In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('ggplot')
from tqdm.notebook import tqdm
import nltk
from openpyxl import load_workbook
from datetime import datetime

import pymysql
import mysql.connector
```

Followed: https://www.youtube.com/watch?v=QpzMWQvxXWk (https://www.youtube.com/watch?v=QpzMWQvxXWk)

Reading in CSV file. Have to change encoding because it would throw errors if not. Everything stored within db now so no use.

In [32]:
```
# df = pd.read_csv('DataTweets.csv', encoding='cp1252')
# df.head()
```

Out[32]:

	id	name	tweet
0	1	Trey Benson	I think Trey fits us from a schematic standpoi...
1	2	Trey Benson	And then one thing that stands out about Trey ...
2	3	MarShawn Lloyd	No, I would like to get him out there as much ...
3	4	Bucky Irving	The nice thing I like about Bucky is he gets t...
4	5	Bucky Irving	He has taken every detail that weve coached th...

In [7]:
```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
```

Pulling in a very specific model that has been pretrained on sentiment. Hugging face gives us this. The model was already trained on twitter comments, so we don't have to retrain the model at all. Pre-trained weights are already applied

In [8]:
```
MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)
```

```
C:\Users\12505\anaconda3\lib\site-packages\huggingface_hub\file_download.py:1
132: FutureWarning: `resume_download` is deprecated and will be removed in ve
rsion 1.0.0. Downloads always resume when possible. If you want to force a ne
w download, use `force_download=True`.
  warnings.warn(
```

Running the roberta model on our example. First thing is encoding on text using the tokenizer allowing the model to understand it (0s and 1s)

In [9]:
```
# encoded_text = tokenizer(example, return_tensors='pt')
# output = model(**encoded_text)
# scores = output[0][0].detach().numpy()
# scores = softmax(scores)
# scores_dict = {
#     'roberta_neg': scores[0],
#     'roberta_neu': scores[1],
#     'roberta_pos': scores[2]
# }
# print(scores_dict)
```

Creating a function to run the model on for each piece of text we give it.

In [10]:
```
def polarity_scores_roberta(example):
    encoded_text = tokenizer(example, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
    'roberta_neg': scores[0],
    'roberta_neu': scores[1],
    'roberta_pos': scores[2]
}
    return scores_dict
```

Storing our results for the entire dataset into a dictionary which we will then convert to a dataframe

In [11]:
```
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        tweet = row['tweet']
        tweetId = row['id']
        roberta_result = polarity_scores_roberta(tweet)
        res[tweetId] = roberta_result
    except RuntimeError:
        print(f'Broke for id {id}')
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Converting the dictionary to a dataframe and then merging the original dataframe (df) with our new dataframe to get a side by side of the scores with the tweet.

In [12]:
```
results_df = pd.DataFrame(res).T
results_df = results_df.reset_index().rename(columns={'index': 'id'})
results_df = results_df.merge(df, how='left')
```

In [13]:
```
results_df
```

Out[13]:

	id	roberta_neg	roberta_neu	roberta_pos	name	tweet	date
0	1	0.001533	0.030305	0.968162	Bryce Young	Bryce is doing fantastic with the playbook, ma...	2024-05-20
1	2	0.001935	0.096882	0.901182	Andy Dalton	Andy is also showing accuracy in his throws, s...	2024-05-20
2	3	0.002263	0.094104	0.903633	Jack Coan	Jack is part of the group learning the core co...	2024-05-20
3	4	0.001943	0.049137	0.948920	Ian Thomas	Ian made a few deep catches and another on the...	2024-05-20
4	5	0.001127	0.043991	0.954882	Adam Thielen	Adam connected with Bryce on a few deep passes...	2024-05-20
...	...	...	...	...	...	...	...
92	93	0.002147	0.033955	0.963898	Jordan Love	You feel the confidence from him, the way he p...	2024-05-21
93	94	0.003614	0.074835	0.921550	Josh Jacobs	He's working through a little hamstring issue,...	2024-05-21
94	95	0.008322	0.344314	0.647364	Tucker Kraft	He's going to be out until training camp, and ...	2024-05-21
95	96	0.002269	0.034937	0.962794	Anthony Johnson Jr.	He's done a nice job trying to take his game t...	2024-05-21
96	97	0.001189	0.012235	0.986576	Xavier McKinney	He's a very talented player and a quick learne...	2024-05-21

97 rows × 7 columns

Average the scores for each of the three columns, grouped by the player names.

In [16]:
```
grouped_df = results_df[['name', 'roberta_neg', 'roberta_neu', 'roberta_pos']]

# Calculate the mean of each group
averages_df = grouped_df.mean().reset_index()
averages_df = averages_df.rename(columns={
    'roberta_neg': 'Negative',
    'roberta_neu': 'Neutral',
    'roberta_pos': 'Positive'
})
# Print the resulting dataframe
print(averages_df)
```

```
                   name  Negative   Neutral  Positive
0          Adam Thielen  0.001127  0.043991  0.954882
1       Adonai Mitchell  0.020694  0.691770  0.287536
2           Amare Barno  0.002882  0.110235  0.886884
3           Andy Dalton  0.001935  0.096882  0.901182
4   Anthony Johnson Jr.  0.002269  0.034937  0.962794
..                  ...       ...       ...       ...
80       Xavier Legette  0.014052  0.430930  0.555018
81      Xavier McKinney  0.001189  0.012235  0.986576
82         Xavier Woods  0.002173  0.077798  0.920029
83        Xavier Worthy  0.004107  0.065081  0.930813
84           Zach Evans  0.008223  0.384265  0.607511

[85 rows x 4 columns]
```

Writing these scores into the excel sheet. (old way)

