Creating a dataset for FPL points prediction with 2022/23 season historical data and Twitter sentiment values

This notebook contains code to extract data from the github repo https://github.com/vaastav/Fantasy-Premier-League/tree/master use it, along with Twitter sentiment data, to create a dataset suited to predicting points scored by players in FPL.

The fine-tuned BERT model https://huggingface.co/3style/bert-fpl-twitter trained on FPL twitter data is used to model sentiment for FPL tweets, which is then incorporated as a feature of the final dataset.

Step 1: Download data from github

```
import os
import requests

REPO_URL = "https://raw.githubusercontent.com/vaastav/Fantasy-Premier-League/master/data/2022-23/gws/"

os.makedirs("gameweeks", exist_ok=True)

for gw in range(1, 39):
    filename = f"gw{gw}.csv"
    file_url = REPO_URL + filename

    response = requests.get(file_url)
    if response.status_code == 200:
        with open(f"gameweeks/{filename}", "wb") as f:
            f.write(response.content)
    else:
        print(f"Failed to download {filename}")

print("All files downloaded")
```

```
All files downloaded
```

Step 2: Select and generate relevant features from raw data

Define the dataset features. These are split into 3 categories:

- Information about the match (the player's opponent, home/away fixture, player price changes etc.)
- Player stats from last few games (points scored, goals scored, threat etc.)
- Team stats from last few games (team goals scored, team clean sheets)

```
feature_columns = [
    'points_last_week', 'points_2_weeks_ago', 'points_3_weeks_ago', 'avg_points_last_3_weeks',
    'expected_goal_involvements_last_3_weeks', 'ict_index_last_3_weeks', 'creativity_last_3_weeks',
    'threat_last_3_weeks', 'bps_last_3_weeks', 'minutes_last_week', 'minutes_last_3_weeks',
    'starts_last_3_weeks', 'transfers_in_last_week', 'transfers_out_last_week', 'value_change_last_week',
    'was_home_last_week', 'team_goals_scored_last_3_weeks', 'team_clean_sheets_last_3_weeks',
    'total_points'
]
```

Load the data and compute the rolling features

```
import pandas as pd
import glob

def load_data():
    files = sorted(glob.glob("gameweeks/gw*.csv"), key=lambda x: int(x.split("/")[-1][2:-4]))  # Sort files chronologically
    dfs = []
    for gw, file in enumerate(files, start=1):
        df = pd.read_csv(file)
        df['gameweek'] = gw
        dfs.append(df)
    return pd.concat(dfs, ignore_index=True)


# Compute rolling features
def compute_features(df):
    # Ensure players with the same names are not grouped together by adding team
```

```
    # No players on the same team have the same name in this dataset
    df = df.sort_values(by=['name', 'team', 'gameweek'])

    df['points_last_week'] = df.groupby('name')['total_points'].shift(1)
    df['points_2_weeks_ago'] = df.groupby('name')['total_points'].shift(2)
    df['points_3_weeks_ago'] = df.groupby('name')['total_points'].shift(3)
    df['avg_points_last_3_weeks'] = df.groupby('name')['total_points'].shift(1).rolling(3).mean().round(2)

    df['expected_goal_involvements_last_3_weeks'] = df.groupby('name')['expected_goal_involvements'].shift(1).rolling(3).sum
    df['ict_index_last_3_weeks'] = df.groupby('name')['ict_index'].shift(1).rolling(3).sum().round(2)
    df['creativity_last_3_weeks'] = df.groupby('name')['creativity'].shift(1).rolling(3).sum().round(2)
    df['threat_last_3_weeks'] = df.groupby('name')['threat'].shift(1).rolling(3).sum()
    df['bps_last_3_weeks'] = df.groupby('name')['bps'].shift(1).rolling(3).sum()

    df['minutes_last_week'] = df.groupby('name')['minutes'].shift(1)
    df['minutes_last_3_weeks'] = df.groupby('name')['minutes'].shift(1).rolling(3).sum()
    df['starts_last_3_weeks'] = df.groupby('name')['starts'].shift(1).rolling(3).sum()

    df['transfers_in_last_week'] = df.groupby('name')['transfers_in'].shift(1)
    df['transfers_out_last_week'] = df.groupby('name')['transfers_out'].shift(1)
    df['value_change_last_week'] = df.groupby('name')['value'].diff()

    df['was_home_last_week'] = df.groupby('name')['was_home'].shift(1)

    # Compute team-level goals scored and clean sheets per gameweek
    team_goals = df.groupby(['team', 'gameweek'])['goals_scored'].sum().reset_index()
    team_clean_sheets = df.groupby(['team', 'gameweek'])['clean_sheets'].max().reset_index()  # Use max in case of duplicate

    # Compute rolling sum of team goals and clean sheets
    team_goals['team_goals_scored_last_3_weeks'] = team_goals.groupby('team')['goals_scored'].shift(1).rolling(3).sum()
    team_clean_sheets['team_clean_sheets_last_3_weeks'] = team_clean_sheets.groupby('team')['clean_sheets'].shift(1).rolling

    # Merge back into main dataframe
    df = df.merge(team_goals[['team', 'gameweek', 'team_goals_scored_last_3_weeks']], on=['team', 'gameweek'], how='left')
    df = df.merge(team_clean_sheets[['team', 'gameweek', 'team_clean_sheets_last_3_weeks']], on=['team', 'gameweek'], how='l

    return df[['name', 'team', 'gameweek'] + feature_columns]
```

Save the engineered data, will be used to train the baseline model

```
data = load_data()
data = compute_features(data)

# Drop rows with NaN values from rolling calculations
data = data.dropna()

data.to_csv("data.csv", index=False)
```

```
<ipython-input-3-ec9cb0e24206>:11: FutureWarning: The behavior of DataFrame concatenation with empty or all-NA entries i
  return pd.concat(dfs, ignore_index=True)
```

Examine the data to ensure correct values

```
data.shape
```

```
(24139, 22)
```

```
data.iloc[1000:1008, 0:20]
```

	name	team	gameweek	points_last_week	points_2_weeks_ago	points_3_weeks_ago	avg_points_last_3_weeks	expected
1096	Alphonse Areola	West Ham	14	0	0	0	0.00	
1097	Alphonse Areola	West Ham	15	1	0	0	0.33	
1098	Alphonse Areola	West Ham	16	0	1	0	0.33	
1099	Alphonse Areola	West Ham	17	0	0	1	0.33	
1100	Alphonse Areola	West Ham	18	0	0	0	0.00	
1101	Alphonse Areola	West Ham	19	0	0	0	0.00	
1102	Alphonse Areola	West Ham	20	0	0	0	0.00	
1103	Alphonse Areola	West Ham	21	0	0	0	0.00	

Step 3: Add sentiment predictions from FPL Twitter model

Take the unseen raw twitter data and run it through the model to derive sentiment. Then match each tweet to a player and gameweek, to generate a positive_tweets and negative_tweets value for each entry in the dataset

```
%pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.3.2-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from datasets) (3.17.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in /usr/local/lib/python3.11/dist-packages (from fsspec[http
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.13)
Requirement already satisfied: huggingface-hub>=0.24.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (0.28.
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->dataset
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (25.1.0
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-h
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->dat
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->dat
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas-
Downloading datasets-3.3.2-py3-none-any.whl (485 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 485.4/485.4 kB 15.1 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 11.7 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 12.6 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 17.2 MB/s eta 0:00:00
Installing collected packages: xxhash, dill, multiprocess, datasets
Successfully installed datasets-3.3.2 dill-0.3.8 multiprocess-0.70.16 xxhash-3.5.0
```

```
from datasets import load_dataset

twitter_data = load_dataset("csv", data_files="https://huggingface.co/3style/bert-fpl-twitter/resolve/main/unseen_data.csv")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
unseen_data.csv: 100%                                   26.1M/26.1M [00:00<00:00, 39.5MB/s]

Generating train split:      147701/0 [00:00<00:00, 250564.27 examples/s]
```

```
import pandas as pd

# Load data
twitter_df = pd.DataFrame(twitter_data["train"])

# Remove unneeded columnmns
twitter_df = twitter_df.drop(columns=["vader_polarity", "vader_emotion", "tb_polarity", "tb_emotion", "tweet_date", "when"])

# Convert name casing to capitalised
twitter_df["player_name"] = twitter_df["player_name"].str.title()

# Show results
twitter_df.head()
```

	player_name	text	game_date
0	Keylor Navas	Exactly that s even for the elite Look at keyl...	2023-02-25
1	Christian Pulisic	Is the usmnt player laughing like Chelsea didn...	2023-04-18
2	Junior Firpo	Barcelona Were Close To Signing Inter Milan St...	2023-02-8
3	Kai Havertz	We Have Super Kai Havertz for you Bayern abeg ...	2023-05-2
4	David De Gea	Manchester United 0 Southampton 0 on Sunday Ca...	2023-03-16

Tag each tweet with the FPL 'gameweek' it belongs to

Each row in the dataset has a 'game_date' the tweet relates to, we can map this to a gameweek by taking the fixtures data and working out the range of dates each gameweek falls into.

```
import pandas as pd

df = pd.read_csv("https://raw.githubusercontent.com/vaastav/Fantasy-Premier-League/master/data/2022-23/fixtures.csv")

# Calculate the first and last kickoff time for each 'event' (gameweek)
df["kickoff_time"] = pd.to_datetime(df["kickoff_time"])
gw_date_ranges = df.groupby("event")["kickoff_time"].agg([min, max]).reset_index()

# Rename columns
gw_date_ranges.columns = ["gameweek", "start_date", "end_date"]

# Remove timezone information
gw_date_ranges["start_date"] = gw_date_ranges["start_date"].dt.tz_localize(None)
gw_date_ranges["end_date"] = gw_date_ranges["end_date"].dt.tz_localize(None)

# Preview Results
gw_date_ranges.head()
```

```
<ipython-input-11-6c81096c9a48>:7: FutureWarning: The provided callable <built-in function min> is currently using Serie
  gw_date_ranges = df.groupby("event")["kickoff_time"].agg([min, max]).reset_index()
<ipython-input-11-6c81096c9a48>:7: FutureWarning: The provided callable <built-in function max> is currently using Serie
  gw_date_ranges = df.groupby("event")["kickoff_time"].agg([min, max]).reset_index()
```

	gameweek	start_date	end_date
0	1	2022-08-05 19:00:00	2022-08-07 15:30:00
1	2	2022-08-13 11:30:00	2022-08-15 19:00:00
2	3	2022-08-20 11:30:00	2022-08-22 19:00:00
3	4	2022-08-27 11:30:00	2022-08-28 15:30:00
4	5	2022-08-30 18:30:00	2022-09-01 19:00:00

Next steps: Generate code with gw_date_ranges | View recommended plots | New interactive sheet

```
import pandas as pd

# Convert 'game_date' to datetime format
twitter_df["game_date"] = pd.to_datetime(twitter_df["game_date"])

# Merge on the closest previous `start_date`
twitter_df = pd.merge_asof(
    twitter_df.sort_values("game_date"),  # Sort before merging
    gw_date_ranges[["gameweek", "start_date", "end_date"]].sort_values("start_date"),
    left_on="game_date",
    right_on="start_date",
    direction="backward"  # Match the closest past start_date
)

# Ensure the game_date falls within the correct gameweek range
twitter_df = twitter_df[twitter_df["game_date"] <= twitter_df["end_date"]]

# Drop unnecessary columns
twitter_df = twitter_df.drop(columns=["start_date", "end_date", "game_date"])

# Convert gameweek to integer
twitter_df["gameweek"] = twitter_df["gameweek"].astype(int)

# Display results
twitter_df.head()
```

	player_name	text	gameweek
1397	Paul Dummett	Mate wait till you find out about a href PaulD...	1
1398	N'Golo Kant🅾	Administrator you who have good contacts we ne...	1
1399	N'Golo Kant🅾	I remember when Arsenal fans compared this guy...	1
1400	Diego Carlos	It s very careless to ignore the fact that Vil...	1
1401	Cafu	Maicon is the best RB EVER and even better tha...	1

Next steps: Generate code with twitter_df | View recommended plots | New interactive sheet

Run tweets through sentiment analysis model to get sentiment values

```
# Construct the auxiliary sentence
twitter_df['question'] = twitter_df['player_name'].apply(lambda x: f"What do you think of the sentiment towards {x}?")
twitter_df.head()
```

	player_name	text	gameweek	question
1397	Paul Dummett	Mate wait till you find out about a href PaulD...	1	What do you think of the sentiment towards Pau...
1398	N'Golo Kant🅾	Administrator you who have good contacts we ne...	1	What do you think of the sentiment towards N'G...
1399	N'Golo Kant🅾	I remember when Arsenal fans compared this guy...	1	What do you think of the sentiment towards N'G...
1400	Diego Carlos	It s very careless to ignore the fact that Vil...	1	What do you think of the sentiment towards Die...
1401	Cafu	Maicon is the best RB EVER and even better tha...	1	What do you think of the sentiment towards Cafu?

Next steps: Generate code with twitter_df | View recommended plots | New interactive sheet

```
from transformers import AutoTokenizer
from datasets import Dataset
import pandas as pd

# Load tokeniser
tokeniser = AutoTokenizer.from_pretrained("bert-base-cased")

# Convert dataframe to dataset object
dataset = Dataset.from_pandas(twitter_df)

def tokenise_function(input):
    return tokeniser(input["question"], input["text"], padding="max_length", truncation=True)

# Tokenise the text
tokenised_dataset = dataset.map(tokenise_function, batched=True)
```

```
The cache for model files in Transformers v4.22.0 has been updated. Migrating your old cache. This is a one-time only op
0/0 [00:00<?, ?it/s]
tokenizer_config.json: 100% 49.0/49.0 [00:00<00:00, 4.86kB/s]
config.json: 100% 570/570 [00:00<00:00, 45.9kB/s]
vocab.txt: 100% 213k/213k [00:00<00:00, 6.75MB/s]
tokenizer.json: 100% 436k/436k [00:00<00:00, 33.6MB/s]
Map: 100% 75313/75313 [00:29<00:00, 2796.14 examples/s]
```

```
from transformers import BertForSequenceClassification
import torch

# Utilise GPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = BertForSequenceClassification.from_pretrained("3style/bert-fpl-twitter", subfolder="checkpoint-2394")
model = model.to(device)  # Move model to the GPU

model.eval()

# Set the batch size
batch_size = 32
num_batches = len(tokenised_dataset) // batch_size + 1

# Initialize an empty list to store the results
predicted_labels = []

# Loop over batches
for i in range(num_batches):
    # Get the batch of data
    batch = tokenised_dataset[i * batch_size: (i + 1) * batch_size]

    # Prepare input tensors
    input_ids = torch.tensor(batch["input_ids"]).to(device)
    attention_mask = torch.tensor(batch["attention_mask"]).to(device)

    # Perform inference
    with torch.no_grad():
        outputs = model(input_ids=input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        predictions = logits.argmax(dim=-1)

    # Define the reverse label mapping
    reverse_label_mapping = {0: "positive", 1: "negative", 2: "neutral"}

    # Map predictions to sentiment labels
    predicted_labels.extend([reverse_label_mapping[int(pred)] for pred in predictions])

# Display results
predicted_labels
```

```
config.json: 100% 891/891 [00:00<00:00, 59.0kB/s]
model.safetensors: 100% 433M/433M [00:09<00:00, 52.3MB/s]
```

```
# Add the predictions to the dataframe as a new column
twitter_df["sentiment"] = predicted_labels

# Drop unneccessary columns
twitter_df = twitter_df.drop(columns=["question"])

# Display the first few rows to verify
twitter_df.head()
```

	player_name	text	gameweek	sentiment
1397	Paul Dummett	Mate wait till you find out about a href PaulD...	1	neutral
1398	N'Golo Kant&#65533;	Administrator you who have good contacts we ne...	1	positive
1399	N'Golo Kant&#65533;	I remember when Arsenal fans compared this guy...	1	neutral
1400	Diego Carlos	It s very careless to ignore the fact that Vil...	1	positive
1401	Cafu	Maicon is the best RB EVER and even better tha...	1	positive

Match twitter data with the main dataset, add positive_tweets and negative_tweets features to the dataset

```
%pip install rapidfuzz
```

Show hidden output

```
import pandas as pd
from rapidfuzz import process, fuzz

# Normalize player names for better matching
data["name_normalized"] = data["name"].str.lower().str.strip()
twitter_df["player_name_normalized"] = twitter_df["player_name"].str.lower().str.strip()

# Create a dictionary to store matches
name_mapping = {}

# Find best fuzzy match (matching library that accounts for typos) for each player
for player in twitter_df["player_name_normalized"].unique():
    match, score, _ = process.extractOne(player, data["name_normalized"], scorer=fuzz.partial_ratio)
    if score >= 80:  # Only consider high confidence matches
        name_mapping[player] = match

# Apply fuzzy matches to twitter_df
twitter_df["matched_name"] = twitter_df["player_name_normalized"].map(name_mapping)

# Drop rows where no good match was found
twitter_df = twitter_df.dropna(subset=["matched_name"])

# Count positive and negative tweets for each matched player and gameweek
sentiment_counts = twitter_df.groupby(["matched_name", "gameweek"])["sentiment"].value_counts().unstack(fill_value=0)

# Keep only positive and negative tweets
sentiment_counts = sentiment_counts[["positive", "negative"]]

# Rename columns
sentiment_counts = sentiment_counts.rename(columns={"positive": "positive_tweets", "negative": "negative_tweets"}).reset_inc

# Merge with `data`, using fuzzy-matched names
data = data.merge(sentiment_counts, left_on=["name_normalized", "gameweek"], right_on=["matched_name", "gameweek"], how="lef

# Drop temporary columns
data = data.drop(columns=["matched_name", "name_normalized"], errors="ignore")

# Fill NaN values with 0 (for players without tweets)
data["positive_tweets"] = data["positive_tweets"].fillna(0).astype(int)
data["negative_tweets"] = data["negative_tweets"].fillna(0).astype(int)
```

