Import Statements

In [ ]:
```
import pandas as pd
from sentence_transformers import CrossEncoder
!pip install -q transformers
import pandas as pd
from transformers import pipeline
import torch
from transformers import RobertaModel, RobertaTokenizer
from sklearn.metrics.pairwise import cosine_similarity
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_absolute_error
```

```
/usr/local/lib/python3.10/dist-packages/sentence_transformers/cross_encoder/
CrossEncoder.py:13: TqdmExperimentalWarning: Using `tqdm.autonotebook.tqdm`
in notebook mode. Use `tqdm.tqdm` instead to force console mode (e.g. in jup
yter console)
  from tqdm.autonotebook import tqdm, trange
```

Read articles dataframe in

In [ ]:
```
df_articles = pd.read_csv("all_teams_data.csv")
df_articles = df_articles.rename(columns={'MetaData': 'Content'})
print((df_articles.head()))
```

```
        Team                                              Title  \
0  Air Force  10 reasons why Air Force football will embark ...
1  Air Force  Air Force hosts Robert Morris to start 2023 se...
2  Air Force  To keep wins coming, Air Force Falcons face 's...
3  Air Force  Group of 5 Conferences: Preview and Prediction...
4  Air Force   Air Force Football Announces New Fan Experiences

                                             Content
0                                     URL Not Parsed
1  Air Force Falcons Game vs Robert Morris Game P...
2  To keep wins coming, Air Force Falcons face ‘s...
3  Group of 5 Conferences: Preview and Prediction...
4  Air Force Academy Athletics Air Force Football...
```

Scrape team results

In [ ]:
```
import requests
from bs4 import BeautifulSoup

url = 'https://www.sportsoddshistory.com/cfb-win/?y=2023&sa=cfb&t=win&o=t'
```

```
response = requests.get(url)
if response.status_code == 200:
    soup = BeautifulSoup(response.content, 'html.parser')

    rows = soup.find_all('tr')

    team_data = []
    for row in rows:
        columns = row.find_all('td')

        if len(columns) >= 6:
            team_name = columns[0].get_text(strip=True)
            win_prediction = columns[1].get_text(strip=True)
            odds_minus = columns[2].get_text(strip=True)
            odds_plus = columns[3].get_text(strip=True)
            adj_win_prediction = columns[4].get_text(strip=True)
            bet_type = columns[5].get_text(strip=True)

            team_data.append({
                'Team': team_name,
                'Win Prediction': win_prediction,
                'Odds Minus': odds_minus,
                'Odds Plus': odds_plus,
                'Actual Win': adj_win_prediction,
                'Bet Type': bet_type
            })

    df_teams = pd.DataFrame(team_data)
    df_teams['Actual Win'] = pd.to_numeric(df_teams['Actual Win'], errors='c
    df_teams['Win Prediction'] = pd.to_numeric(df_teams['Win Prediction'], e

    df_teams['Win Diff'] = df_teams['Actual Win'] - df_teams['Win Predictior

    print(df_teams)
else:
    print("Failed to retrieve the webpage:", response.status_code)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-29-e3fec109762b> in <cell line: 6>()
      4 url = 'https://www.sportsoddshistory.com/cfb-win/?y=2023&sa=cfb&t=wi
n&o=t'
      5
----> 6 response = requests.get(url)
      7 if response.status_code == 200:
      8     soup = BeautifulSoup(response.content, 'html.parser')

/usr/local/lib/python3.10/dist-packages/requests/api.py in get(url, params,
**kwargs)
```

```
     71     """
     72 
---> 73     return request("get", url, params=params, **kwargs)
     74 
     75 

/usr/local/lib/python3.10/dist-packages/requests/api.py in request(method, u
rl, **kwargs)
     57     # cases, and look like a memory leak in others.
     58     with sessions.Session() as session:
---> 59         return session.request(method=method, url=url, **kwargs)
     60 
     61 

/usr/local/lib/python3.10/dist-packages/requests/sessions.py in request(self
, method, url, params, data, headers, cookies, files, auth, timeout, allow_r
edirects, proxies, hooks, stream, verify, cert, json)
    587         }
    588         send_kwargs.update(settings)
--> 589         resp = self.send(prep, **send_kwargs)
    590 
    591         return resp

/usr/local/lib/python3.10/dist-packages/requests/sessions.py in send(self, r
equest, **kwargs)
    701 
    702         # Send the request
--> 703         r = adapter.send(request, **kwargs)
    704 
    705         # Total elapsed time of the request (approximately)

/usr/local/lib/python3.10/dist-packages/requests/adapters.py in send(self, r
equest, stream, timeout, verify, cert, proxies)
    665 
    666         try:
--> 667             resp = conn.urlopen(
    668                 method=request.method,
    669                 url=url,

/usr/local/lib/python3.10/dist-packages/urllib3/connectionpool.py in urlopen
(self, method, url, body, headers, retries, redirect, assert_same_host, time
out, pool_timeout, release_conn, chunked, body_pos, preload_content, decode_
content, **response_kw)
    787 
    788             # Make the request on the HTTPConnection object
--> 789             response = self._make_request(
    790                 conn,
    791                 method,

/usr/local/lib/python3.10/dist-packages/urllib3/connectionpool.py in _make_r
```

```
equest(self, conn, method, url, body, headers, retries, timeout, chunked, re
sponse_conn, preload_content, decode_content, enforce_content_length)
    534         # Receive the response from the server
    535         try:
--> 536             response = conn.getresponse()
    537         except (BaseSSLError, OSError) as e:
    538             self._raise_timeout(err=e, url=url, timeout_value=read_t
imeout)

/usr/local/lib/python3.10/dist-packages/urllib3/connection.py in getresponse
(self)
    505 
    506         # Get the response from http.client.HTTPConnection
--> 507         httplib_response = super().getresponse()
    508 
    509         try:

/usr/lib/python3.10/http/client.py in getresponse(self)
   1373         try:
   1374             try:
-> 1375                 response.begin()
   1376             except ConnectionError:
   1377                 self.close()

/usr/lib/python3.10/http/client.py in begin(self)
    316         # read until we get a non-100 response
    317         while True:
--> 318             version, status, reason = self._read_status()
    319             if status != CONTINUE:
    320                 break

/usr/lib/python3.10/http/client.py in _read_status(self)
    277 
    278     def _read_status(self):
--> 279         line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
    280         if len(line) > _MAXLINE:
    281             raise LineTooLong("status line")

/usr/lib/python3.10/socket.py in readinto(self, b)
    703         while True:
    704             try:
--> 705                 return self._sock.recv_into(b)
    706             except timeout:
    707                 self._timeout_occurred = True

/usr/lib/python3.10/ssl.py in recv_into(self, buffer, nbytes, flags)
   1301                   "non-zero flags not allowed in calls to recv_into(
) on %s" %
   1302                   self.__class__)
-> 1303             return self.read(nbytes, buffer)
```

```
   1304         else:
   1305             return super().recv_into(buffer, nbytes, flags)

/usr/lib/python3.10/ssl.py in read(self, len, buffer)
   1157         try:
   1158             if buffer is not None:
-> 1159                 return self._sslobj.read(len, buffer)
   1160             else:
   1161                 return self._sslobj.read(len)

KeyboardInterrupt:
```

In [ ]:
```
df_all = df_teams.merge(df_articles, on='Team', how='inner')
```

In [ ]:
```
print(df_all.columns)
```

Filter out bad data

In [ ]:
```
print((df_all['Content'][1]))
```

In [ ]:
```
df_all = df_all[df_all["Content"] != "URL Not Parsed"]
```

Filter out irrelevant articles by using the embedding of the title

In [ ]:
```
from sentence_transformers import SentenceTransformer
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

reference_query = "Looking ahead to football season preview and predictions.

def get_embedding(text):
    embedding = model.encode([text])
    return embedding

reference_embedding = get_embedding(reference_query)

list_bad_titles = []
avg_embedding = []


def get_similarity(title):
    title_embedding = get_embedding(title)
```

```
    similarity_score = cosine_similarity(reference_embedding, title_embeddin
    return similarity_score

def adjust_score_based_on_keywords(title, score):
    keywords = ['season', 'preview', 'prediction', 'predictions', 'examining
    if any(keyword in title.lower() for keyword in keywords):
        score *= 1.5
    if score < 0.15:
        list_bad_titles.append((title, score))
    avg_embedding.append(score)
    return score

df_all['similarity'] = df_all['Title'].apply(lambda title: adjust_score_base

print(f"Mean of the embeddings: {np.mean(avg_embedding)}")
print(len(list_bad_titles))
print(list_bad_titles)
```

Perform sentiment Analysis on articles

In [ ]:

```
import pandas as pd
from transformers import pipeline

counter = 0


sentiment_pipeline = pipeline("sentiment-analysis")
df_all['Sentiment'] = None

def get_sentiment(text):
    result = sentiment_pipeline(text)
    return result[0]['label'], result[0]['score']

def analyze_content_sentiment(content):
    chunk_size = 1000
    num_chunks = (len(content) // chunk_size) + 1
    total_score = 0
    count = 0

    for i in range(num_chunks):
        chunk = content[i * chunk_size: (i + 1) * chunk_size]
        sentiment_label, sentiment_score = get_sentiment(chunk)

        if sentiment_label == 'NEGATIVE':
            sentiment_score *= -1
        total_score += sentiment_score
        count += 1
```

```
    avg_sentiment_score = total_score / count if count > 0 else 0
    return avg_sentiment_score

for index, row in df_all.iterrows():
    # if index == 80:
    #   break
    content = row['Content']
    avg_sentiment_score = analyze_content_sentiment(content)
    df_all.at[index, 'Sentiment'] = avg_sentiment_score
    counter += 1
    print(counter)

print(df_all)
```

In [ ]:
```
df_all.to_csv("all_teams_data_sentiment.csv")
```

In [ ]:
```
df_team_sentiment_avg = df_all.groupby('Team', as_index=False).agg({
    'Sentiment': 'mean',
    'Win Diff': lambda x: x.mode()[0],
    'Bet Type': lambda x: x.mode()[0],
})

df_team_sentiment_avg.rename(columns={'Sentiment': 'Sentiment_Avg'}, inplace

df_team_sentiment_avg = df_team_sentiment_avg.reset_index(drop=True)

for index, row in df_team_sentiment_avg.iterrows():
    # print(row["Bet Type"])
    if row["Bet Type"] != 'Under' and row["Bet Type"] != 'Over':
        df_team_sentiment_avg.drop(index, inplace=True)

df_team_sentiment_avg.to_csv("all_teams_data_sentiment_full.csv")
```

LOAD FROM HERE AFTER IMPORTS: Sentiment Analysis DF completed

In [ ]:
```
df_team_sentiment_avg = pd.read_csv("all_teams_data_sentiment_full.csv")
print(df_team_sentiment_avg)
```

```
     Unnamed: 0                Team  Sentiment_Avg  Win Diff Bet Type
0             0           Air Force      -0.276178      -0.5    Under
1             1               Akron      -0.476147      -2.0    Under
2             2             Alabama      -0.219753       1.0     Over
3             3   Appalachian State      -0.021634       1.0     Over
4             4             Arizona      -0.110086       4.0     Over
..          ...                 ...            ...       ...      ...
116         116       West Virginia      -0.731522       3.5     Over
117         117    Western Kentucky      -0.234153      -1.5    Under
118         118    Western Michigan      -0.357670       0.5     Over
119         119           Wisconsin      -0.200952      -1.5    Under
120         120             Wyoming      -0.189214       2.0     Over

[121 rows x 5 columns]
```

In [ ]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

df = df_team_sentiment_avg
X = df[['Sentiment_Avg']]
y = df['Win Diff']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rar
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 3.3171764072001624
R-squared: -0.02545207213662004
```

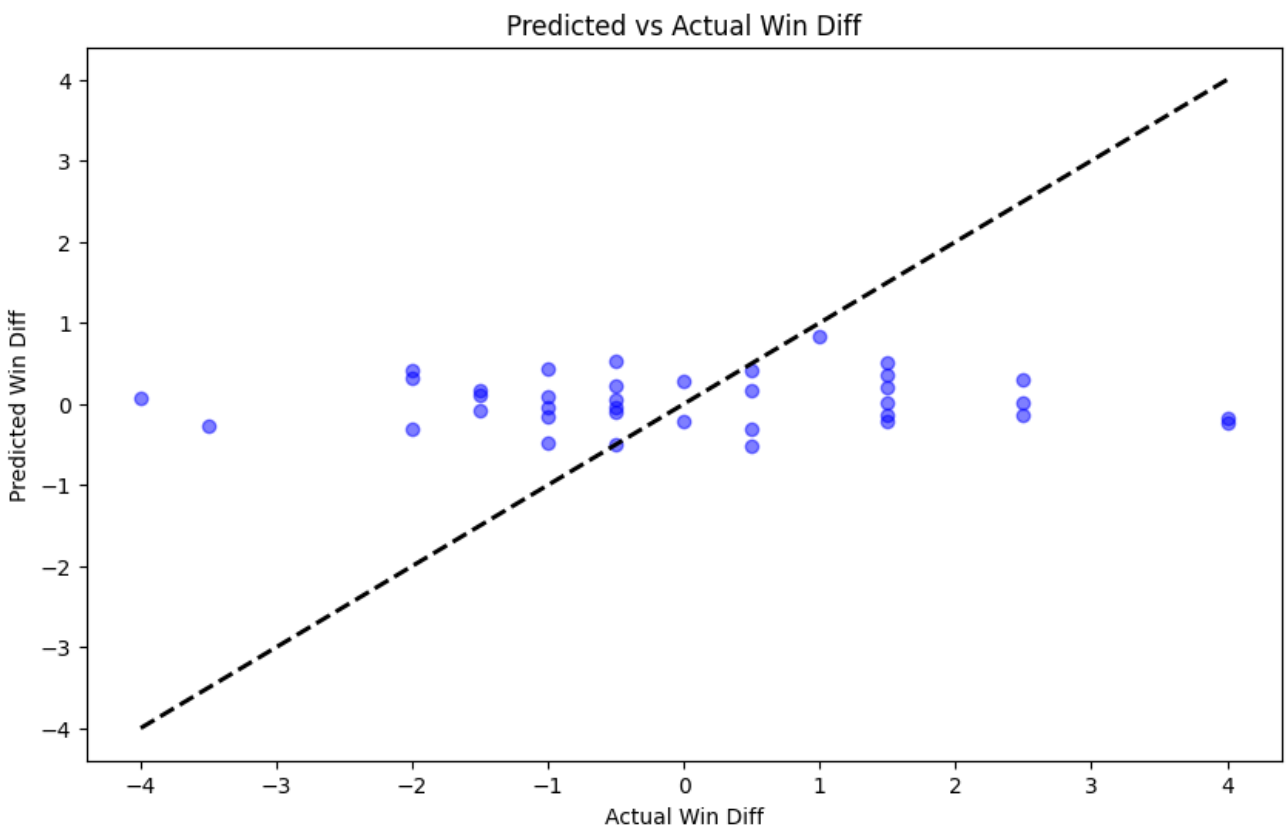
Show the overall accuracy if making a binary over / under prediction

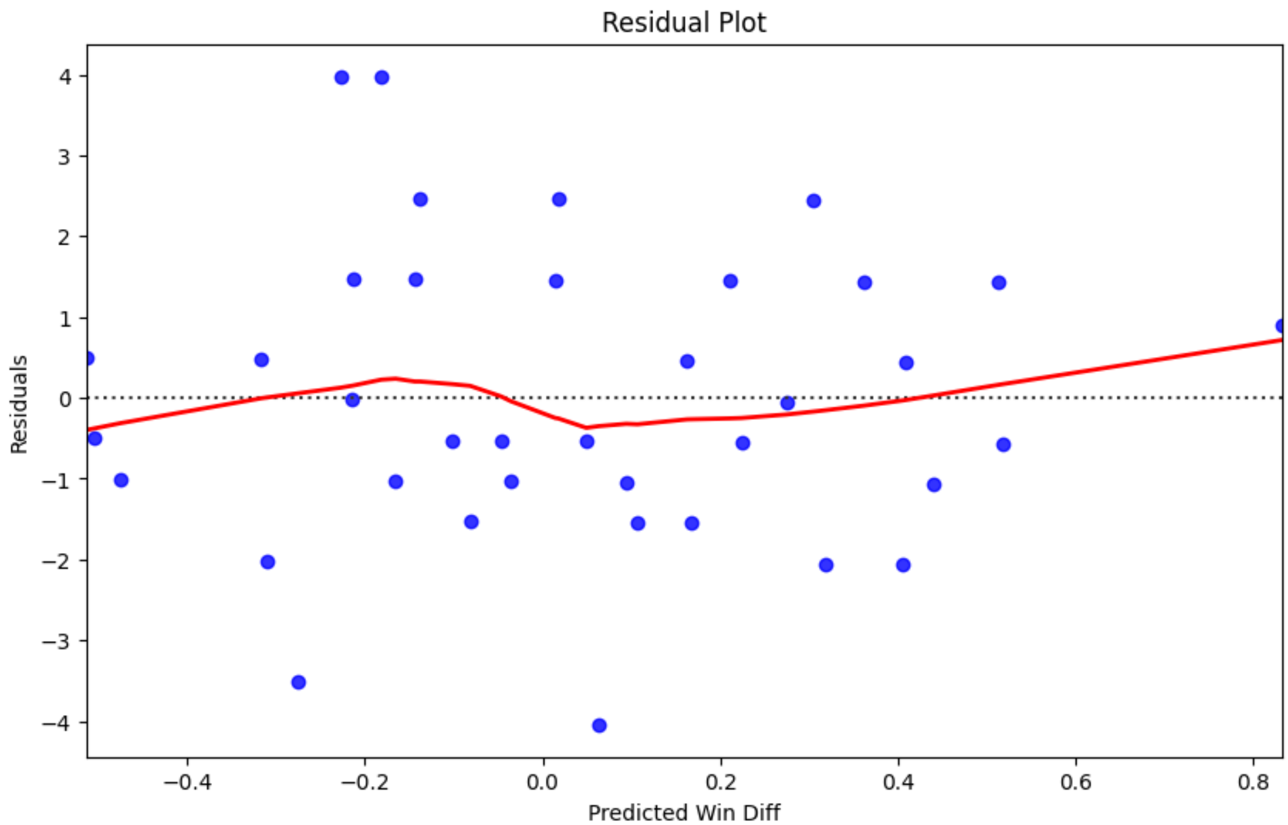
In [ ]:

```
from sklearn.metrics import accuracy_score

y_pred_bet = ['Under' if pred < 0 else 'Over' for pred in y_pred]

accuracy = accuracy_score(y_test.apply(lambda x: 'Under' if x < 0 else 'Over

print(f"Accuracy: {accuracy}")
```

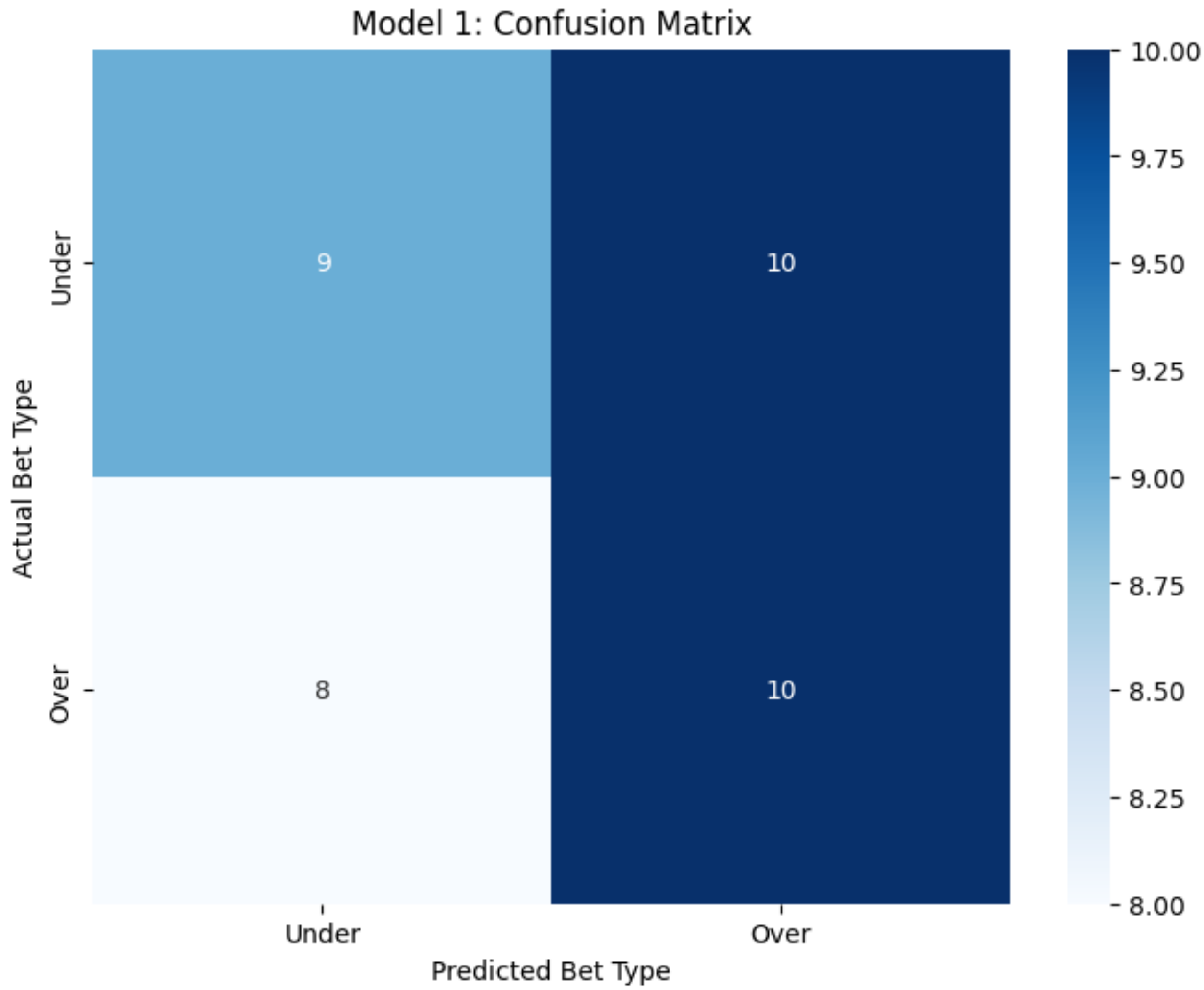
```
Accuracy: 0.5135135135135135
```

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_absolute_error

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',
plt.title("Predicted vs Actual Win Diff")
plt.xlabel("Actual Win Diff")
plt.ylabel("Predicted Win Diff")
plt.show()

residuals = y_test - y_pred
plt.figure(figsize=(10, 6))
sns.residplot(x=y_pred, y=residuals, lowess=True, color='blue', line_kws={'c
plt.title("Residual Plot")
plt.xlabel("Predicted Win Diff")
plt.ylabel("Residuals")
plt.show()
```





In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accura

y_test_bet = y_test.apply(lambda x: 'Under' if x < 0 else 'Over')
y_pred_bet = ['Under' if pred < 0 else 'Over' for pred in y_pred]

cm = confusion_matrix(y_test_bet, y_pred_bet, labels=['Under', 'Over'])

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Under', 'Ov
plt.xlabel("Predicted Bet Type")
plt.ylabel("Actual Bet Type")
plt.title("Model 1: Confusion Matrix")
plt.show()
```



Model number 2: Statistical prediction

Webscrape ESPN for 2022 statistics (year prior)

Format returing starters data

In [ ]:
```
raw_data_returning_starters ="""1. Florida St.
2. Kansas
3. Florida Atlantic
4. Wyoming
5. Michigan
6. Connecticut
7. Texas A&M
8. Boston Coll.
9. Missouri
10. Temple
```

```
11. Toledo
12. Northern Illinois
13. South Alabama
14. USC
15. Massachusetts
16. Utah
17. Navy
18. Florida International
19. Texas
20. North Texas
21. Rice
22. Washington
23. Rutgers
24. Syracuse
25. Coastal Carolina
26. Louisiana Tech
27. Wisconsin
28. Auburn
29. Sam Houston
30. Ole Miss
31. Tulane
32. LSU
33. Duke
34. James Madison
35. Miami (FL)
36. Clemson
37. Middle Tennessee State
38. Virginia Tech
39. Nebraska
40. Miami (OH)
41. Indiana
42. UNLV
43. North Carolina
44. Notre Dame
45. Michigan St.
46. California
47. Ga. Tech
48. Ohio St.
49. Boise St.
50. Louisiana
51. UCF
52. Central Michigan
53. Oregon St.
54. Oregon
55. UTEP
56. Penn St.
57. Purdue
58. Vanderbilt
59. UCLA
60. Army
```

```
61. New Mexico
62. Colorado St.
63. New Mexico State
64. Texas Tech
65. Maryland
66. West Virginia
67. Iowa St.
68. Memphis
69. Tennessee
70. Brigham Young
71. Illinois
72. J'ville St.
73. Kentucky
74. Akron
75. Eastern Michigan
76. Kansas St.
77. Oklahoma
78. Washington State
79. Marshall
80. Georgia
81. Southern Mississippi
82. Houston
83. Troy
84. Fresno St.
85. Bowling Green State
86. Air Force
87. Minnesota
88. Old Dominion
89. Oklahoma St.
90. Arizona
91. Mississippi State
92. Ball St.
93. Colorado
94. Iowa
95. South Florida
96. Northwestern
97. San Diego State
98. North Carolina State
99. Louisville
100. Baylor
101. Arkansas St.
102. South Carolina
103. Utah St.
104. Liberty
105. Western Kentucky
106. Arkansas
107. Florida
108. Nevada
109. Arizona St.
110. Buffalo
```

```
111. San Jose State
112. Wake Forest
113. Louisiana–Monroe
114. Ohio
115. UTSA
116. Virginia
117. SMU
118. TCU
119. Pitt
120. Charlotte
121. Western Michigan
122. Hawaii
123. Ga. Southern
124. Cincinnati
125. Alabama
126. Tulsa
127. UAB
128. Texas St.
129. Stanford
130. East Carolina
131. Appalachian State
132. Georgia St.
133. Kent St."""