In [11]:
```
# # Read the Excel sheet
# excel_file = 'DataTweets.xlsx'
# sheet_name = 'Scores'

# # Read the existing data from the Excel file
# existing_data = pd.read_excel(excel_file, sheet_name=sheet_name)

# # Update the existing data with the new data for matching columns
# for col in averages_df.columns:
#     if col in existing_data.columns:
#         existing_data[col] = averages_df[col]


# # Load the existing workbook
# workbook = load_workbook(excel_file)

# # Open the existing sheet
# writer = pd.ExcelWriter(excel_file, engine='openpyxl')
# writer.book = workbook
# writer.sheets = {ws.title: ws for ws in workbook.worksheets}

# # Write the updated data back to the existing sheet in the Excel file
# existing_data.to_excel(writer, index=False, sheet_name=sheet_name)

# # Save the changes
# writer.save()
# writer.close()
```

Quick and easy way to run sentiment predictions via the pretrained pipelines that hugging face offers. There are more models that you can specify but this is a quick way of doing it.

In [31]:
```
# from transformers import pipeline
# sent_pipeline = pipeline("sentiment-analysis")
# sent_pipeline(df['Tweet'][15])
```

#Database Connection

In [4]:
```
db_name = "testdb"
db_host = "localhost"
db_username = "kkhatra"
db_password = "password"
```

In [5]:
```
try:
    cnx = mysql.connector.connect(user=db_username, password=db_password,
                                  host=db_host,
                                  database=db_name)
    cursor = cnx.cursor()
except mysql.connector.Error as err:
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        print("Something is wrong with your user name or password")
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        print("Database does not exist")
    else:
        print(err)
else:
    print("connected!")
```

```
connected!
```

Inserting all the data tweets rows into our database

In [39]:
```
add_tweet = ("INSERT INTO quotes"
             "(name, tweet, date)"
             "VALUES (%s, %s, %s)")
```

In [41]:
```
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        tweets = (row['name'], row['tweet'], datetime.now().date())
        cursor.execute(add_tweet, tweets)
    except RuntimeError:
        print('Error')
cnx.commit()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

Retrieving rows back

In [6]:
```
query = ("SELECT * FROM quotes")
cursor.execute(query)
results = cursor.fetchall()

df = pd.DataFrame(results, columns = cursor.column_names)
```

Closing connection

In [66]:
```
cursor.close()
cnx.close()
```

In [19]:
```
df
```

Out[19]:

	id	name	tweet	date
0	1	Bryce Young	Bryce is doing fantastic with the playbook, ma...	2024-05-20
1	2	Andy Dalton	Andy is also showing accuracy in his throws, s...	2024-05-20
2	3	Jack Coan	Jack is part of the group learning the core co...	2024-05-20
3	4	Ian Thomas	Ian made a few deep catches and another on the...	2024-05-20
4	5	Adam Thielen	Adam connected with Bryce on a few deep passes...	2024-05-20
...	...	...	...	...
79	80	Bobby Brown	Bobby Brown is unique to the defensive front a...	2024-05-20
80	81	Darius Williams	Darius Williams is back on the field, and the ...	2024-05-20
81	82	Kobie Durant	Kobie Durant is working hard and trying to get...	2024-05-20
82	83	Cam Curl	Cam Curl, a rookie, was doing DB drills, and t...	2024-05-20
83	84	Darion Kendrick	Darion Kendrick is getting another chance to f...	2024-05-20

84 rows × 4 columns

In [17]:
```
print(averages_df.head())
```

```
                  name  Negative   Neutral  Positive
0         Adam Thielen  0.001127  0.043991  0.954882
1      Adonai Mitchell  0.020694  0.691770  0.287536
2          Amare Barno  0.002882  0.110235  0.886884
3          Andy Dalton  0.001935  0.096882  0.901182
4  Anthony Johnson Jr.  0.002269  0.034937  0.962794
```

Inserting all the averages into our db

In [18]:
```
add_sentiment = ("INSERT INTO sentiment"
            "(name, negative, neutral, positive, overall, date)"
            "VALUES (%s, %s, %s, %s, %s, %s)")
```

In [19]:
```
for i, row in tqdm(averages_df.iterrows(), total=len(averages_df)):
    try:
        overall = (row['Positive'] - row['Negative']) / (row['Positive'] + row
        sentiments = (row['name'], row['Negative'], row['Neutral'], row['Posit
        cursor.execute(add_sentiment, sentiments)
    except RuntimeError:
        print('Error')
cnx.commit()
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [20]:
```
sentiment_query = ("SELECT * FROM sentiment ORDER BY name")
cursor.execute(sentiment_query)
sentiment_results = cursor.fetchall()

sentiment_df_fromdb = pd.DataFrame(sentiment_results, columns = cursor.column_
sentiment_df_fromdb
```

Out[20]:

	name	negative	neutral	positive	overall	date
0	Adam Thielen	0.001127	0.043991	0.954882	95.375484	2024-05-21
1	Adonai Mitchell	0.020694	0.691770	0.287536	26.684249	2024-05-21
2	Amare Barno	0.002882	0.110235	0.886884	88.400245	2024-05-21
3	Andy Dalton	0.001935	0.096882	0.901182	89.924728	2024-05-21
4	Anthony Johnson Jr.	0.002269	0.034937	0.962794	96.052453	2024-05-21
...	...	...	...	...	...	...
80	Xavier Legette	0.014052	0.430930	0.555018	54.096610	2024-05-21
81	Xavier McKinney	0.001189	0.012235	0.986576	98.538722	2024-05-21
82	Xavier Woods	0.002173	0.077798	0.920029	91.785688	2024-05-21
83	Xavier Worthy	0.004107	0.065081	0.930813	92.670562	2024-05-21
84	Zach Evans	0.008223	0.384265	0.607511	59.928799	2024-05-21

85 rows × 6 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: