1.824
Prob(Omnibus):                  0.331   Jarque-Bera (JB):                2.120
Skew:                           0.069   Prob(JB):                        0.346
Kurtosis:                       3.178   Cond. No.                         751.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

In [51]:

```python
from sklearn.linear_model import LogisticRegression

social_df = pd.read_csv("https://raw.githubusercontent.com/gurlv/SocialMediaDataset
social_df_regression = social_df.copy()

# Standardize the variables to their std so they dont dominate each other in the en
# feature importance.
social_df_regression['Daily_Usage_Minutes_Std'] = (social_df_regression['Daily_Usag
social_df_regression['Likes_Received_Per_Day_Std'] = (social_df_regression['Likes_R
social_df_regression['Messages_Sent_Per_Day_Std'] = (social_df_regression['Messages
social_df_regression['Comments_Received_Per_Day_Std'] = (social_df_regression['Comm

social_df_regression['Overall_Engagement'] = social_df_regression['Daily_Usage_Minu

# Convert 'Dominant_Emotion' to numerical labels
le = LabelEncoder()
social_df_regression['Dominant_Emotion_Encoded'] = le.fit_transform(social_df_regre

# Define your predictor variables (X) and target variable (y)
y = social_df_regression['Dominant_Emotion_Encoded']
X = social_df_regression[['Daily_Usage_Minutes_Std', 'Likes_Received_Per_Day_Std',
```

```
# 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_sta

# Create andexpand_more fit the multinomial logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs', max_iter=1000
model.fit(X_train, y_train)

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

# Evaluate the model (Classification Report)
print('Multi Logistic Regression Model Results\n')

print(classification_report(y_test, y_pred, target_names=le.classes_))

# Examine the model's coefficients (for interpretation)
print('\nCoefficients:')
for i, class_name in enumerate(le.classes_):
    print(f'\n{class_name}:')
    for feature, coef in zip(X.columns, model.coef_[i]):
        print(f'{feature}: {coef:.2f}')
```

```
Multi Logistic Regression Model Results

              precision    recall  f1-score   support

       Anger       0.47      0.48      0.47        29
     Anxiety       0.14      0.10      0.12        39
     Boredom       0.48      0.71      0.57        31
   Happiness       0.76      0.91      0.83        35
     Neutral       0.36      0.36      0.36        42
     Sadness       0.09      0.04      0.06        24

    accuracy                           0.44       200
   macro avg       0.38      0.43      0.40       200
weighted avg       0.39      0.44      0.41       200


Coefficients:

Anger:
Daily_Usage_Minutes_Std: -2.73
Likes_Received_Per_Day_Std: -1.40
Overall_Engagement: 1.19

Anxiety:
Daily_Usage_Minutes_Std: -0.46
Likes_Received_Per_Day_Std: -0.12
Overall_Engagement: 0.34

Boredom:
Daily_Usage_Minutes_Std: 1.91
Likes_Received_Per_Day_Std: -1.46
Overall_Engagement: -0.92

Happiness:
Daily_Usage_Minutes_Std: 1.04
Likes_Received_Per_Day_Std: 0.11
Overall_Engagement: 0.33

Neutral:
Daily_Usage_Minutes_Std: 0.67
Likes_Received_Per_Day_Std: 1.69
Overall_Engagement: -0.74

Sadness:
Daily_Usage_Minutes_Std: -0.42
Likes_Received_Per_Day_Std: 1.18
Overall_Engagement: -0.20
```