returning_starters = []
for line in raw_data_returning_starters.split('\n'):
  temp = line
  temp = temp.replace("St.", 'State')
  temp = temp.replace("Coll", 'College')
  temp = temp.replace('.', ':', 1)
  temp = temp.replace('.', '')
  temp = temp.replace('Ga', 'Georgia')
  temp = temp.replace('GA', 'Georgia')
  temp  = temp.split(':')
  temp[0] = 1 - (int(temp[0]) / 133.01)
  returning_starters.append((temp[1].strip(), temp[0])) ## Team, Percentile
returning_starters = returning_starters[:len(returning_starters) - 1]
print(returning_starters)
```

```
[('Florida State', 0.9924817682880986), ('Kansas', 0.9849635365761973), ('Fl
orida Atlantic', 0.9774453048642959), ('Wyoming', 0.9699270731523946), ('Mic
higan', 0.9624088414404932), ('Connecticut', 0.9548906097285919), ('Texas A&
M', 0.9473723780166905), ('Boston College', 0.9398541463047891), ('Missouri'
, 0.9323359145928878), ('Temple', 0.9248176828809864), ('Toledo', 0.91729945
11690851), ('Northern Illinois', 0.9097812194571837), ('South Alabama', 0.90
22629877452824), ('USC', 0.894744756033381), ('Massachusetts', 0.88722652432
14796), ('Utah', 0.8797082926095783), ('Navy', 0.8721900608976769), ('Florid
a International', 0.8646718291857756), ('Texas', 0.8571535974738741), ('Nort
h Texas', 0.8496353657619727), ('Rice', 0.8421171340500714), ('Washington',
0.83459890233817), ('Rutgers', 0.8270806706262687), ('Syracuse', 0.819562438
```

9143673), ('Coastal Carolina', 0.812044207202466), ('Louisiana Tech', 0.8045
259754905646), ('Wisconsin', 0.7970077437786632), ('Auburn', 0.7894895120667
619), ('Sam Houston', 0.7819712803548605), ('Ole Miss', 0.7744530486429592),
('Tulane', 0.7669348169310578), ('LSU', 0.7594165852191564), ('Duke', 0.7518
983535072551), ('James Madison', 0.7443801217953537), ('Miami (FL)', 0.73686
18900834524), ('Clemson', 0.729343658371551), ('Middle Tennessee State', 0.7
218254266596495), ('Virginia Tech', 0.7143071949477482), ('Nebraska', 0.7067
889632358468), ('Miami (OH)', 0.6992707315239455), ('Indiana', 0.69175249981
20441), ('UNLV', 0.6842342681001428), ('North Carolina', 0.6767160363882414)
, ('Notre Dame', 0.66919780467634), ('Michigan State', 0.6616795729644387),
('California', 0.6541613412525373), ('Georgia Tech', 0.646643109540636), ('O
hio State', 0.6391248778287346), ('Boise State', 0.6316066461168333), ('Loui
siana', 0.6240884144049319), ('UCF', 0.6165701826930305), ('Central Michigan
', 0.6090519509811292), ('Oregon State', 0.6015337192692278), ('Oregon', 0.5
940154875573265), ('UTEP', 0.5864972558454251), ('Penn State', 0.57897902413
35238), ('Purdue', 0.5714607924216224), ('Vanderbilt', 0.563942560709721), (
'UCLA', 0.5564243289978197), ('Army', 0.5489060972859183), ('New Mexico', 0.
541387865574017), ('Colorado State', 0.5338696338621156), ('New Mexico State
', 0.5263514021502143), ('Texas Tech', 0.5188331704383129), ('Maryland', 0.5
113149387264115), ('West Virginia', 0.5037967070145102), ('Iowa State', 0.49
62784753026088), ('Memphis', 0.48876024359070747), ('Tennessee', 0.481242011
8788061), ('Brigham Young', 0.47372378016690475), ('Illinois', 0.46620554845
50034), ("J'ville State", 0.45868731674310204), ('Kentucky', 0.4511690850312
007), ('Akron', 0.4436508533192992), ('Eastern Michigan', 0.4361326216073978
5), ('Kansas State', 0.4286143898954965), ('Oklahoma', 0.42109615818359514),
('Washington State', 0.4135779264716938), ('Marshall', 0.4060596947597924),
('Georgia', 0.39854146304789106), ('Southern Mississippi', 0.391023231335989
7), ('Houston', 0.38350499962408835), ('Troy', 0.375986767912187), ('Fresno
State', 0.36846853620028563), ('Bowling Green State', 0.3609503044883843), (
'Air Force', 0.3534320727764829), ('Minnesota', 0.34591384106458156), ('Old
Dominion', 0.3383956093526802), ('Oklahoma State', 0.33087737764077885), ('A
rizona', 0.3233591459288775), ('Mississippi State', 0.31584091421697613), ('
Ball State', 0.3083226825050748), ('Colorado', 0.3008044507931734), ('Iowa',
0.29328621908127206), ('South Florida', 0.2857679873693707), ('Northwestern'
, 0.27824975565746934), ('San Diego State', 0.270731523945568), ('North Caro
lina State', 0.26321329223366663), ('Louisville', 0.25569506052176527), ('Ba
ylor', 0.24817682880986391), ('Arkansas State', 0.24065859709796256), ('Sout
h Carolina', 0.2331403653860612), ('Utah State', 0.22562213367415973), ('Lib
erty', 0.21810390196225837), ('Western Kentucky', 0.21058567025035702), ('Ar
kansas', 0.20306743853845566), ('Florida', 0.1955492068265543), ('Nevada', 0
.18803097511465294), ('Arizona State', 0.18051274340275159), ('Buffalo', 0.1
7299451169085023), ('San Jose State', 0.16547627997894887), ('Wake Forest',
0.1579580482670475), ('Louisiana-Monroe', 0.15043981655514616), ('Ohio', 0.1
429215848432448), ('UTSA', 0.13540335313134344), ('Virginia', 0.127885121419
44208), ('SMU', 0.12036688970754073), ('TCU', 0.11284865799563937), ('Pitt',
0.10533042628373801), ('Charlotte', 0.09781219457183665), ('Western Michigan
', 0.0902939628599353), ('Hawaii', 0.08277573114803394), ('Georgia Southern'
, 0.07525749943613258), ('Cincinnati', 0.06773926772423122), ('Alabama', 0.0
6022103601232986 5), ('Tulsa', 0.05270280430042851), ('UAB', 0.04518457258852
715), ('Texas State', 0.03766634087662579), ('Stanford', 0.03014810916472443

```
5), ('East Carolina', 0.022629877452823077), ('Appalachian State', 0.0151116
4574092172), ('Georgia State', 0.007593414029020251)]
```

In [ ]:
```
pd.set_option('display.max_rows', None)  # Show all rows
pd.set_option('display.max_columns', None)  # Show all columns
pd.set_option('display.width', None)  # Avoid line wrapping
pd.set_option('display.max_colwidth', None)  # Show full content in each col


df_team_stats_offense = pd.read_csv('2022_offensive_stats.csv')
df_team_stats_defense = pd.read_csv('2022_defensive_stats.csv')
```

In [ ]:
```
# Rename specific columns for offense
new_columns_offense = df_team_stats_offense.columns.tolist()
new_columns_offense[0] = 'Team Name'
for i in range(1, len(new_columns_offense)):
    new_columns_offense[i] = 'Offense: ' + new_columns_offense[i]
df_team_stats_offense.columns = new_columns_offense
```

In [ ]:
```
# Rename specific columns for defense
new_columns_defense = df_team_stats_defense.columns.tolist()
new_columns_defense[0] = 'Team Name'
for i in range(1, len(new_columns_defense)):
    new_columns_defense[i] = 'Defense: ' + new_columns_defense[i]
df_team_stats_defense.columns = new_columns_defense
print(df_team_stats_offense)
```

```
                 Team Name  Offense: Passing  Offense: Rushing  Offense: To
tal Offense  \
0                Tennessee                13              46.1
22.3
1              Ohio State                 13              44.2
21.1
2                      USC                14              41.4
24.6
3                  Alabama                13              41.1
21.5
4                  Georgia                15              41.1
22.4
5                 Michigan                14              40.4
17.0
6               Washington                13              39.7
28.7
7                     UCLA                13              39.2
22.5
8                   Oregon                13              38.8
23.7
9                      TCU                15              38.8
19.5
10                    Utah                14              38.6
```

20.6

11	SMU	13	37.2	25.6
12	James Madison	11	37.0	18.3
13	UTSA	14	36.8	24.4
14	Western Kentucky	14	36.4	28.6
15	Florida State	13	36.1	19.2
16	Houston	13	36.1	25.9
17	Wake Forest	13	36.1	23.2
18	Tulane	14	36.0	17.2
19	Penn State	13	35.8	20.8
20	Kansas	13	35.6	18.6
21	Memphis	13	35.3	22.3
22	Appalachian State	12	34.9	18.8
23	LSU	14	34.5	23.0
24	Texas	13	34.5	19.1
25	North Carolina	14	34.4	24.6
26	Texas Tech	13	34.2	26.8
27	North Texas	14	33.8	17.2
28	Ole Miss	13	33.5	18.4
29	Clemson	14	33.2	21.3
30	UCF	14	32.9	20.8
31	Duke	13	32.8	19.6
32	Oklahoma	13	32.8	19.3
33	Georgia Southern	13	32.7	28.8
34	Arkansas	13	32.5	17.8
35	East Carolina	13	32.5	

```
24.8
36      Kansas State        14    32.3
17.3
37      Baylor              13    32.2
19.0
38      Oregon State        13    32.2
15.5
39      South Carolina      13    32.2
21.6
40      Notre Dame          13    31.8
16.2
41      Ohio                14    31.8
21.6
42      Mississippi State   13    31.6
32.9
43      Brigham Young       13    31.3
19.8
44      Pitt                13    31.3
17.2
45      Toledo              14    31.3
17.6
46      South Alabama       13    31.2
22.3
47      Arizona             12    30.8
23.6
48      Fresno State        14    30.6
24.0
49      Oklahoma State      13    30.6
22.5
50      Tulsa               12    30.6
20.1
51      West Virginia       12    30.6
21.2
52      UAB                 13    30.1
14.9
53      Georgia State       12    30.0
14.5
54      Eastern Michigan    13    29.8
18.8
55      Florida Atlantic    12    29.8
18.1
56      Boise State         14    29.5
16.4
57      Florida             13    29.5
15.3
58      Cincinnati          13    29.2
19.1
59      Coastal Carolina    13    29.1
18.2
60      Louisiana Tech      12    29.0
```

21.2
61 Middle Tennessee State 13 28.8
26.2
62 Army 12 28.6
3.5
63 Buffalo 13 28.5
21.0
64 Kent State 12 28.4
16.2
65 Maryland 13 28.2
22.6
66 Minnesota 13 28.2
13.2
67 South Florida 12 28.0
15.7
68 Air Force 13 27.8
3.2
69 Syracuse 13 27.7
17.5
70 Liberty 13 27.5
17.7
71 San Jose State 12 27.4
21.7
72 Northern Illinois 12 27.3
15.4
73 Louisville 13 26.9
16.8
74 Purdue 14 26.6
26.4
75 UNLV 12 26.3
18.9
76 Wisconsin 13 26.3
14.5
77 Louisiana 13 26.2
19.5
78 Arizona State 12 26.1
22.5
79 Washington State 13 26.1
24.8
80 Troy 14 25.6
17.9
81 New Mexico State 13 25.5
12.2
82 Southern Mississippi 13 25.3
14.7
83 Rice 13 25.2
17.9
84 Arkansas State 12 25.0
20.6
85 Auburn 12 24.8

13.3

86	Central Michigan	12	24.8	18.3
87	Missouri	13	24.8	19.3
88	Vanderbilt	12	24.6	16.7
89	Marshall	13	24.5	17.3
90	Charlotte	12	24.4	21.4
91	Michigan State	12	24.4	21.5
92	UTEP	12	24.4	16.8
93	North Carolina State	13	24.3	20.7
94	Illinois	13	24.2	21.2
95	California	12	23.9	24.3
96	Miami (FL)	12	23.6	21.3
97	Bowling Green State	13	23.5	20.5
98	Ball State	12	23.3	23.9
99	Indiana	12	23.3	21.8
100	Texas A&M	12	22.8	18.3
101	Nebraska	12	22.6	17.1
102	Louisiana–Monroe	12	22.3	18.4
103	Utah State	13	22.2	17.7
104	Navy	12	21.9	4.5
105	Temple	12	21.9	24.0
106	Akron	12	21.8	26.9
107	San Diego State	13	21.5	14.3
108	Stanford	12	21.3	22.9
109	Wyoming	13	21.2	12.0
110	Texas State	12	21.1	

```
22.9
111                 Kentucky                13               20.4
17.2
112               Iowa State                12               20.2
26.1
113               Miami (OH)                13               20.2
14.2
114                   Hawaii                13               19.8
20.1
115             Old Dominion                12               19.5
19.8
116              Connecticut                13               19.4
12.1
117            Virginia Tech                11               19.3
18.5
118         Western Michigan                12               19.0
13.9
119                   Nevada                12               18.8
18.1
120   Florida International                 12               18.7
22.6
121           Boston College                12               17.8
21.8
122                     Iowa                13               17.7
14.8
123                  Rutgers                12               17.4
14.0
124             Georgia Tech                12               17.2
18.3
125                 Virginia                10               17.0
18.5
126                 Colorado                12               15.4
14.9
127             Northwestern                12               13.8
20.4
128           Colorado State                12               13.2
16.9
129               New Mexico                12               13.1
11.4
130            Massachusetts                12               12.5
11.3

     Offense: First Downs  Offense: Penalties  Offense: Turnovers  Offense:
Rk  Offense: School  \
0                    32.5                68.7               326.1          2
.9              40.2
1                    31.5                66.8               298.3          3
.2              35.8
2                    36.8                67.0               335.4          3
.1              33.8
```

3		33.7	63.9	281.5	2
.8	35.2				
4		32.9	68.2	295.9	2
.1	37.1				
5		26.4	64.3	219.9	1
.7	42.9				
6		44.2	64.9	369.8	2
.5	30.8				
7		32.3	69.5	266.4	2
.2	39.5				
8		33.2	71.3	284.8	2
.3	39.1				
9		30.3	64.3	261.7	2
.2	37.7				
10		31.9	64.6	249.2	2
.2	40.0				
11		39.5	64.9	316.7	2
.8	37.8				
12		29.8	61.3	265.7	2
.5	43.5				
13		35.9	67.8	300.7	2
.4	38.8				
14		44.5	64.4	352.2	3
.1	29.6				
15		30.4	63.0	270.2	2
.2	39.2				
16		38.5	67.4	314.0	3
.1	30.9				
17		36.4	63.8	311.9	3
.3	38.8				
18		27.1	63.6	236.6	2
.1	39.8				
19		32.5	64.0	252.5	2
.2	37.6				
20		28.5	65.2	254.4	2
.5	34.3				
21		34.9	63.9	279.2	1
.8	37.9				
22		29.9	63.0	250.9	2
.3	39.5				
23		34.3	67.1	269.3	1
.6	36.9				
24		31.2	61.2	241.4	1
.7	36.2				
25		37.6	65.4	309.3	2
.7	36.3				
26		43.8	61.2	302.5	2
.0	40.4				
27		30.9	55.8	261.9	2
.4	38.3				

28		29.8	61.8	239.8	1
.7	47.2				
29		34.3	62.1	232.4	1
.7	39.0				
30		32.2	64.5	241.2	1
.6	43.9				
31		30.8	63.8	231.5	1
.5	37.2				
32		31.3	61.7	254.6	2
.1	44.7				
33		47.1	61.1	329.8	2
.2	28.5				
34		27.5	64.7	233.8	2
.0	46.5				
35		37.2	66.7	290.5	2
.2	32.5				
36		27.8	62.2	210.5	1
.5	40.6				
37		30.2	63.0	231.5	1
.5	41.2				
38		24.8	62.4	199.5	1
.2	40.3				
39		32.4	66.7	260.0	1
.8	31.6				
40		26.0	62.4	207.1	1
.9	40.9				
41		33.1	65.1	278.1	2
.0	35.2				
42		48.8	67.5	311.2	2
.8	22.7				
43		30.2	65.6	249.7	2
.5	33.7				
44		30.1	57.0	222.8	1
.0	40.1				
45		31.4	56.3	225.2	2
.2	41.5				
46		34.3	65.0	267.6	2
.2	38.8				
47		38.3	61.7	318.4	2
.2	29.3				
48		33.8	71.0	270.6	1
.6	33.3				
49		40.3	55.7	279.5	1
.8	36.6				
50		34.5	58.2	273.2	2
.3	35.8				
51		35.4	59.8	227.5	1
.7	37.1				
52		24.7	60.4	202.6	1
.2	40.6				

53		24.8	58.4	203.6	1
.5	48.1				
54		30.2	62.2	223.9	1
.8	38.7				
55		31.6	57.3	227.2	2
.1	40.9				
56		27.5	59.5	190.6	1
.4	39.4				
57		28.2	54.4	223.8	1
.4	36.3				
58		31.9	59.8	242.7	1
.7	32.8				
59		27.8	65.7	246.2	2
.1	39.5				
60		36.2	58.5	267.3	2
.1	33.3				
61		39.4	66.6	264.8	1
.6	34.8				
62		8.7	40.4	76.7	0
.4	54.0				
63		35.6	59.0	235.3	1
.4	41.5				
64		29.1	55.6	215.2	1
.4	43.1				
65		34.3	65.9	260.2	1
.6	36.1				
66		21.6	61.2	182.2	0
.9	44.7				
67		27.2	57.7	192.9	1
.5	36.7				
68		6.7	47.1	70.5	0
.6	61.7				
69		28.5	61.6	231.9	1
.5	34.2				
70		30.5	58.1	216.7	1
.6	37.9				
71		35.9	60.3	272.5	1
.9	28.6				
72		27.3	56.4	182.7	1
.5	39.8				
73		28.3	59.2	205.5	1
.0	39.8				
74		41.6	63.3	278.8	1
.9	33.3				
75		30.1	62.9	215.0	1
.3	33.5				
76		25.1	57.7	183.8	1
.6	38.5				
77		34.0	57.2	222.5	2
.1	35.1				

78		33.6	67.0	251.9	1
.5	32.1				
79		38.5	64.6	253.8	1
.8	28.2				
80		28.6	62.6	242.9	1
.4	35.4				
81		23.5	52.1	169.2	1
.4	34.8				
82		26.7	55.0	207.5	1
.5	36.8				
83		31.0	57.8	232.9	1
.8	35.3				
84		33.0	62.4	226.6	1
.3	31.8				
85		25.7	51.6	172.7	0
.8	40.7				
86		32.3	56.6	208.1	1
.3	36.9				
87		30.5	63.4	214.1	1
.1	37.7				
88		28.8	57.8	187.3	1
.8	37.1				
89		28.2	61.3	192.0	1
.2	45.9				
90		35.8	59.9	271.0	2
.3	30.9				
91		34.3	62.6	240.0	1
.9	30.1				
92		31.7	53.2	217.7	1
.3	38.7				
93		36.2	57.1	226.1	1
.8	34.0				
94		30.4	69.6	211.8	1
.2	41.8				
95		38.8	62.4	268.0	1
.9	26.6				
96		34.4	61.7	239.0	1
.5	34.4				
97		33.7	60.7	234.7	1
.9	31.9				
98		40.1	59.7	228.5	1
.5	34.8				
99		40.3	54.0	217.4	1
.3	33.0				
100		32.6	56.3	219.4	1
.5	30.5				
101		28.3	60.3	220.8	1
.4	35.0				
102		27.7	66.6	205.3	1
.3	34.8				

103.4	42.3	30.6	57.8	195.2	1
104.8	59.0	10.5	42.9	85.7	0
105.5	26.5	40.3	59.5	268.1	1
106.3	31.1	41.8	64.3	283.1	1
107.2	36.2	26.4	54.2	181.5	1
108.3	31.3	37.3	61.4	254.3	1
109.8	36.8	23.4	51.3	132.2	0
110.6	31.9	37.0	61.9	221.1	1
111.6	35.6	26.8	64.2	208.5	1
112.6	32.8	39.9	65.3	261.8	1
113.2	37.2	25.9	54.6	165.5	1
114.0	31.2	37.7	53.3	205.5	1
115.6	25.9	35.1	56.3	247.2	1
116.8	40.4	21.2	56.9	111.9	0
117.8	35.3	31.5	58.7	203.3	0
118.8	38.8	27.9	49.9	165.7	0
119.6	35.6	33.2	54.5	185.6	0
120.3	29.7	39.0	57.9	218.8	1
121.8	30.2	36.7	59.5	247.1	1
122.5	32.5	26.8	55.0	156.7	0
123.9	35.8	27.7	50.6	153.9	0
124.8	35.5	31.8	57.5	192.2	0
125.7	32.9	34.1	54.3	221.0	0
126.8	32.3	30.0	49.7	172.9	0
127.8	39.1	34.8	58.8	210.3	0

```
128                 26.8                63.0                196.8         1
.0              32.1
129                 21.3                53.7                104.9         0
.3              36.6
130                 23.1                49.1                116.9         0
.3              43.3

     Offense: G  Offense: Pts  Offense: Cmp  Offense: Att  Offense: Pct  Off
ense: TD  \
0         199.5           5.0           3.1          72.7         525.5
11.9
1         192.4           5.4           2.3          67.4         490.7
13.2
2         171.1           5.1           2.1          70.6         506.6
13.9
3         195.5           5.6           2.0          68.9         477.1
12.6
4         205.3           5.5           2.9          70.0         501.2
13.8
5         238.9           5.6           2.9          69.3         458.8
9.3
6         146.0           4.7           2.5          75.0         515.8
17.1
7         237.2           6.0           2.8          71.8         503.6
12.2
8         215.8           5.5           2.6          72.3         500.5
12.2
9         193.3           5.1           2.5          68.0         455.0
10.3
10        217.6           5.4           2.7          71.9         466.9
11.1
11        156.2           4.1           2.0          77.3         472.8
13.4
12        186.7           4.3           2.2          73.4         452.5
11.1
13        175.3           4.5           2.0          74.7         476.0
14.8
14        145.1           4.9           1.1          74.1         497.3
14.9
15        214.1           5.5           2.5          69.5         484.2
11.0
16        141.8           4.6           1.3          69.4         455.8
14.9
17        130.5           3.4           1.3          75.2         442.4
14.8
18        204.8           5.1           2.4          66.9         441.4
9.8
19        181.1           4.8           2.2          70.1         433.6
11.8
20        184.2           5.4           2.2          62.8         438.6
```

```
11.3
21          142.2           3.8            2.3            72.8            421.4
12.0
22          204.4           5.2            2.3            69.4            455.3
10.5
23          183.9           5.0            2.8            71.2            453.1
12.6
24          188.2           5.2            2.2            67.3            429.5
11.3
25          153.5           4.2            1.7            73.9            462.8
13.9
26          158.9           3.9            2.2            84.2            461.4
13.9
27          199.9           5.2            1.7            69.1            461.8
11.0
28          256.6           5.4            2.5            77.0            496.4
9.8
29          177.9           4.6            2.2            73.3            410.3
10.4
30          228.4           5.2            2.5            76.1            469.6
11.2
31          184.2           4.9            2.4            68.0            415.7
9.8
32          219.4           4.9            2.2            76.0            474.0
10.6
33          136.8           4.8            1.8            75.5            466.7
14.8
34          236.7           5.1            2.1            74.0            470.5
9.7
35          170.6           5.3            1.9            69.7            461.1
12.5
36          208.3           5.1            2.3            68.4            418.8
9.4
37          182.4           4.4            2.6            71.3            413.8
10.6
38          196.5           4.9            2.8            65.1            396.0
9.5
39          118.8           3.8            2.1            64.0            378.8
9.7
40          189.1           4.6            1.9            66.9            396.2
9.5
41          143.1           4.1            1.6            68.4            421.2
12.0
42           81.7           3.6            0.9            71.5            392.9
15.2
43          176.6           5.2            1.5            63.8            426.3
10.7
44          183.0           4.6            2.3            70.2            405.8
9.2
45          177.4           4.3            1.6            72.9            402.6
```

						10.4
46	156.8	4.0	1.5	73.2	424.4	12.8
47	143.5	4.9	1.6	67.6	461.9	14.2
48	132.2	4.0	2.0	67.1	402.8	12.5
49	125.5	3.4	1.5	76.9	405.1	12.5
50	138.7	3.9	1.5	70.3	411.8	11.3
51	171.5	4.6	2.0	72.5	399.0	10.9
52	235.0	5.8	2.5	65.3	437.6	8.4
53	213.4	4.4	2.2	72.9	417.0	8.8
54	140.8	3.6	1.9	68.8	364.7	9.8
55	184.3	4.5	1.6	72.5	411.5	10.7
56	196.5	5.0	1.9	66.9	387.1	8.6
57	200.2	5.5	2.2	64.5	424.1	9.9
58	129.7	4.0	1.5	64.7	372.4	11.2
59	159.3	4.0	1.7	67.2	405.5	10.8
60	124.8	3.8	1.2	69.4	392.0	10.1
61	108.5	3.1	1.5	74.2	373.3	11.7
62	289.4	5.4	3.2	62.7	366.1	2.3
63	141.9	3.4	1.7	77.1	377.2	11.0
64	202.1	4.7	1.9	72.2	417.3	10.0
65	141.3	3.9	1.8	70.4	401.5	11.5
66	207.3	4.6	2.5	66.3	389.5	8.1
67	197.8	5.4	2.1	63.8	390.8	8.8
68	326.7	5.3	2.8	68.4	397.2	2.5
69	142.5	4.2	1.6	62.6	374.4	10.7
70	172.7	4.6	1.8	68.4	389.4	

```
10.1
71     95.8     3.3     1.6     64.5     368.3
11.9
72    190.4     4.8     1.6     67.2     373.1
8.6
73    200.6     5.0     1.9     68.2     406.2
9.6
74    121.0     3.6     1.2     74.9     399.8
12.7
75    137.1     4.1     1.5     63.6     352.1
9.0
76    179.8     4.7     1.6     63.6     363.6
8.5
77    141.8     4.0     0.8     69.1     364.3
9.6
78    135.3     4.2     1.8     65.7     387.2
12.1
79    106.8     3.8     1.3     66.7     360.7
11.5
80    117.0     3.3     1.5     64.1     359.9
10.4
81    161.5     4.6     1.9     58.3     330.6
7.0
82    141.8     3.9     1.2     63.5     349.3
8.4
83    141.5     4.0     1.3     66.3     374.4
10.0
84     88.2     2.8     1.3     64.8     314.8
10.3
85    205.8     5.1     2.1     66.3     378.5
7.3
86    160.4     4.3     1.8     69.2     368.5
9.8
87    155.7     4.1     1.5     68.2     369.8
9.5
88    159.9     4.3     1.1     65.9     347.3
7.8
89    205.9     4.5     1.5     74.2     397.9
7.8
90    116.1     3.8     1.1     66.7     387.1
12.2
91    113.0     3.8     1.3     64.4     353.0
11.2
92    167.2     4.3     1.2     70.3     384.8
9.9
93    113.8     3.3     0.6     70.2     339.8
10.5
94    166.2     4.0     1.4     72.2     378.0
10.4
95     96.6     3.6     0.8     65.4     364.6
```

```
11.6
96      127.9       3.7       1.0       68.8      366.9
10.9
97      100.2       3.1       0.6       65.6      334.8
9.7
98      155.7       4.5       1.2       74.8      384.2
11.3
99      110.8       3.4       1.3       73.3      328.3
9.8
100     141.5       4.6       1.0       63.1      360.9
9.2
101     123.3       3.5       1.4       63.3      344.2
9.3
102     120.7       3.5       1.3       62.5      326.0
8.4
103     159.2       3.8       1.1       72.9      354.3
8.9
104     241.2       4.1       1.8       69.5      326.8
2.9
105      82.4       3.1       0.9       66.8      350.5
12.9
106      89.6       2.9       1.3       72.9      372.7
14.0
107     143.4       4.0       0.8       62.6      324.9
8.2
108     107.3       3.4       1.2       68.7      361.6
12.1
109     181.4       4.9       1.2       60.2      313.6
6.2
110     104.1       3.3       0.8       68.9      325.2
9.3
111     116.2       3.3       0.7       62.5      324.7
8.9
112     108.0       3.3       0.8       72.7      369.8
12.8
113     140.1       3.8       1.1       63.1      305.6
7.4
114     137.2       4.4       1.2       68.9      342.8
9.2
115      92.3       3.6       0.7       61.0      339.4
10.0
116     191.0       4.7       1.5       61.6      302.9
5.0
117     110.2       3.1       1.4       66.7      313.5
9.0
118     136.3       3.5       1.2       66.7      301.9
7.6
119     118.4       3.3       1.6       68.8      304.0
8.4
120     105.0       3.5       0.9       68.7      323.8
```

```
10.4
121        62.8          2.1           0.5           66.8          309.8
10.1
122        94.8          2.9           0.9           59.3          251.5
7.2
123       127.8          3.6           0.9           63.5          281.7
6.9
124       132.7          3.7           1.1           67.3          324.8
8.9
125       123.1          3.7           1.3           67.0          344.1
9.2
126       108.3          3.4           0.9           62.3          281.3
7.2
127       125.1          3.2           0.9           73.8          335.4
9.5
128        89.0          2.8           0.3           58.9          285.8
7.9
129       123.2          3.4           1.0           57.8          228.1
4.1
130       148.8          3.4           0.8           66.4          265.8
5.2

     Offense: Att.1  Offense: Avg  Offense: TD.1  Offense: Plays  Offense: A
vg.1  Offense: Pass
0              10.3          25.4            8.1            69.6
0.2            0.8
1               8.7          23.8            5.6            48.0
0.5            0.8
2               9.9          25.6            6.3            58.0
0.4            0.5
3               9.3          23.8            7.9            68.7
0.6            1.2
4               9.9          25.1            4.4            44.9
0.5            1.2
5              12.6          23.3            4.1            32.0
0.4            0.7
6               8.7          27.2            6.8            60.5
0.7            0.8
7              12.2          25.7            6.5            52.5
0.9            1.6
8              12.7          26.5            6.8            57.2
0.7            0.9
9               9.4          21.1            4.8            49.6
0.5            1.1
10             12.5          26.1            4.5            40.2
0.7            1.3
11              9.4          25.5            4.3            42.3
0.8            1.5
12             10.6          22.6            5.1            39.9
0.9            1.8
```

13	9.7	25.9	7.0	61.5	0.6	1.4
14	8.1	24.6	6.2	57.5	0.9	1.6
15	11.2	24.2	6.7	59.1	0.5	1.1
16	7.2	23.5	7.9	70.4	0.8	1.5
17	8.2	25.4	4.8	40.2	1.0	1.5
18	10.4	21.6	4.0	34.3	0.4	1.1
19	7.9	21.4	5.3	48.3	0.5	1.2
20	7.8	20.5	5.3	46.5	0.6	1.5
21	8.9	22.6	4.1	36.8	0.6	1.2
22	10.4	23.0	5.6	51.6	0.5	0.9
23	9.9	24.7	6.1	54.3	0.5	1.3
24	9.5	22.9	6.1	47.3	0.5	0.9
25	8.9	24.7	6.1	55.3	0.5	1.0
26	10.5	26.5	5.7	44.3	1.4	1.9
27	9.4	21.9	5.3	42.4	1.1	1.6
28	13.5	25.2	6.8	63.5	0.9	1.6
29	10.2	22.9	5.4	48.2	0.7	1.6
30	11.8	24.4	4.9	46.4	0.6	1.5
31	9.8	21.8	5.2	50.5	0.5	0.8
32	11.9	23.8	5.7	50.2	0.6	1.2
33	7.7	25.2	5.1	45.8	1.2	1.6
34	12.8	23.9	6.4	55.5	0.5	1.4
35	8.0	22.6	4.3	37.5	0.4	0.5
36	9.5	20.1	5.3	42.9	0.4	0.9
37	10.3	22.8	4.8	41.1	0.8	1.3

38	11.3	22.4	6.0	54.7
1.0	1.4			
39	7.3	19.0	6.9	63.8
1.1	2.1			
40	10.7	21.6	4.6	42.3
0.8	1.4			
41	7.9	20.9	4.9	44.1
0.4	0.9			
42	5.4	22.8	6.3	60.8
0.7	1.5			
43	9.4	21.9	5.8	54.0
0.5	0.8			
44	11.2	21.5	7.4	63.7
0.8	1.5			
45	9.4	22.0	6.7	55.0
1.1	1.9			
46	8.3	22.5	7.8	71.9
0.9	1.2			
47	8.2	24.2	5.4	50.0
1.1	1.8			
48	7.9	22.1	4.5	40.1
0.6	1.0			
49	7.7	22.2	3.5	33.5
1.4	1.8			
50	7.6	21.4	6.2	57.1
0.8	1.5			
51	9.3	23.3	6.4	53.0
1.0	1.6			
52	11.0	21.5	7.0	60.5
0.5	1.4			
53	11.5	22.2	7.1	64.9
0.6	1.7			
54	9.2	21.3	7.9	73.5
0.9	1.5			
55	9.0	22.2	6.2	58.0
0.4	1.1			
56	10.0	19.5	5.6	47.4
0.6	1.2			
57	8.7	20.0	6.8	50.1
0.7	1.1			
58	6.5	18.5	7.5	64.4
0.6	1.2			
59	8.7	21.8	6.2	56.9
0.5	1.5			
60	7.0	19.5	6.8	64.3
1.6	2.1			
61	5.7	20.2	6.5	58.3
0.8	1.4			
62	14.6	17.7	4.8	38.9
0.3	1.1			

```
63           9.2        22.1          5.5          51.2
0.6          1.4
64          11.6        22.8          6.8          54.8
0.7          1.1
65           7.2        20.2          7.8          71.1
0.6          1.2
66          12.5        21.6          3.5          31.8
0.7          1.0
67           9.4        19.5          4.6          36.9
0.8          1.7
68          17.2        20.4          3.9          29.7
0.2          1.0
69           7.8        20.3          8.5          63.0
0.7          1.0
70           8.6        20.5          6.2          53.5
1.2          2.1
71           5.8        20.3          5.8          48.9
0.6          0.8
72           8.8        18.8          6.3          62.5
1.2          1.3
73           8.9        20.3          6.9          66.0
0.9          1.8
74           7.9        23.1          5.4          53.6
1.2          1.6
75           7.3        17.7          5.8          57.1
0.6          1.3
76           7.9        18.0          6.2          57.8
0.8          1.4
77           8.8        19.6          6.5          55.7
0.8          1.5
78           7.3        21.3          7.3          72.3
0.8          1.5
79           6.5        19.8          5.4          45.3
0.7          1.4
80           7.2        19.6          5.4          49.9
1.1          1.5
81           7.3        16.7          6.5          56.6
1.0          1.5
82           7.7        17.9          6.6          58.4
1.1          1.9
83           8.4        20.1          6.0          52.5
1.5          2.5
84           5.5        17.2          5.8          55.3
0.4          0.8
85          10.0        18.1          7.5          57.3
1.0          1.9
86           7.9        20.1          7.0          63.2
0.8          2.3
87           7.8        18.6          7.8          63.7
0.7          1.6
```

88	7.8	17.5	5.3	46.1	0.5	1.3
89	11.7	20.9	7.2	66.1	0.8	1.5
90	6.8	21.3	6.8	59.5	1.2	1.8
91	6.7	20.2	5.8	54.9	1.0	1.3
92	9.3	21.0	6.5	55.8	1.0	1.8
93	7.4	20.4	7.3	57.5	0.6	1.2
94	8.9	21.1	6.5	61.4	0.4	1.3
95	5.7	19.3	5.3	44.6	0.8	1.0
96	8.3	21.3	7.1	60.0	1.0	2.1
97	6.2	17.5	7.8	70.5	0.8	1.8
98	7.9	21.3	6.0	55.9	1.2	1.8
99	6.8	19.2	5.0	45.3	1.1	1.5
100	7.8	19.5	6.3	49.1	0.5	1.4
101	7.0	18.1	5.4	42.5	1.1	1.6
102	7.3	17.8	4.9	44.8	0.6	1.3
103	8.8	19.8	8.5	77.5	1.6	2.1
104	13.3	17.6	3.8	31.3	0.5	1.3
105	3.9	18.4	4.6	37.0	1.0	1.9
106	6.5	22.8	6.7	60.1	1.0	2.2
107	6.8	15.8	7.4	59.4	1.1	1.9
108	6.5	20.8	4.6	41.3	0.7	1.8
109	8.6	16.0	4.3	37.5	0.9	1.2
110	6.6	17.8	4.9	44.7	0.8	1.4
111	6.9	17.6	5.7	39.5	1.0	1.5
112	6.1	20.4	5.6	46.1	1.3	1.8

```
113              7.7          17.1            5.5            48.2
0.4            0.9
114              8.1          18.8            5.6            50.2
0.9            1.5
115              4.6          16.0            6.8            62.1
0.6            1.6
116              9.5          16.2            7.1            60.0
0.8            1.7
117              6.5          17.5            7.4            58.5
0.8            1.4
118              7.0          16.8            6.7            62.9
1.1            1.8
119              7.0          17.2            6.0            55.1
0.7            1.4
120              6.0          17.8            6.2            53.3
1.3            1.8
121              5.4          16.8            6.0            48.5
1.2            2.2
122              5.5          13.8            4.1            32.7
0.5            1.3
123              7.2          14.8            7.8            64.2
1.2            1.7
124              7.2          17.7            5.8            51.9
0.8            1.1
125              7.4          19.1            7.0            62.2
1.2            2.2
126              7.2          15.9            5.3            48.3
0.8            1.8
127              7.8          19.0            4.6            40.8
1.4            2.6
128              5.5          14.7            6.9            60.4
0.9            1.7
129              7.7          13.3            6.1            51.2
0.8            1.2
130              7.8          14.6            7.6            66.7
1.2            1.7
```

In [ ]:
```
df_team_stats_full = pd.merge(df_team_stats_offense, df_team_stats_defense,
print(df_team_stats_full.head())
```

```
    Team Name  Offense: Passing  Offense: Rushing  Offense: Total Offense  O
ffense: First Downs  \
0   Tennessee                13              46.1                    22.3
32.5
1  Ohio State                13              44.2                    21.1
31.5
2         USC                14              41.4                    24.6
36.8
3     Alabama                13              41.1                    21.5
33.7
```

```
4     Georgia                 15              41.1                    22.4
32.9

   Offense: Penalties  Offense: Turnovers  Offense: Rk  Offense: School  Off
ense: G  Offense: Pts  \
0                 68.7               326.1          2.9             40.2
199.5           5.0
1                 66.8               298.3          3.2             35.8
192.4           5.4
2                 67.0               335.4          3.1             33.8
171.1           5.1
3                 63.9               281.5          2.8             35.2
195.5           5.6
4                 68.2               295.9          2.1             37.1
205.3           5.5

   Offense: Cmp  Offense: Att  Offense: Pct  Offense: TD  Offense: Att.1  Of
fense: Avg  \
0           3.1          72.7         525.5         11.9            10.3
25.4
1           2.3          67.4         490.7         13.2             8.7
23.8
2           2.1          70.6         506.6         13.9             9.9
25.6
3           2.0          68.9         477.1         12.6             9.3
23.8
4           2.9          70.0         501.2         13.8             9.9
25.1

   Offense: TD.1  Offense: Plays  Offense: Avg.1  Offense: Pass  Defense: Pa
ssing  \
0            8.1            69.6             0.2            0.8
13
1            5.6            48.0             0.5            0.8
13
2            6.3            58.0             0.4            0.5
14
3            7.9            68.7             0.6            1.2
13
4            4.4            44.9             0.5            1.2
15