	name	team	gameweek	points_last_week	points_2_weeks_ago	points_3_weeks_ago	avg_points_last_3_weeks	expected_go
0	Aaron Cresswell	West Ham	4	0	2	1	1.00	
1	Aaron Cresswell	West Ham	5	6	0	2	2.67	
2	Aaron Cresswell	West Ham	6	2	6	0	2.67	
3	Aaron Cresswell	West Ham	8	0	2	6	2.67	
4	Aaron Cresswell	West Ham	9	2	0	2	1.33	

5 rows × 24 columns

```
Warning: Total number of columns (24) exceeds max_columns (20) limiting to first (20) columns.
```

```
sentiment_data = data[["name", "gameweek", "positive_tweets", "negative_tweets", "total_points"]]
print(sentiment_data.head(35))
```

```
              name  gameweek  positive_tweets  negative_tweets total_points
0   Aaron Cresswell         4                0                0            6
1   Aaron Cresswell         5                0                0            2
2   Aaron Cresswell         6                0                0            0
3   Aaron Cresswell         8                0                0            2
4   Aaron Cresswell         9                0                0            5
5   Aaron Cresswell        10                0                0            2
6   Aaron Cresswell        11                0                0            3
7   Aaron Cresswell        12                0                0            2
```

8	Aaron Cresswell	13	0	0	8
9	Aaron Cresswell	14	0	0	2
10	Aaron Cresswell	15	0	0	1
11	Aaron Cresswell	16	0	0	1
12	Aaron Cresswell	17	0	0	1
13	Aaron Cresswell	18	0	0	1
14	Aaron Cresswell	19	13	2	1
15	Aaron Cresswell	20	23	2	2
16	Aaron Cresswell	21	0	0	0
17	Aaron Cresswell	22	0	0	0
18	Aaron Cresswell	23	0	0	0
19	Aaron Cresswell	24	0	0	0
20	Aaron Cresswell	25	24	0	1
21	Aaron Cresswell	26	0	0	0
22	Aaron Cresswell	27	0	0	0
23	Aaron Cresswell	29	23	0	1
24	Aaron Cresswell	29	23	0	0
25	Aaron Cresswell	30	0	0	8
26	Aaron Cresswell	31	24	1	0
27	Aaron Cresswell	32	17	6	12
28	Aaron Cresswell	33	22	6	1
29	Aaron Cresswell	34	5	0	1
30	Aaron Cresswell	34	5	0	1
31	Aaron Cresswell	35	0	0	6
32	Aaron Cresswell	36	0	0	0
33	Aaron Cresswell	37	0	0	0
34	Aaron Cresswell	38	0	0	1

```
data.to_csv("data_with_sentiment.csv", index=False)
```

Step 4: Upload dataset to hugging face

Add the datasets both with and without sentiment values to a hugging face repo for training a point predicting model

```
from huggingface_hub import notebook_login
notebook_login()
```

```
from huggingface_hub import create_repo

repo_name = "fpl-expected-points"
create_repo(repo_name, private=False)
```

```
RepoUrl('https://huggingface.co/3style/fpl-expected-points', endpoint='https://huggingface.co', repo_type='model',
 repo_id='3style/fpl-expected-points')
```

```
import os

# Create the 'datasets' directory if it doesn't exist
if not os.path.exists('datasets'):
    os.makedirs('datasets')

# Move the files
!mv data.csv datasets/
!mv data_with_sentiment.csv datasets/
```

```
from huggingface_hub import upload_folder

upload_folder(
    folder_path="datasets",
    path_in_repo=".",
    repo_id="3style/fpl-expected-points"
)
```