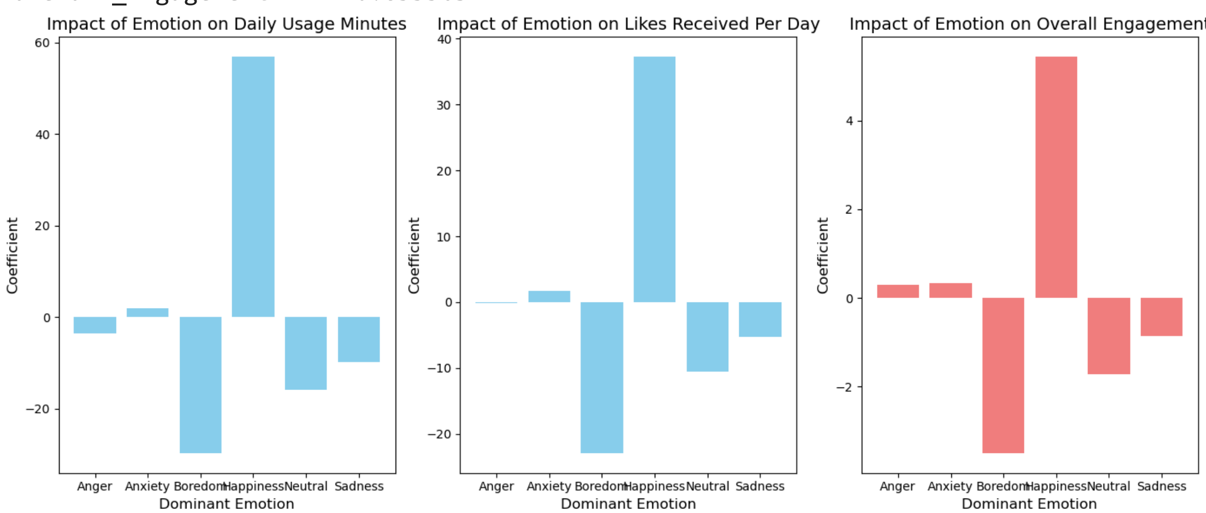
In [52]:

```
social_df = pd.read_csv("https://raw.githubusercontent.com/gurlv/SocialMediaDataset
social_df_regression = social_df.copy()

# Standardize the variables to their std so they dont dominate each other in the en
# feature importance.
social_df_regression['Daily_Usage_Minutes_Std'] = (social_df_regression['Daily_Usag
social_df_regression['Likes_Received_Per_Day_Std'] = (social_df_regression['Likes_R
social_df_regression['Messages_Sent_Per_Day_Std'] = (social_df_regression['Messages
```

```
social_df_regression['Comments_Received_Per_Day_Std'] = (social_df_regression['Comm

social_df_regression['Overall_Engagement'] = social_df_regression['Daily_Usage_Minu

X = social_df_regression[['Dominant_Emotion']]
y = social_df_regression[['Daily_Usage_Minutes', 'Likes_Received_Per_Day', 'Overall

# Create a preprocessing pipeline for categorical encoding
preprocessor = ColumnTransformer(
    transformers=[
        ('encoder', OneHotEncoder(handle_unknown='ignore'), ['Dominant_Emotion'])
    ]
)

# Create a pipeline that combines preprocessing and regression
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# Fit the model
pipeline.fit(X, y)

# Get the coefficients (slopes) for each dependent variable
coefficients = pd.DataFrame(
    pipeline.named_steps['regressor'].coef_,
    index=y.columns,
    columns=pipeline.named_steps['preprocessor'].transformers_[0][1].get_feature_na
)
coefficients.columns = [col.replace("Dominant_Emotion_", "") for col in coefficient

print("Coefficients:\n", coefficients)

# Plotting with Matplotlib
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(14, 6))

# Plot for Daily Usage Minutes
axes[0].bar(coefficients.columns, coefficients.loc['Daily_Usage_Minutes'], color='s
axes[0].set_xlabel('Dominant Emotion', fontsize=12)
axes[0].set_ylabel('Coefficient', fontsize=12)
axes[0].set_title('Impact of Emotion on Daily Usage Minutes', fontsize=14)

# Plot for Daily Usage Minutes
axes[1].bar(coefficients.columns, coefficients.loc['Likes_Received_Per_Day'], color
axes[1].set_xlabel('Dominant Emotion', fontsize=12)
axes[1].set_ylabel('Coefficient', fontsize=12)
axes[1].set_title('Impact of Emotion on Likes Received Per Day', fontsize=14)

# Plot for Overall Engagement
axes[2].bar(coefficients.columns, coefficients.loc['Overall_Engagement'], color='li
axes[2].set_xlabel('Dominant Emotion', fontsize=12)
axes[2].set_ylabel('Coefficient', fontsize=12)
axes[2].set_title('Impact of Emotion on Overall Engagement', fontsize=14)

plt.tight_layout()
plt.show()
```

```
# we analyzed the results but something felt off. We address the possible reasons f

# print("""Impact on Daily Usage Minutes:
# Happiness has the strongest positive impact, suggesting that users who predominan
# Boredom and Neutral emotions have a negative impact, indicating that users feelin
# Anger, Anxiety, and Sadness also show a slight negative association with daily us

# Impact on Overall Engagement:
# Happiness again has the most substantial positive impact, suggesting that happy u
# Boredom has the strongest negative impact, indicating that bored users engage les
# Anxiety has a positive impact, suggesting that anxious users might engage more, p
# Anger, Neutral, and Sadness show a negative association with overall engagement."
```

```
Coefficients:
                          Anger  Anxiety    Boredom  Happiness    Neutral  \
Daily_Usage_Minutes    -3.586134  2.002101 -29.657563  56.913866 -15.836134
Likes_Received_Per_Day -0.157993  1.732052 -22.943158  37.269699 -10.600301
Overall_Engagement      0.299787  0.333069  -3.500337   5.440180  -1.717136

                        Sadness
Daily_Usage_Minutes    -9.836134
Likes_Received_Per_Day -5.300301
Overall_Engagement     -0.855563
```



In [53]:
```
# attempting GLM analysis. As expected, anyone who posts and sends messages are ass
# this makes sense!

x_values = social_df[['Posts_Per_Day','Messages_Sent_Per_Day']]
y = social_df['Daily_Usage_Minutes']

# Add constant for intercept
x_values = sm.add_constant(x_values)

# Fit the GLM
model = sm.GLM(y, x_values, family=sm.families.Gaussian())
results = model.fit()

print('Normal GLM')
print(results.summary())
```

```
gamma_model = sm.GLM(y, x_values, family=sm.families.Gamma(link=sm.families.links.L
gamma_results = gamma_model.fit()

print('Gamma GLM')
print(gamma_results.summary())
```

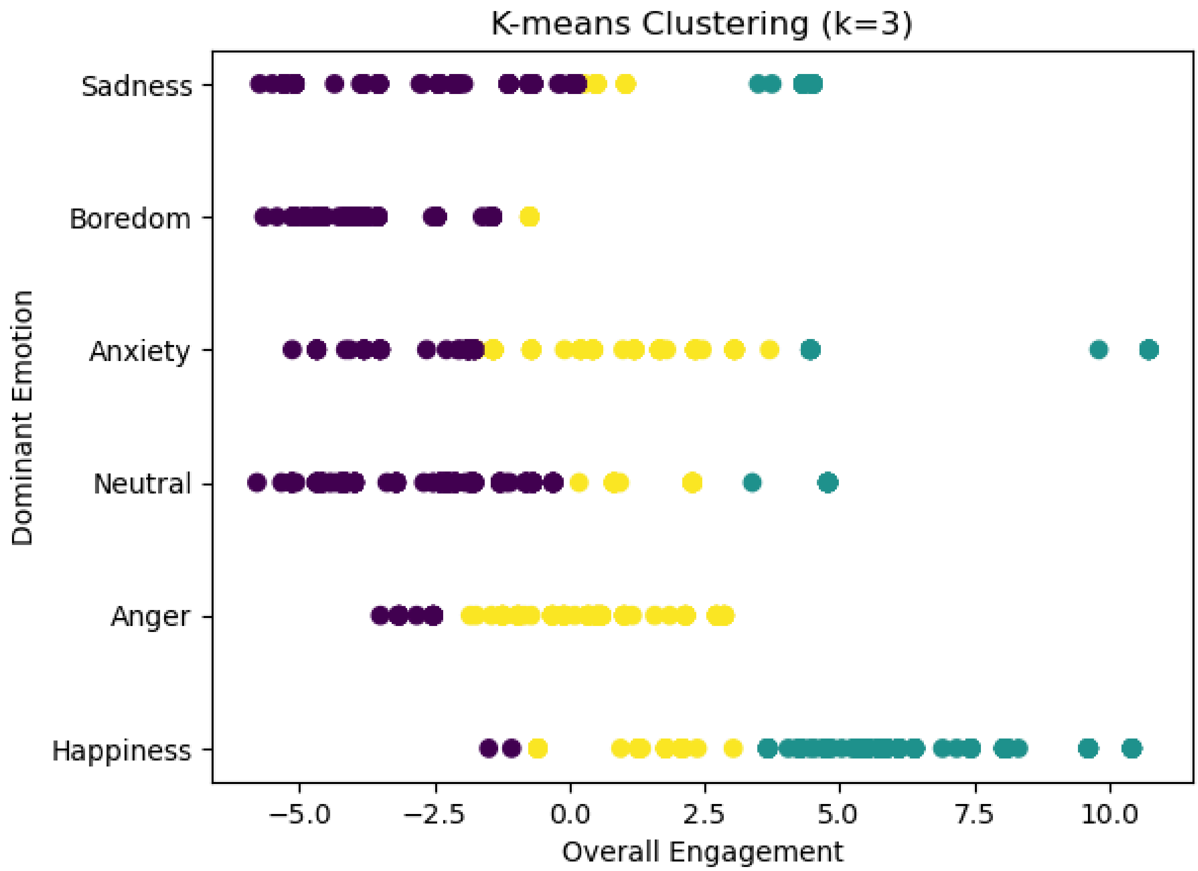
```
Normal GLM
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:     Daily_Usage_Minutes   No. Observations:                 1000
Model:                             GLM   Df Residuals:                      997
Model Family:                 Gaussian   Df Model:                            2
Link Function:                Identity   Scale:                          193.83
Method:                           IRLS   Log-Likelihood:                -4050.9
Date:                 Fri, 21 Jun 2024   Deviance:                   1.9325e+05
Time:                         03:02:59   Pearson chi2:                 1.93e+05
No. Iterations:                      3   Pseudo R-squ. (CS):             0.9989
Covariance Type:             nonrobust
===================================================================================
=====
                          coef    std err          z      P>|z|      [0.025
0.975]
-----------------------------------------------------------------------------------
-----
const                  10.1242      1.357      7.463      0.000       7.465      1
2.783
Posts_Per_Day           7.5595      0.476     15.865      0.000       6.626
8.493
Messages_Sent_Per_Day   2.6915      0.107     25.126      0.000       2.482
2.901
===================================================================================
=====
Gamma GLM
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:     Daily_Usage_Minutes   No. Observations:                 1000
Model:                             GLM   Df Residuals:                      997
Model Family:                    Gamma   Df Model:                            2
Link Function:                     Log   Scale:                        0.022310
Method:                           IRLS   Log-Likelihood:                -4001.1
Date:                 Fri, 21 Jun 2024   Deviance:                       22.083
Time:                         03:02:59   Pearson chi2:                     22.2
No. Iterations:                      9   Pseudo R-squ. (CS):             0.9973
Covariance Type:             nonrobust
===================================================================================
=====
                          coef    std err          z      P>|z|      [0.025
0.975]
-----------------------------------------------------------------------------------
-----
const                   3.6394      0.015    250.055      0.000       3.611
3.668
Posts_Per_Day           0.0726      0.005     14.192      0.000       0.063
0.083
Messages_Sent_Per_Day   0.0274      0.001     23.812      0.000       0.025
0.030
===================================================================================
=====
```

In [54]:
```
# Encode the categorical 'Dominant_Emotion' variable
le = LabelEncoder()
social_df = pd.read_csv("https://raw.githubusercontent.com/gurlv/SocialMediaDataset
```

```
social_df_cluster = social_df
social_df_cluster['Dominant_Emotion_Encoded'] = le.fit_transform(social_df_cluster[
# print the numbers
# Get the unique emotion labels from the original data
emotion_labels = social_df_cluster['Dominant_Emotion'].unique()
print(emotion_labels)
optimal_k = 3

# normalized to prevent one from going over the other. random forest decided on rel
social_df_cluster['Daily_Usage_Minutes_Std'] = (social_df_cluster['Daily_Usage_Minu
social_df_cluster['Likes_Received_Per_Day_Std'] = (social_df_cluster['Likes_Receive
social_df_cluster['Messages_Sent_Per_Day_Std'] = (social_df_cluster['Messages_Sent_
social_df_cluster['Comments_Received_Per_Day'] = (social_df_cluster['Comments_Recei

social_df_cluster['Overall_Engagement'] = social_df_cluster['Daily_Usage_Minutes_St

# Apply K-means clustering
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
social_df['Cluster'] = kmeans.fit_predict(social_df_cluster[['Overall_Engagement',

# Plot the clusters (Likes_Received_Per_Day vs. Dominant_Emotion_Encoded)
plt.scatter(social_df_cluster['Overall_Engagement'], social_df_cluster['Dominant_Em
plt.xlabel('Overall Engagement')
plt.ylabel('Dominant Emotion')
plt.title(f'K-means Clustering (k={optimal_k})')
plt.show()
```

```
['Happiness' 'Anger' 'Neutral' 'Anxiety' 'Boredom' 'Sadness']
```

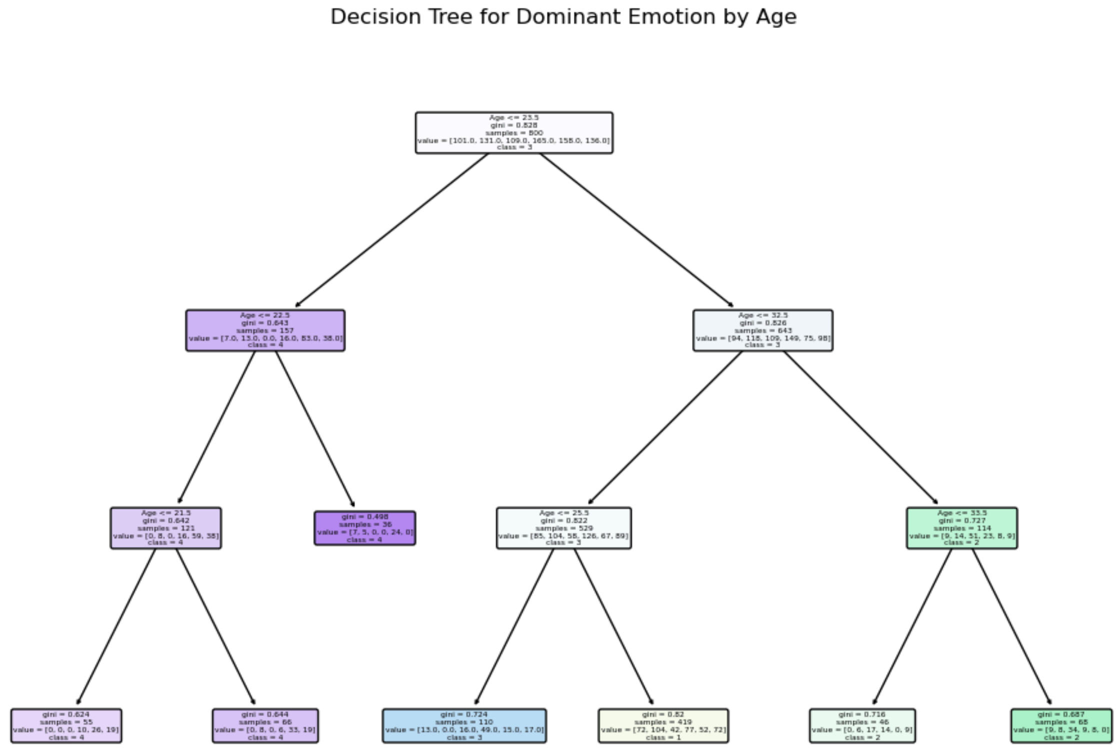
```
C:\Users\infin\anaconda3\envs\Pandas\Lib\site-packages\sklearn\cluster\_kmeans.py:14
46: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when the
re are less chunks than available threads. You can avoid it by setting the environme
nt variable OMP_NUM_THREADS=4.
  warnings.warn(
```



In [55]:
```
# no need to normalize random forest, just get features we care about.

features_random_forest = ['Daily_Usage_Minutes','Likes_Received_Per_Day','Comments_

X = social_df[['Daily_Usage_Minutes','Posts_Per_Day','Likes_Received_Per_Day','Comm
y = social_df["Dominant_Emotion"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

model_random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
model_random_forest.fit(X_train[features_random_forest], y_train)
y_pred = model_random_forest.predict(X_test[features_random_forest])
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       Anger       1.00      1.00      1.00        29
     Anxiety       1.00      0.92      0.96        39
     Boredom       0.89      1.00      0.94        31
   Happiness       1.00      1.00      1.00        35
     Neutral       1.00      0.98      0.99        42
     Sadness       1.00      1.00      1.00        24

    accuracy                           0.98       200
   macro avg       0.98      0.98      0.98       200
weighted avg       0.98      0.98      0.98       200
```

In [56]:
```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split

# decided to give it a try with all the data.
tree_df = social_df

# Convert categorical variable to numerical (using label encoding for simplicity)
tree_df['Dominant_Emotion'] = tree_df['Dominant_Emotion'].astype('category').cat.co

# Split data into features (Age) and target (Dominant_Emotion)
X = tree_df[['Age']]
y = tree_df['Dominant_Emotion']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Create andexpand_more train the decision tree classifier
clf = DecisionTreeClassifier(max_depth=3, random_state=42)  # Adjust max_depth for
clf.fit(X_train, y_train)

# Get the original class names
class_names = tree_df['Dominant_Emotion'].astype('category').cat.categories

# Convert class names to strings
class_names_str = [str(name) for name in class_names]

# Visualize the decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=["Age"], class_names=class_names_str, rou
plt.title("Decision Tree for Dominant Emotion by Age")
plt.show()
```



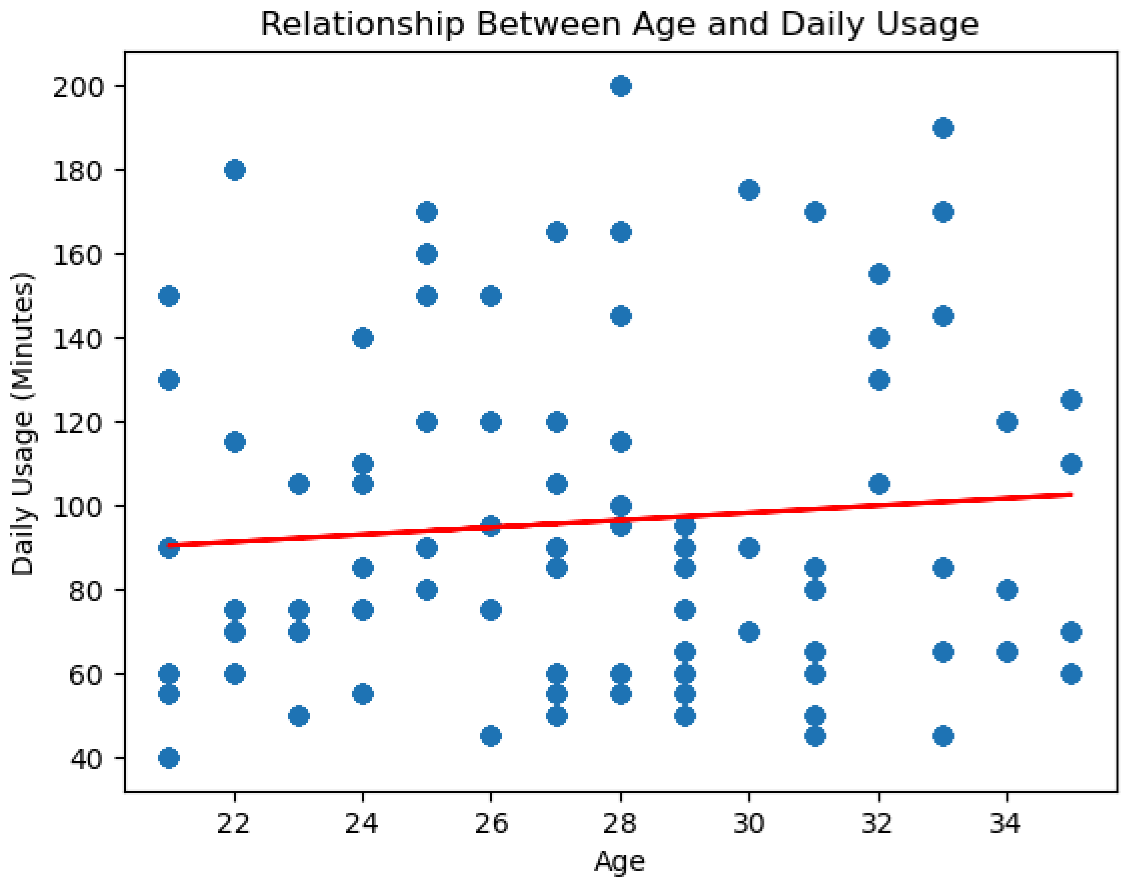
In [57]:

```
#random forest train/test sets. This one implies our data is good at predicting peo

features_random_forest = ['Daily_Usage_Minutes','Likes_Received_Per_Day','Comments_

X = social_df[['Daily_Usage_Minutes','Posts_Per_Day','Likes_Received_Per_Day','Comm
y = social_df["Age"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

model_random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
model_random_forest.fit(X_train[features_random_forest], y_train)
y_pred = model_random_forest.predict(X_test[features_random_forest])
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

          21       1.00      0.80      0.89         5
          22       0.93      0.93      0.93        14
          23       1.00      0.93      0.96        14
          24       1.00      0.95      0.97        19
          25       0.92      1.00      0.96        11
          26       1.00      1.00      1.00        17
          27       0.93      1.00      0.96        26
          28       1.00      0.94      0.97        18
          29       0.85      1.00      0.92        17
          30       1.00      1.00      1.00        14
          31       1.00      0.80      0.89        15
          32       1.00      1.00      1.00         4
          33       1.00      0.93      0.96        14
          34       0.71      1.00      0.83         5
          35       0.86      0.86      0.86         7

    accuracy                           0.95       200
   macro avg       0.95      0.94      0.94       200
weighted avg       0.96      0.95      0.95       200
```

Additional Analysis - Separate from above

In [61]:
```
x = social_df['Age']
y = social_df['Daily_Usage_Minutes']

# Calculate regression line
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)

# Create the scatterplot
plt.scatter(x, y)
plt.xlabel("Age")
plt.ylabel("Daily Usage (Minutes)")
plt.title("Relationship Between Age and Daily Usage")
# Create line values using the original x values and the regression equation
line = slope * x + intercept
# Plot the regression line
plt.plot(x, line, color='red', label=f'y = {slope:.2f}x + {intercept:.2f}')
plt.show()
```



In [ ]:
```
import statsmodels.formula.api as smf
```

In [ ]:
```
social_df.columns
```

In [ ]:
```
fit2 = smf.glm(formula = 'Daily_Usage_Minutes ~ C(Platform)', data = social_df,
               family = sm.families.Gamma()).fit()
print(fit2.summary())
```

In [ ]:
```
social_df = pd.get_dummies(social_df, columns=['Platform'], prefix='Platform')
print(social_df.head())

import statsmodels.api as sm
import statsmodels.formula.api as smf
```

In [ ]:
```
# Encode the 'Dominant_Emotion' variable
social_df['Dominant_Emotion'] = social_df['Dominant_Emotion'].astype('category').ca
```

In [ ]:
```
# Define the formula for the regression
formula = 'Dominant_Emotion ~ Daily_Usage_Minutes + Platform_Instagram'

# Fit the multinomial logistic regression model
model = smf.mnlogit(formula, data=social_df)
result = model.fit()

# Print the summary
print(result.summary())
```

In [ ]:
```
# Define the formula for the regression
formula = 'Dominant_Emotion ~ Daily_Usage_Minutes + Age + Likes_Received_Per_Day'

# Fit the multinomial logistic regression model
model = smf.mnlogit(formula, data=social_df)
result = model.fit()

# Print the summary
print(result.summary())
```

In [ ]:
```

```