   Defense: Rushing  Defense: Total Offense  Defense: First Downs  Defense:
Penalties  \
0              22.8                    25.6                  40.9
62.6
1              21.0                    16.3                  27.7
58.9
2              29.2                    20.9                  32.9
63.7
```

```
3           18.2                    18.3                  33.4
54.8
4           14.3                    19.6                  34.1
57.5

   Defense: Turnovers  Defense: Rk  Defense: School  Defense: G  Defense: Pt
s  Defense: Cmp  \
0               289.5          1.6             35.3       115.8           3.
3           1.2
1               200.5          1.5             34.4       121.1           3.
5           0.9
2               264.1          1.8             32.1       159.8           5.
0           2.1
3               187.8          0.9             35.9       130.4           3.
6           1.2
4               219.7          1.0             26.7        77.1           2.
9           0.5

   Defense: Att  Defense: Pct  Defense: TD  Defense: Att.1  Defense: Avg  De
fense: TD.1  \
0          76.2         405.3         13.7             7.4          23.6
8.5
1          62.1         321.5          7.8             5.6          15.0
5.4
2          64.9         423.9         11.6             8.6          22.1
5.9
3          69.3         318.2          8.8             7.2          18.5
6.7
4          60.8         296.8          9.4             4.5          15.5
5.5

   Defense: Plays  Defense: Avg.1  Defense: Pass
0            67.4             0.8            1.6
1            45.4             0.8            1.4
2            49.9             1.4            2.1
3            51.0             0.5            1.1
4            40.8             0.8            1.3
```

In [ ]:

```python
df_team_stats_full['Returning Starters'] = None

for team, score in returning_starters:
    if team in df_team_stats_full['Team Name'].values:
        df_team_stats_full.loc[df_team_stats_full['Team Name'] == team, 'Ret
    else:
        print(f"Team not found in DataFrame: {team}")

df_team_stats_full.rename(columns={'Team Name': 'Team'}, inplace=True)


print(df_team_stats_full.head())
```

```
Team not found in DataFrame: Sam Houston
Team not found in DataFrame: J'ville State
         Team  Offense: Passing  Offense: Rushing  Offense: Total Offense  O
ffense: First Downs  \
0   Tennessee                13              46.1                    22.3
32.5
1  Ohio State                13              44.2                    21.1
31.5
2         USC                14              41.4                    24.6
36.8
3     Alabama                13              41.1                    21.5
33.7
4     Georgia                15              41.1                    22.4
32.9

   Offense: Penalties  Offense: Turnovers  Offense: Rk  Offense: School  Off
ense: G  Offense: Pts  \
0                68.7               326.1          2.9             40.2
199.5           5.0
1                66.8               298.3          3.2             35.8
192.4           5.4
2                67.0               335.4          3.1             33.8
171.1           5.1
3                63.9               281.5          2.8             35.2
195.5           5.6
4                68.2               295.9          2.1             37.1
205.3           5.5

   Offense: Cmp  Offense: Att  Offense: Pct  Offense: TD  Offense: Att.1  Of
fense: Avg  \
0           3.1          72.7         525.5         11.9            10.3
25.4
1           2.3          67.4         490.7         13.2             8.7
23.8
2           2.1          70.6         506.6         13.9             9.9
25.6
3           2.0          68.9         477.1         12.6             9.3
23.8
4           2.9          70.0         501.2         13.8             9.9
25.1

   Offense: TD.1  Offense: Plays  Offense: Avg.1  Offense: Pass  Defense: Pa
ssing  \
0            8.1            69.6             0.2            0.8
13
1            5.6            48.0             0.5            0.8
13
2            6.3            58.0             0.4            0.5
14
3            7.9            68.7             0.6            1.2
```

```
13
4          4.4          44.9            0.5           1.2
15

   Defense: Rushing  Defense: Total Offense  Defense: First Downs  Defense:
Penalties  \
0             22.8                    25.6                  40.9
62.6
1             21.0                    16.3                  27.7
58.9
2             29.2                    20.9                  32.9
63.7
3             18.2                    18.3                  33.4
54.8
4             14.3                    19.6                  34.1
57.5

   Defense: Turnovers  Defense: Rk  Defense: School  Defense: G  Defense: Pt
s  Defense: Cmp  \
0               289.5          1.6             35.3       115.8           3.
3           1.2
1               200.5          1.5             34.4       121.1           3.
5           0.9
2               264.1          1.8             32.1       159.8           5.
0           2.1
3               187.8          0.9             35.9       130.4           3.
6           1.2
4               219.7          1.0             26.7        77.1           2.
9           0.5

   Defense: Att  Defense: Pct  Defense: TD  Defense: Att.1  Defense: Avg  De
fense: TD.1  \
0          76.2         405.3         13.7             7.4          23.6
8.5
1          62.1         321.5          7.8             5.6          15.0
5.4
2          64.9         423.9         11.6             8.6          22.1
5.9
3          69.3         318.2          8.8             7.2          18.5
6.7
4          60.8         296.8          9.4             4.5          15.5
5.5

   Defense: Plays  Defense: Avg.1  Defense: Pass Returning Starters
0            67.4             0.8            1.6           0.481242
1            45.4             0.8            1.4           0.639125
2            49.9             1.4            2.1           0.894745
3            51.0             0.5            1.1           0.060221
4            40.8             0.8            1.3           0.398541
```

Merge the sentiment and statistics tables

In [ ]:
```
from sklearn.preprocessing import StandardScaler

df_team_sentiment_avg = pd.read_csv("all_teams_data_sentiment_full.csv")
df_sentiment_and_stats = df_team_stats_full

df_sentiment_and_stats = pd.merge(df_team_sentiment_avg, df_team_stats_full,
df_sentiment_and_stats['Returning Starters'] = pd.to_numeric(df_sentiment_ar
df_sentiment_and_stats = df_sentiment_and_stats.drop(columns=['Unnamed: 0'],

columns_to_normalize = [col for col in df_sentiment_and_stats.columns
                        if df_sentiment_and_stats[col].dtype in ['float64',
                        and col not in ['Sentiment_Avg', 'Win Diff']]

df_normalized = df_sentiment_and_stats.dropna(axis=0, how='any').copy()
scaler = StandardScaler()
df_normalized[columns_to_normalize] = scaler.fit_transform(df_normalized[col
if 'Returning Starters' in df_normalized.columns:
    df_normalized['Returning Starters'] *= 8
if 'Offense: Total Offense' in df_normalized.columns:
    df_normalized['Offense: Total Offense'] *= 4
if 'Defense: Total Offense' in df_normalized.columns:
    df_normalized['Defense: Total Offense'] *= 4
# print(df_normalized.head())

print(df_normalized.columns)
df_normalized = df_normalized.drop(columns=['Unnamed: 0'], errors='ignore')
X_train, X_test = train_test_split(df_normalized, test_size=0.2, random_stat
print(X_train.head())
# Check the split
print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")
# missing_teams = df_sentiment_and_stats[df_sentiment_and_stats['Team'].isna

# if not missing_teams.empty:
#     print("Missing Teams (from df_team_stats_full):")
#     for team in missing_teams['Team']:
#         print(team)
# else:
#     print("No teams are missing in the merge.")
```

```
Index(['Team', 'Sentiment_Avg', 'Win Diff', 'Bet Type', 'Offense: Passing',
'Offense: Rushing',
       'Offense: Total Offense', 'Offense: First Downs', 'Offense: Penalties
',
       'Offense: Turnovers', 'Offense: Rk', 'Offense: School', 'Offense: G',
'Offense: Pts',
       'Offense: Cmp', 'Offense: Att', 'Offense: Pct', 'Offense: TD', 'Offen
se: Att.1',
```

```
       'Offense: Avg', 'Offense: TD.1', 'Offense: Plays', 'Offense: Avg.1',
'Offense: Pass',
       'Defense: Passing', 'Defense: Rushing', 'Defense: Total Offense', 'De
fense: First Downs',
       'Defense: Penalties', 'Defense: Turnovers', 'Defense: Rk', 'Defense:
School', 'Defense: G',
       'Defense: Pts', 'Defense: Cmp', 'Defense: Att', 'Defense: Pct', 'Defe
nse: TD',
       'Defense: Att.1', 'Defense: Avg', 'Defense: TD.1', 'Defense: Plays',
'Defense: Avg.1',
       'Defense: Pass', 'Returning Starters'],
      dtype='object')
              Team  Sentiment_Avg  Win Diff Bet Type  Offense: Passing  Offe
nse: Rushing  \
98       Texas A&M      -0.308615      -1.0    Under         -0.927366
-0.770339
56      Miami (FL)      -0.207101      -0.5    Under         -0.927366
-0.657694
27   East Carolina      -0.216950      -3.5    Under          0.332020
0.595483
59  Michigan State       0.003733      -1.0    Under         -0.927366
-0.545049
65        Nebraska      -0.361603      -1.5    Under         -0.927366
-0.798500

    Offense: Total Offense  Offense: First Downs  Offense: Penalties  Offens
e: Turnovers  \
98               -0.933504              0.114579           -0.754710
-0.249422
56                1.562900              0.383696            0.237538
0.108937
27                4.475371              0.802324            1.156286
1.050544
59                1.729327              0.368745            0.402913
0.127221
65               -1.932065             -0.528313           -0.019711
-0.223825

    Offense: Rk  Offense: School  Offense: G  Offense: Pts  Offense: Cmp  Of
fense: Att  \
98    -0.280182        -1.047827   -0.371532      0.395590     -1.116122
-1.146628
56    -0.280182        -0.384038   -0.665239     -0.720177     -1.116122
0.067380
27     0.820533        -0.707422    0.256915      1.263409      0.328271
0.259066
59     0.348798        -1.115907   -0.987021     -0.596203     -0.634658
-0.869749
65    -0.437427        -0.281917   -0.764581     -0.968125     -0.474170
-1.104031
```

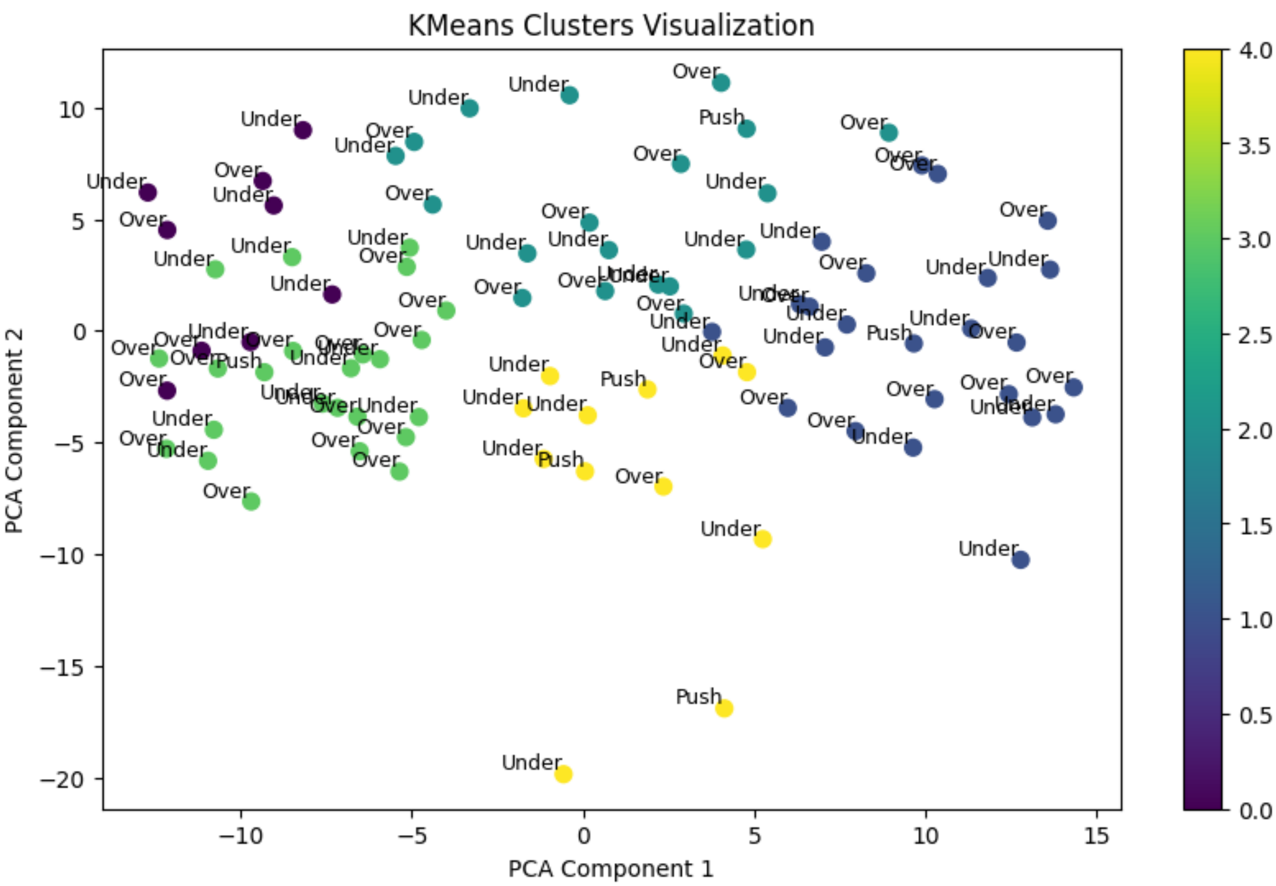
```
    Offense: Pct  Offense: TD  Offense: Att.1  Offense: Avg  Offense: TD.1
Offense: Plays  \
98    -0.499960    -0.428239       -0.361275     -0.388206       0.273250
-0.340190
56    -0.402712     0.233585       -0.130227      0.203517       0.988904
0.718159
27     1.124088     0.856479       -0.268856      0.630872      -1.515886
-1.466507
59    -0.628004     0.350378       -0.869581     -0.158092      -0.174034
0.222968
65    -0.770635    -0.389308       -0.730952     -0.848435      -0.531861
-0.981026

    Offense: Avg.1  Offense: Pass  Defense: Passing  Defense: Rushing  Defen
se: Total Offense  \
98       -0.971045      -0.137030         -0.927366         -0.903662
-6.189567
56        0.662211       1.564795         -0.927366          0.052791
-3.442240
27       -1.297696      -2.325090          0.332020          0.121109
4.961350
59        0.662211      -0.380148         -0.927366          0.155268
1.244378
65        0.988862       0.349206         -0.927366          0.189427
1.890808

    Defense: First Downs  Defense: Penalties  Defense: Turnovers  Defense: R
k  Defense: School  \
98             -1.337795           -1.029621           -2.149003   -0.66501
1         1.807591
56             -1.155293            0.158223            0.188151    0.27282
5        -0.305658
27              0.548059            1.501002            1.940263    1.21066
1        -1.064260
59             -0.516536            1.578471            0.314810    1.44512
0         1.374104
65              0.700144           -0.177472           -0.065166   -0.19609
3         1.347011

    Defense: G  Defense: Pts  Defense: Cmp  Defense: Att  Defense: Pct  Defe
nse: TD  \
98    1.580142      0.908383     -0.842199      0.510156     -0.296017    -1
.412732
56   -0.280948     -0.200323     -0.108402     -1.097657     -0.071759    -0
.759239
27   -1.139478     -1.031852     -0.658750     -0.472397      0.463754     1
.854730
59    0.727260      0.215442     -0.658750      0.755794      0.699612     1
.070539
```

```
65    1.023792      0.492618      0.441946      1.626693      0.659013     0
.482396

    Defense: Att.1  Defense: Avg  Defense: TD.1  Defense: Plays  Defense: Av
g.1  Defense: Pass  \
98         1.432101      0.224999       0.965703        0.910283       -1.867
006       -0.473302
56        -0.342564     -0.595028       0.039302        0.247535        1.297
953        0.918764
27        -0.697497      0.374095      -0.269499        0.034108       -0.108
696        0.083524
59         0.663079      0.933204       1.377437        1.179875       -2.218
668       -1.308542
65         1.609568      1.045026      -0.372432        0.045341       -0.108
696       -0.473302

    Returning Starters
98           12.126733
56            6.379643
27          -13.119412
59            4.327111
65            5.558630
Training set size: 88
Test set size: 22
```

In [ ]:
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.metrics.pairwise import rbf_kernel
from sklearn.cluster import KMeans


# if 'Defense: First Downs' in X_train.columns:
#     X_train['Defense: First Downs'] *= 2
df_numeric_clean = X_train

df_numeric_only = df_numeric_clean.drop(columns=['Win Diff', 'Team', 'Bet Ty

kernel_matrix = rbf_kernel(X, gamma=1.0)

kmeans = KMeans(n_clusters=5, random_state=42)
input_df = df_numeric_only.drop(columns=['Sentiment_Avg', 'Win Diff', 'Clust
input_df = input_df.loc[:, ~input_df.columns.str.contains('^Unnamed')]
print(input_df.columns)
kmeans.fit(input_df)
```

```
df_numeric_clean['Cluster'] = kmeans.labels_

X_train.loc[df_numeric_clean.index, 'Cluster'] = df_numeric_clean['Cluster']

print("Cluster Assignments:")

# Reduce dimensions for visualization
pca = PCA(n_components=2)


reduced_data = pca.fit_transform(input_df)

plt.figure(figsize=(10, 6))
plt.scatter(reduced_data[:, 0], reduced_data[:, 1], c=kmeans.labels_, cmap='
plt.colorbar()
plt.title("KMeans Clusters Visualization")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
try:
  for i, team_name in enumerate(X_train['Bet Type']):
      plt.text(reduced_data[i, 0], reduced_data[i, 1], team_name, fontsize=9
except:
  pass
plt.show()

# calc the Silhouette Score
sil_score = silhouette_score(input_df, kmeans.labels_)
print(f"Silhouette Score : {sil_score:.3f}")

# Calculate WCSS (within - Cluster Sum of Squares)
wcss = kmeans.inertia_
print(f"WCSS (Inertia): {wcss:.3f}")
```

```
Index(['Offense: Passing', 'Offense: Rushing', 'Offense: Total Offense', 'Of
fense: First Downs',
       'Offense: Penalties', 'Offense: Turnovers', 'Offense: Rk', 'Offense:
School', 'Offense: G',
       'Offense: Pts', 'Offense: Cmp', 'Offense: Att', 'Offense: Pct', 'Offe
nse: TD',
       'Offense: Att.1', 'Offense: Avg', 'Offense: TD.1', 'Offense: Plays',
'Offense: Avg.1',
       'Offense: Pass', 'Defense: Passing', 'Defense: Rushing', 'Defense: To
tal Offense',
       'Defense: First Downs', 'Defense: Penalties', 'Defense: Turnovers', '
Defense: Rk',
       'Defense: School', 'Defense: G', 'Defense: Pts', 'Defense: Cmp', 'Def
ense: Att',
       'Defense: Pct', 'Defense: TD', 'Defense: Att.1', 'Defense: Avg', 'Def
ense: TD.1',
       'Defense: Plays', 'Defense: Avg.1', 'Defense: Pass', 'Returning Start
ers'],
      dtype='object')
Cluster Assignments:
```




```
Silhouette Score : 0.153
WCSS (Inertia): 5328.330
```

In [ ]:
```
# Step 12: Assign cluster names to the DataFrame
# You can name the clusters based on their cluster number or analysis
```

```
cluster_names = {0: 'Cluster A', 1: 'Cluster B', 2: 'Cluster C', 3: 'Cluster

X_train['Cluster Name'] = X_train['Cluster'].map(cluster_names)
# try:
#   X_train.drop(columns=['Unnamed: 0'], inplace=True)
# except:
#   pass

# Print the updated DataFrame with 'Team Name' and 'Cluster Name'

# print(X_train[['Team', 'Cluster Name']])
```

In [ ]:
```
print(input_df.columns)
```

```
Index(['Offense: Passing', 'Offense: Rushing', 'Offense: Total Offense', 'Of
fense: First Downs',
       'Offense: Penalties', 'Offense: Turnovers', 'Offense: Rk', 'Offense:
School', 'Offense: G',
       'Offense: Pts', 'Offense: Cmp', 'Offense: Att', 'Offense: Pct', 'Offe
nse: TD',
       'Offense: Att.1', 'Offense: Avg', 'Offense: TD.1', 'Offense: Plays',
'Offense: Avg.1',
       'Offense: Pass', 'Defense: Passing', 'Defense: Rushing', 'Defense: To
tal Offense',
       'Defense: First Downs', 'Defense: Penalties', 'Defense: Turnovers', '
Defense: Rk',
       'Defense: School', 'Defense: G', 'Defense: Pts', 'Defense: Cmp', 'Def
ense: Att',
       'Defense: Pct', 'Defense: TD', 'Defense: Att.1', 'Defense: Avg', 'Def
ense: TD.1',
       'Defense: Plays', 'Defense: Avg.1', 'Defense: Pass', 'Returning Start
ers'],
      dtype='object')
```

In [ ]:
```
# Step 1: Create a list to store the cluster names and win differences for e
cluster_info = []

# Step 2: Iterate over each record in the DataFrame
for index, row in X_train.iterrows():
    cluster_label = row['Cluster']
    cluster_name = cluster_names.get(cluster_label, 'Unknown')  # Get cluste
    win_diff = row['Win Diff']

    # Append the cluster name and win difference to the list
    cluster_info.append((cluster_name, win_diff))

# Step 3: Add the cluster names to the DataFrame for convenience
X_train['Cluster Name'] = [info[0] for info in cluster_info]

average_win_diff_per_cluster = X_train.groupby('Cluster')['Win Diff'].mean()
```

```
average_win_diff_dict = average_win_diff_per_cluster.to_dict()

print(average_win_diff_dict)
print(type(average_win_diff_per_cluster))
# Step 5: Print the average win difference per cluster
print("Average Win Difference per Cluster:")
print(average_win_diff_per_cluster)

# Step 6: Merge the average win difference back to X_train
# We will create a new column 'Avg Win Diff per Cluster' in X_train
# X_train = X_train.merge(average_win_diff_per_cluster, on='Cluster Name', h
X_train['Cluster Prob'] = X_train['Cluster'].map(average_win_diff_dict)


# print(X_train.head())
```

```
{0: -0.6666666666666666, 1: 0.10416666666666667, 2: 0.42105263157894735, 3:
0.20833333333333334, 4: -0.4166666666666667}
<class 'pandas.core.series.Series'>
Average Win Difference per Cluster:
Cluster
0   -0.666667
1    0.104167
2    0.421053
3    0.208333
4   -0.416667
Name: Win Diff, dtype: float64
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler


# Step 1: Apply the same transformations to X_test (same as X_train)
# Assuming you already fit the scaler to X_train
df_numeric_only_test = X_test.select_dtypes(include=[np.number])  # Keep onl
# df_numeric_only_test = X_train.dropna(axis=0, how='any')  # Drops rows wit
df_numeric_only_test = df_numeric_only_test.drop(columns=['Cluster'], errors
df_numeric_only_test = df_numeric_only_test.drop(columns=['Cluster Prob'], e
df_numeric_only_test = df_numeric_only_test.drop(columns=['Win Diff', 'Bet T
X_test_scaled = scaler.transform(df_numeric_only_test)  # Apply the same sca
# print(X_test_scaled)
# Step 2: Use the KMeans model to predict the clusters for X_test
test_cluster_labels = kmeans.predict(X_test_scaled)

# Step 3: Assign the predicted cluster labels to X_test DataFrame
X_test['Cluster'] = test_cluster_labels  # Add the cluster labels to X_test
X_test['Cluster Prob'] = X_test['Cluster'].map(average_win_diff_dict)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X
does not have valid feature names, but KMeans was fitted with feature names
  warnings.warn(
```

Making Predictions simply using K-Means

In [ ]:
```
X_test_split_with_all_data_k_means = pd.merge(X_test, df_sentiment_and_stats
# print(X_test_split_with_all_data_k_means.columns)
matching_count = 0
num_pushes = 0
actuals = []
predictions = []

# # Step 2: Iterate over the DataFrame to check if 'Prediction' and 'Win Dif
for index, row in X_test_split_with_all_data_k_means.iterrows():
    # Get the 'Prediction' and 'Win Diff' values

    prediction_scalar = row['Cluster Prob']
    prediction = None
    if prediction_scalar > 0:
        prediction = "Over"
    elif prediction_scalar < 0:
        prediction = "Under"
    else:
        prediction = "Push"

    # Get the actual 'Win Diff_x' value and determine the label
    win_diff = row['Win Diff_x']
    if win_diff > 0:
        actual = "Over"
    elif win_diff < 0:
        actual = "Under"
    else:
        actual = "Push"

    # Append to the lists
    actuals.append(actual)
    predictions.append(prediction)

    # Check if 'Prediction' and 'Win Diff' have the same sign
    if (prediction == "Over" and win_diff > 0) or (prediction == "Under" and
        matching_count += 1  # Increment the counter if the signs match
    if win_diff == 0:
      num_pushes += 1

# # Step 3: Calculate the percentage of matching predictions
total_records = len(X_test_split_with_all_data_k_means) - num_pushes
percentage_matching = (matching_count / total_records) * 100
print(f'Percentage Correct: {round(percentage_matching, 3)}%')
```

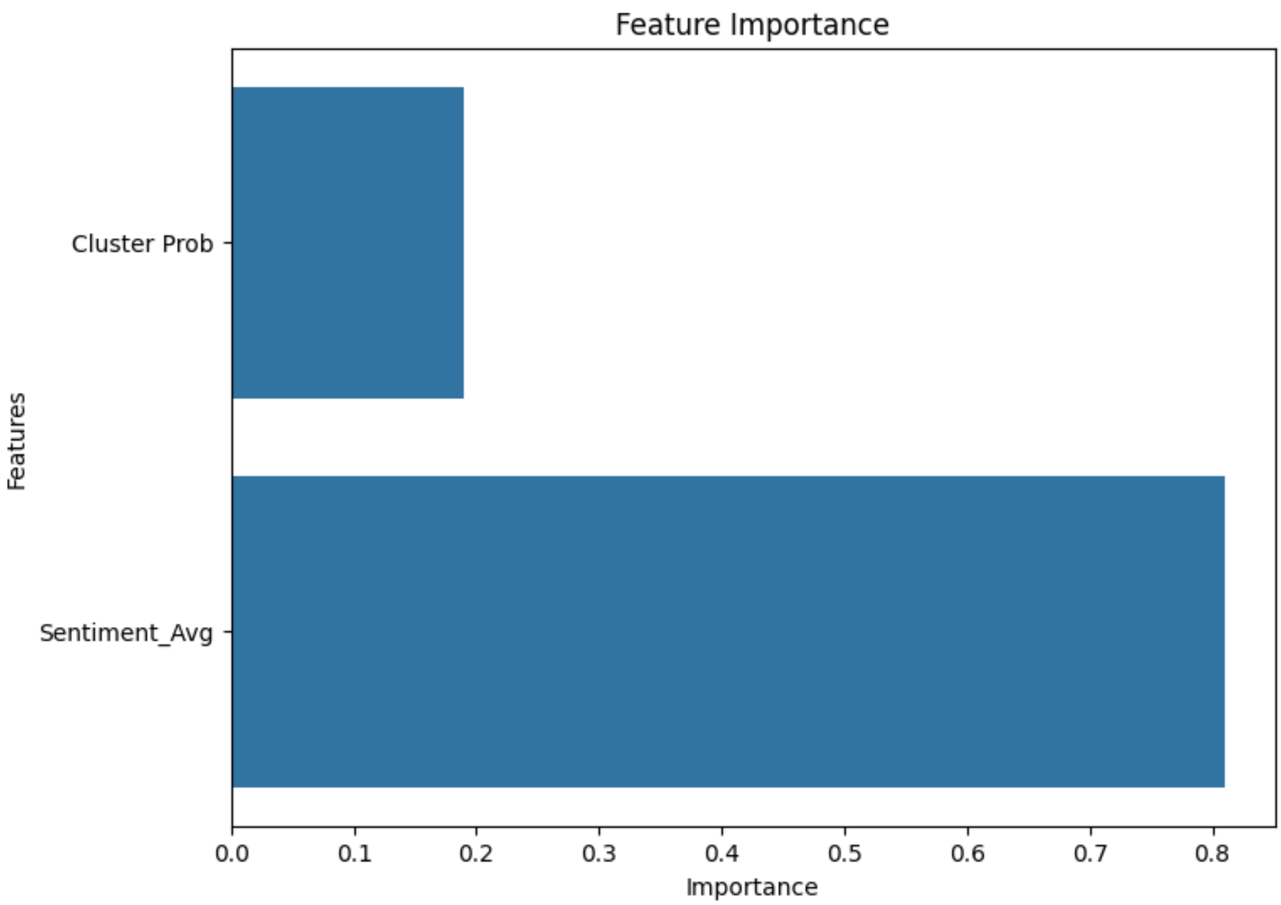
```
Percentage Correct: 57.143%
```

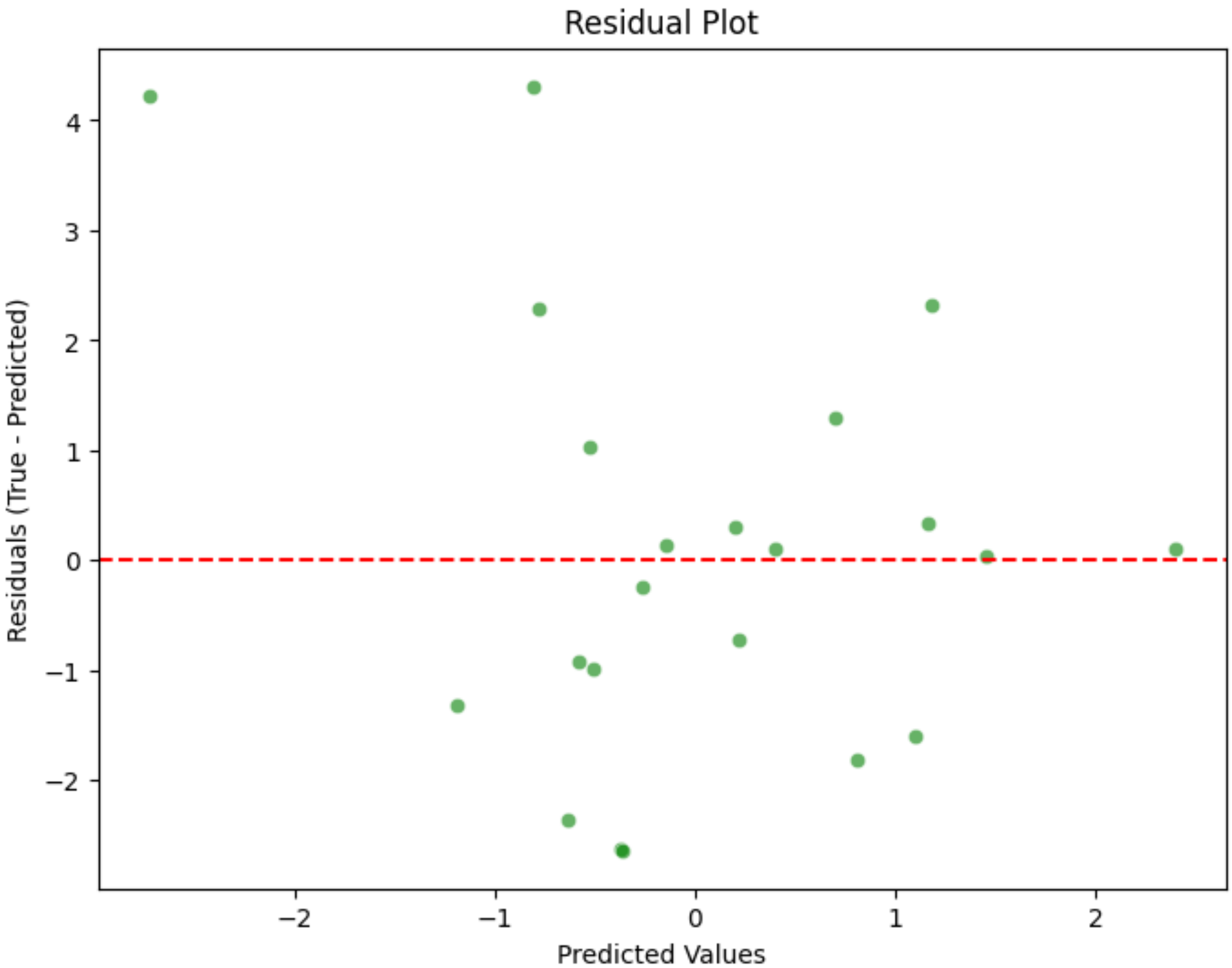
Model 3: Making Predictions Using a random forest

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split

X_train_split = X_train[['Cluster Prob', 'Sentiment_Avg']]  # Features
y_train_split = X_train['Win Diff']  # Target variable

X_test_split = X_test[['Cluster Prob', 'Sentiment_Avg']]  # Features
y_test_split = X_test['Win Diff']
rf_model = RandomForestRegressor(n_estimators=100, random_state=7)

rf_model.fit(X_train_split, y_train_split)

y_pred = rf_model.predict(X_test_split)

prediction_results = []
for pred in y_pred:
    if pred > 0:
        prediction_results.append("Over")
    elif pred < 0:
        prediction_results.append("Under")
    else:
        prediction_results.append("Push")

X_test_split['Prediction'] = prediction_results

mae = mean_absolute_error(y_test_split, y_pred)
r2 = r2_score(y_test_split, y_pred)

print(f"Mean Absolute Error: {mae:.2f}")
print(f"R-squared Score: {r2:.2f}")

# print(X_test_split.head())

# Feature Importance Plot
feature_importances = rf_model.feature_importances_

# Plot feature importances
plt.figure(figsize=(8, 6))
sns.barplot(x=feature_importances, y=X_train_split.columns)
plt.title('Feature Importance')
plt.xlabel('Importance')
plt.ylabel('Features')
plt.show()
```

```python
residuals = y_test_split - y_pred

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred, y=residuals, color='green', alpha=0.6)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residual Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals (True - Predicted)')
plt.show()

# Merging predictions with additional data
X_test_split_with_all_data = pd.merge(X_test_split, df_sentiment_and_stats,
```

```
<ipython-input-25-38fc1aa70c4f>:27: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_test_split['Prediction'] = prediction_results
Mean Absolute Error: 1.44
R-squared Score: 0.05
```



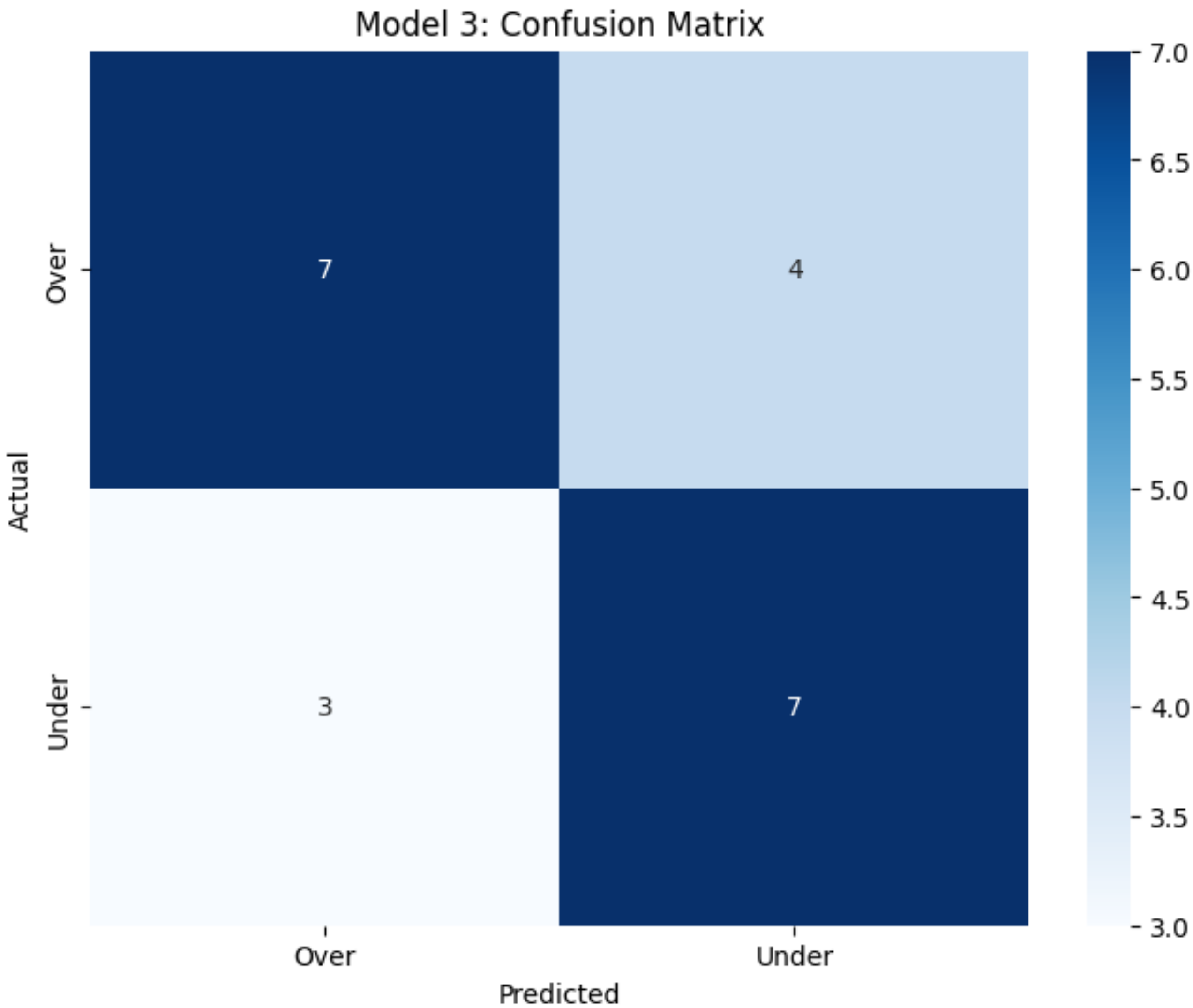


See how well it fared making actual predictions

In [ ]:
```
matching_count = 0
num_pushes = 0
actuals = []
predictions = []

# # Step 2: Iterate over the DataFrame to check if 'Prediction' and 'Win Dif
for index, row in X_test_split_with_all_data.iterrows():
    # Get the 'Prediction' and 'Win Diff' values
    prediction = row['Prediction']
    win_diff = row['Win Diff']

    # Check if 'Prediction' and 'Win Diff' have the same sign
    if (prediction == "Over" and win_diff > 0) or (prediction == "Under" and
        matching_count += 1  # Increment the counter if the signs match
    if win_diff == 0:
      num_pushes += 1
```

```
    if win_diff > 0:
        actual = "Over"
    elif win_diff < 0:
        actual = "Under"
    else:
        actual = "Push"
    actuals.append(actual)
    predictions.append(prediction)


# # Step 3: Calculate the percentage of matching predictions
total_records = len(X_test_split_with_all_data) - num_pushes
percentage_matching = (matching_count / total_records) * 100
print(percentage_matching)

cm = confusion_matrix(actuals, predictions, labels=["Over", "Under"])

# Step 3: Plot the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["Over", "Unc
plt.title("Model 3: Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Step 4: Print the confusion matrix
print("Confusion Matrix:")
print(cm)
```

```
66.66666666666666
```




```
Confusion Matrix:
[[7 4]
 [3 7]]
```

In [ ]:

In [ ]: