

Sentiment Analysis

Twitter on Climate Change

By Group 3

BANERJEE Hiya

ESPEDAL Terje

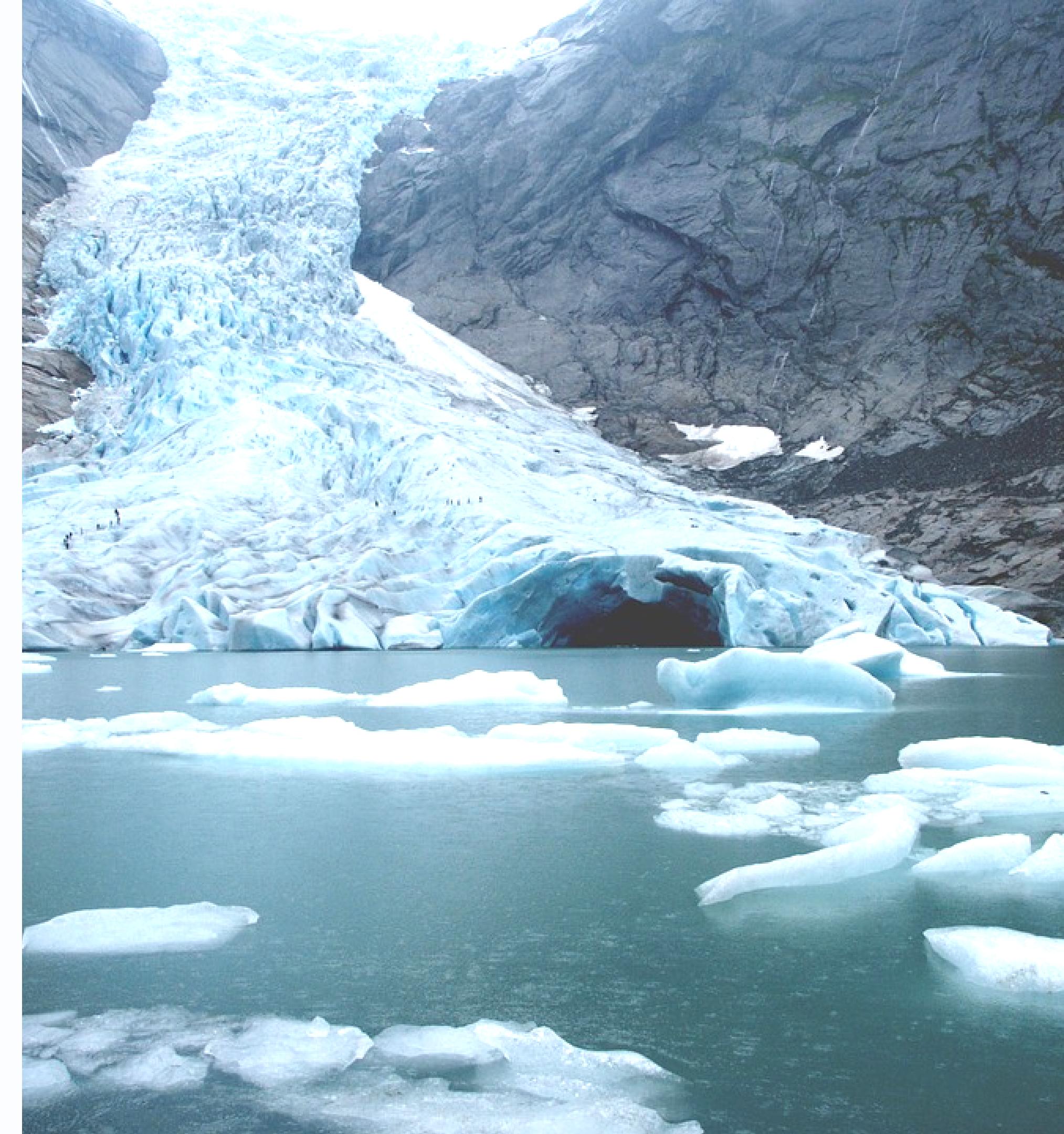
PUNRATTANAPONGS Jitrayu

KASAMANI Serene

EL-HAGE HAMMOUD Samer

STEPHENS Allison

XUE Rui





Introduction

Climate change is a growing issue that requires everyone's immediate attention. In recent years, we have witnessed rising temperatures and sea-levels, eruptions of wildfires in various regions and unstable storm conditions.

With most businesses adopting sustainable and eco-friendly services and products, it is critical to understand people's perception of this ongoing crisis to formulate better strategies for businesses.

To achieve this, we analyzed tweets from Twitter to gain valuable insights towards people's sentiments. The data is sourced from Kaggle and the Twitter Standard API.

Research Questions

01

How has the trend in climate change awareness shifted over time?

We asses two datasets belonging to two different time periods to answer this.

02

What is the ruling sentiment amongst people towards climate change?

By labeling climate change tweets as either positive, negative, or neutral, we aim to better understand the general sentiment towards climate change.

About the dataset



LABELLED DATASET

This dataset is originally from the University of Waterloo. The project was funded by Canada Foundation for Innovation. The dataset contained tweets between 27 Apr 2015 to 21 Feb 2018.

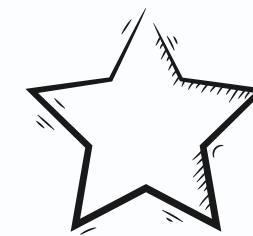
SIZE: 43943 x 3



UNLABELLED DATASET

This unlabelled dataset was scraped tweets that have keywords, such as #climatechange, #energy, #cleanenergy, etc, during the period of this project. The dataset is unlabelled.

SIZE: 99940 x 159



TARGET ATTRIBUTE

Our target attribute for this project is **sentiment about climate change**, which is labelled into 4 classes: News, Pro, Neutral, and Anti.



Data Cleaning & Processing

Data Processing

LABELLED DATA

Load the dataset we get from Kaggle

Get metadata of the tweets in dataset from Twitter API by tweetid

some columns are nested with a list or dict, unnested by customer function

drop missing target var and binary encoded some columns

Fill NA values with mod of each columns



LOAD

EXTRACT

UNNESTED

TRANSFORM

FILL NA

MERGE

scrap tweets by Twiter API according to the search terms.

json_normalize the JSON file input get from Twitter API

drop unrelated columns and binary encoded some columns

Fill NA values with mod of each columns

UNLABELLED DATA

```
1 def get_metadata_for_tweets(df, id_col, chunk_n=100, tweet_fields=TWEET_FIELDS,
2                             expansions=EXPANSIONS, user_fields=USER_FIELDS,
3                             poll_fields=POLL_FIELDS, place_fields=PLACE_FIELDS,
4                             media_fields=MEDIA_FIELDS):
5     """
6         Gets metadata from the twitter API for a dataframe containing at minimum a
7         list of tweet ids. By default divides dataframe into chunks of 100 which
8         is max query for Twitter API. Accepts tweet_field and tweet_expansions
9         as parameters. See Twitter API for a list of accepted fields and expansions.
10        https://developer.twitter.com/en/docs/twitter-api/fields
11        https://developer.twitter.com/en/docs/twitter-api/expansions
12
13    param: df Pandas dataframe containing at minimum tweetids
14    param: id_col Column containing tweet ids
15    param: chunk_n Number of rows per chunk to divide dataframe into.
16        by default 100. This corresponds to the max accepted by Twitter API
17    param: tweet_fields List of extra fields to get from Twitter API
18    param: expansions List of expansions to get from Twitter API
19    """
20
21    chunks = [df[i:i+chunk_n].copy() for i in range(0,df.shape[0],chunk_n)]
22    joined_dfs = []
23
24    for chunk in chunks:
25        # Join ids by "," since query needs to be a string of ids separated by ","
26        ids = ",".join(chunk[id_col].astype("str").values)
27
28        response = client.get_tweets(ids, tweet_fields=tweet_fields, expansions=expansions,
29                                      media_fields=media_fields, user_fields=user_fields,
30                                      poll_fields=poll_fields, place_fields=place_fields)
31        # Expand Tweepy response object into dataframe for easier merging
32        tweets = pd.DataFrame(response.data)
33        joined_dfs.append(chunk.merge(tweets, on="id", how="left"))
34
35    return pd.concat(joined_dfs).reset_index(drop=True)
```

get_metadata_for_tweets()

This custom function is the function that we used to get the metadata of the labelled tweets according to tweetid.

```
 1 def transform_references_binary(df, col_prefix, id_col=None):
 2     cols = [col for col in df.columns if col_prefix in col]
 3     if not id_col:
 4         id_col = f"{col_prefix}_id"
 5     print(id_col, cols)
 6     binar_col = (df.copy()
 7                 .loc[:, id_col]
 8                 .fillna(0)
 9                 .apply(lambda x: np.where(x == 0, 0, 1))
10                 .rename(col_prefix)
11                 )
12
13     transformed_df = (pd.concat([df, binar_col], axis=1)
14                         .drop(cols, axis=1)
15                         )
16
17     return transformed_df
18
19 unlabelled= (unlabelled.loc[:,~unlabelled.columns.duplicated()])
20         .pipe(transform_references_binary, "referenced_tweets")
21         .pipe(transform_references_binary, "in_reply_to_user")
22         .pipe(transform_references_binary, "attachment_poll", "attachment_poll.id")
23         .pipe(transform_references_binary, "attachment_media", "attachment_media")
24         )
25
```

transform_references_binary()

This custom function is used to convert multi-column information in the dataset into single binary columns.



Clustering

OUR GOAL:



Performing
UNSUPERVISED
CLUSTERING to
assign labels to the
New Unlabeled
Tweets according to
the defined classes

●

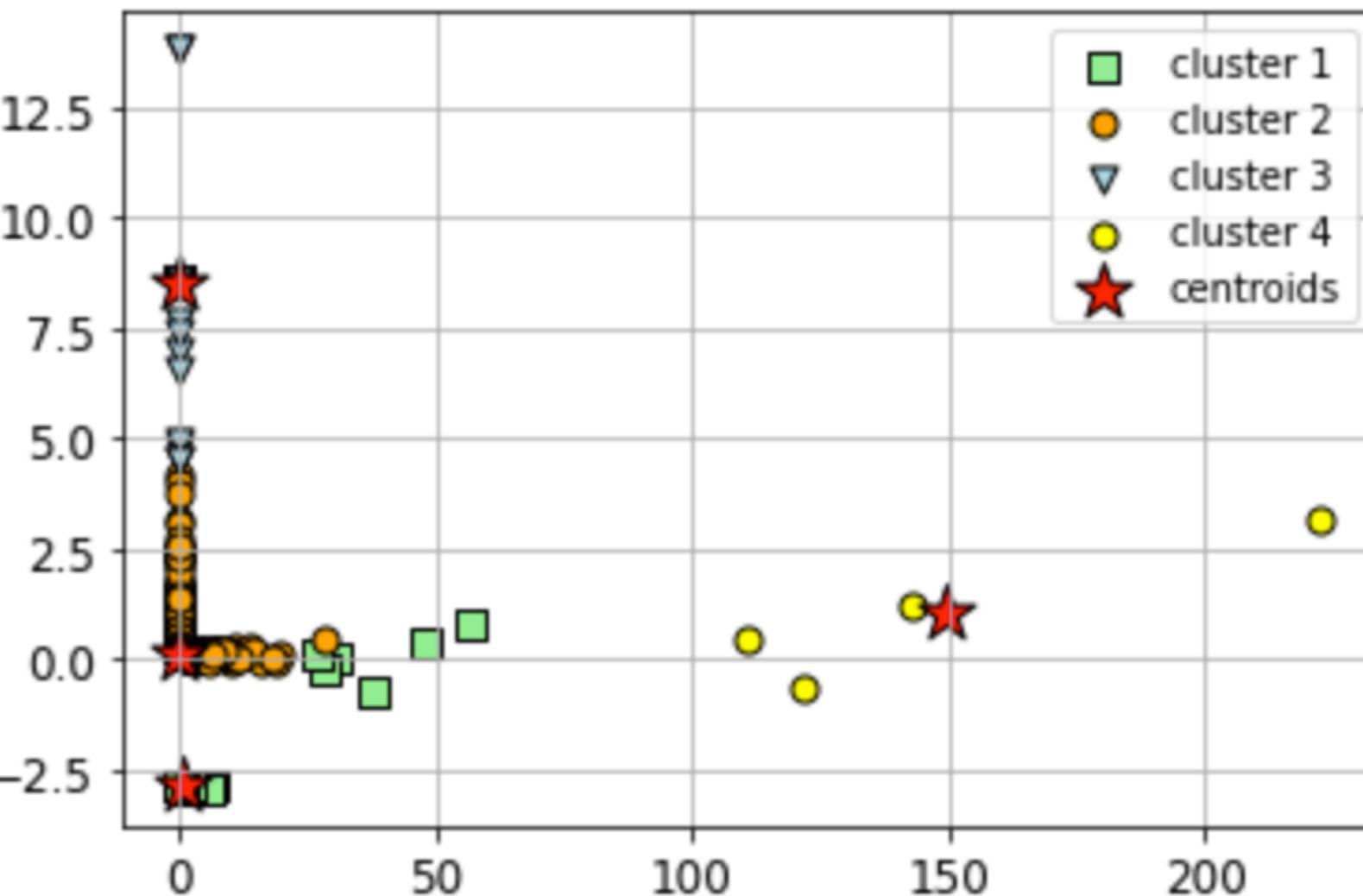
●

●

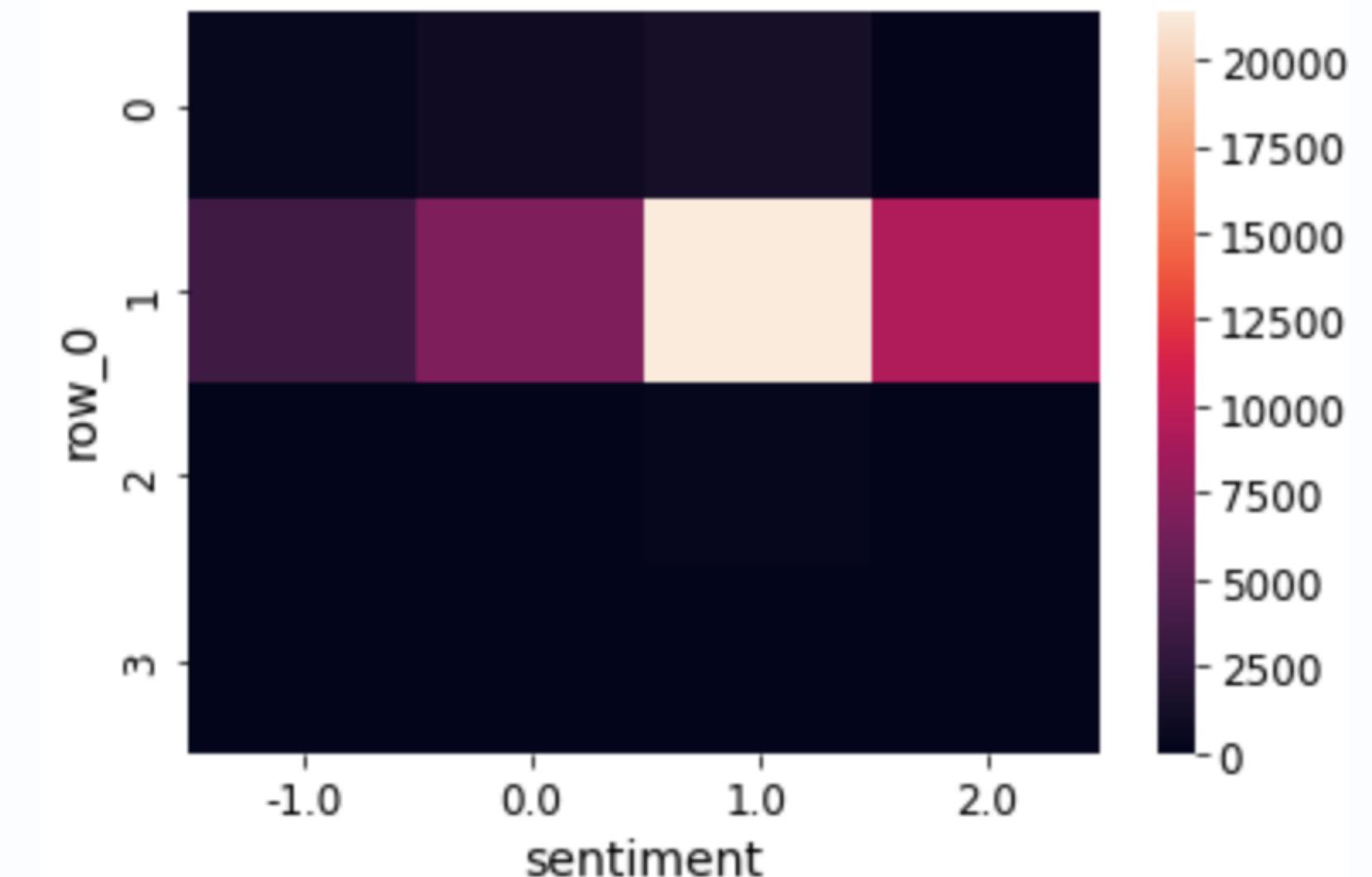
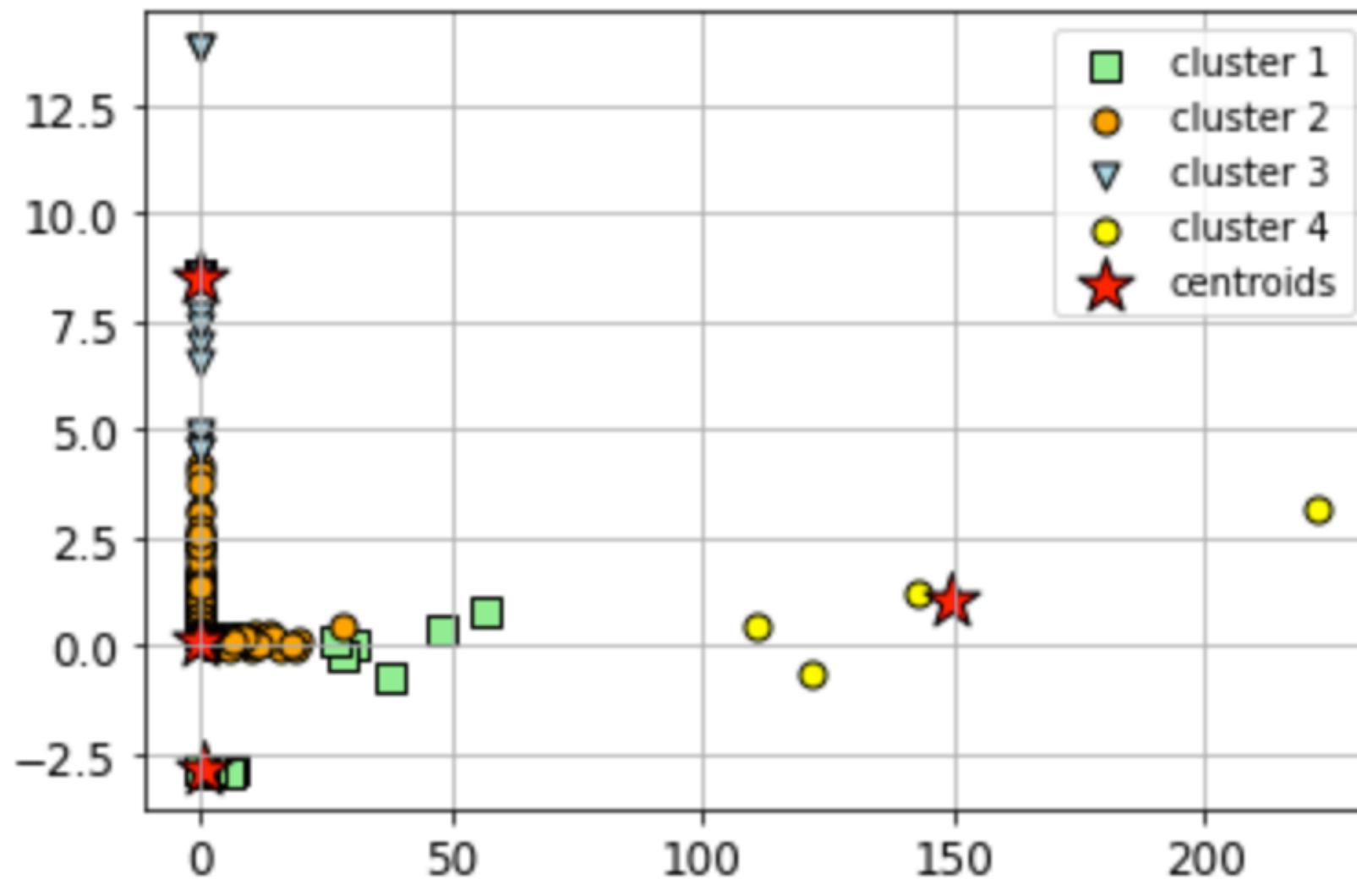
```
1 from sklearn.cluster import KMeans
2
3 km_cluster = KMeans(
4     n_clusters=4, init='random',
5     n_init = 20, max_iter= 10_000,
6     tol= 1e-04, random_state=0
7 )
8
9
10 y_km = km_cluster.fit_predict(labelled_X2D)
11
12 plt.scatter(
13     labelled_X2D[y_km==0]['x'], labelled_X2D[y_km==0]['y'],
14     s=50, c='lightgreen',
15     marker='s', edgecolor='black',
16     label='cluster 1'
17 )
18
19 plt.scatter(
20     labelled_X2D[y_km == 1]['x'], labelled_X2D[y_km == 1]['y'],
21     s=50, c='orange',
22     marker='o', edgecolor='black',
23     label='cluster 2'
24 )
25
26 plt.scatter(
27     labelled_X2D[y_km == 2]['x'], labelled_X2D[y_km == 2]['y'],
28     s=50, c='lightblue',
29     marker='v', edgecolor='black',
30     label='cluster 3'
31 )
32
33 plt.scatter(
34     labelled_X2D[y_km == 3]['x'], labelled_X2D[y_km == 3]['y'],
35     s=50, c='yellow',
36     marker='o', edgecolor='black',
37     label='cluster 4'
38 )
39
40 plt.scatter(
41     km_cluster.cluster_centers_[:, 0], km_cluster.cluster_centers_[:, 1],
42     s=250, marker="*",
43     c='red', edgecolor='black',
44     label= 'centroids'
45 )
46
47 plt.legend(scatterpoints=1)
48 plt.grid()
49 plt.show()
```

Clustering and plotting the cluster

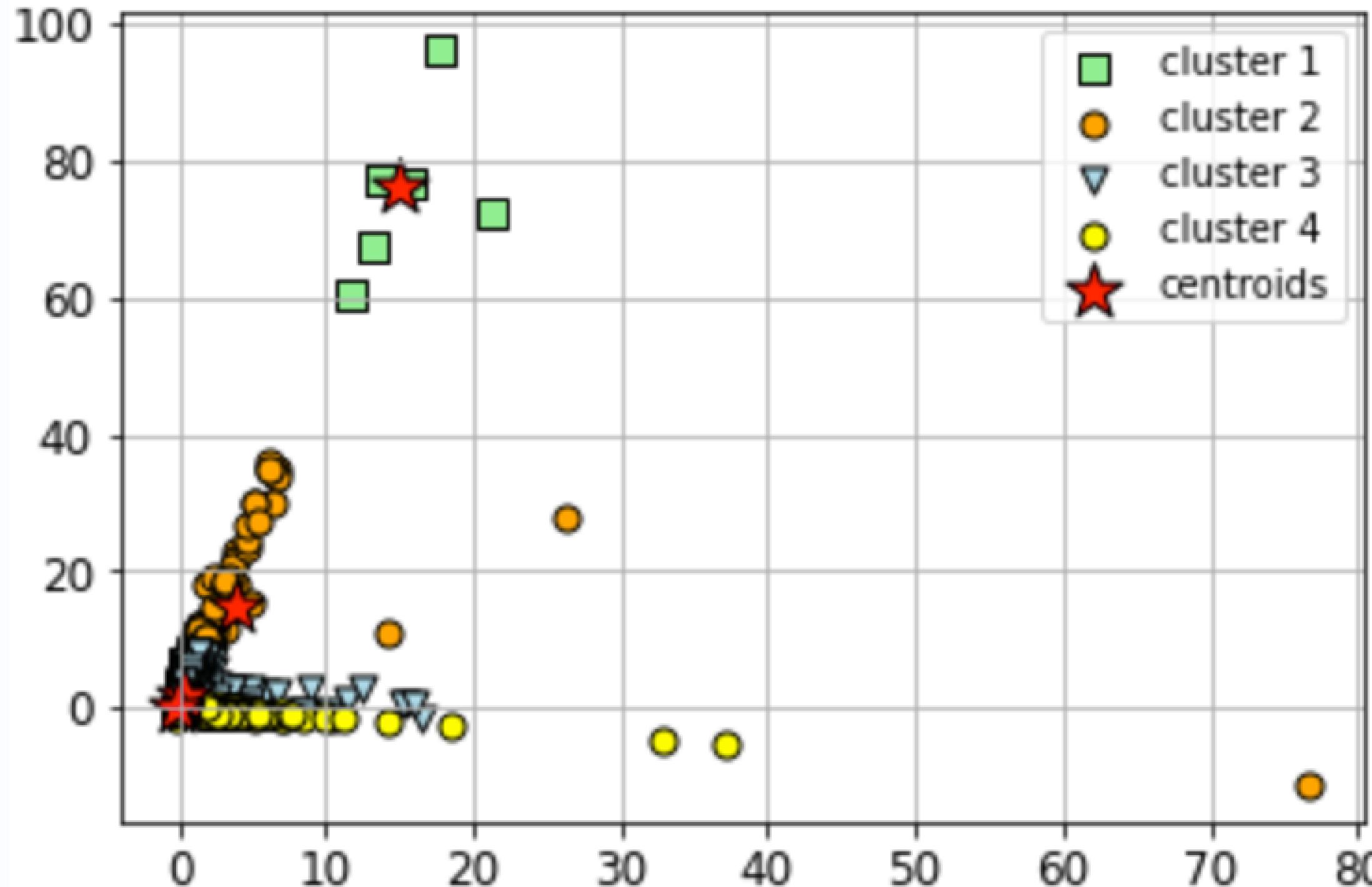
The following code are to create a cluster with KMeans clustering and then plot the cluster with their centroids.



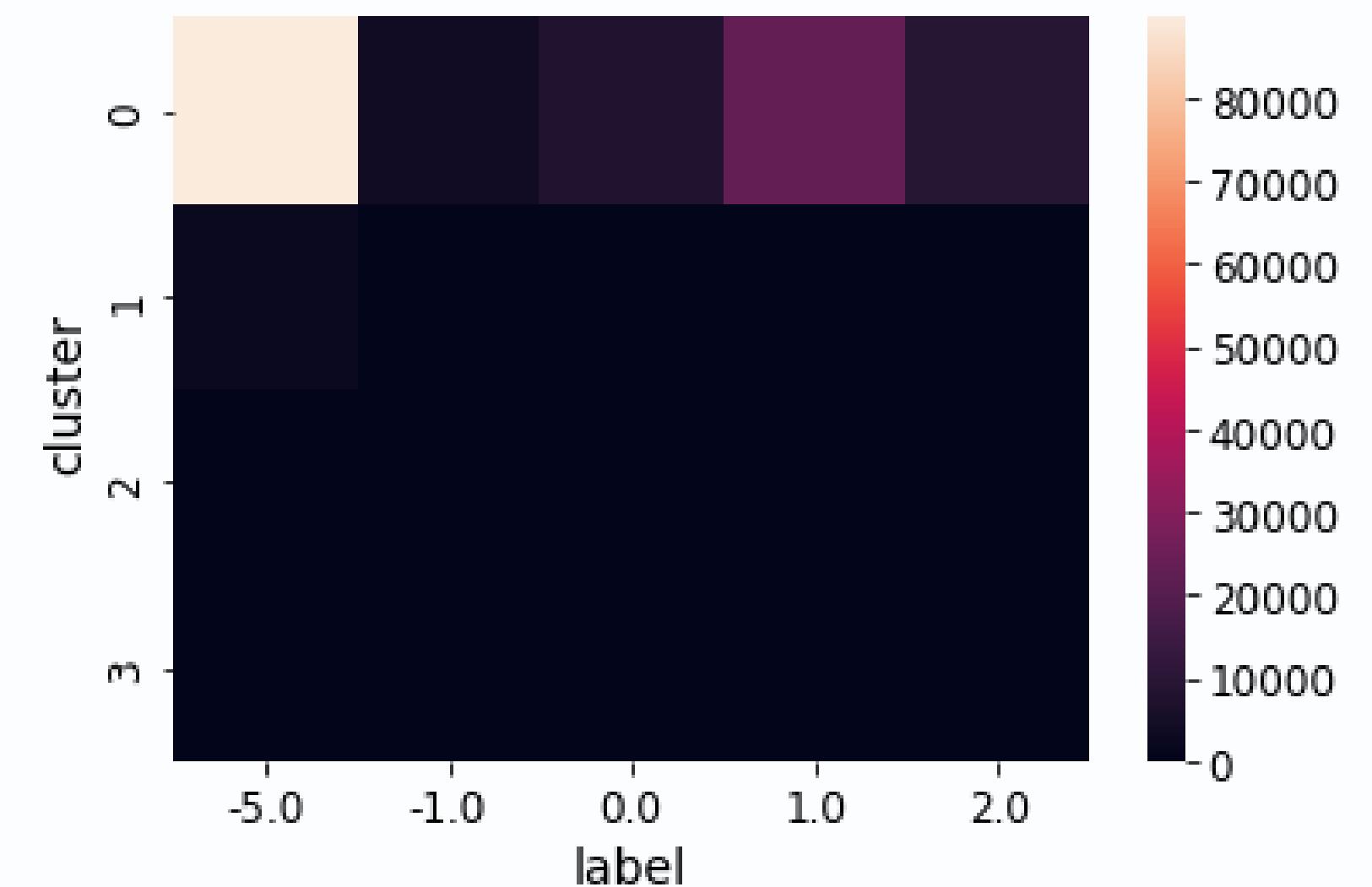
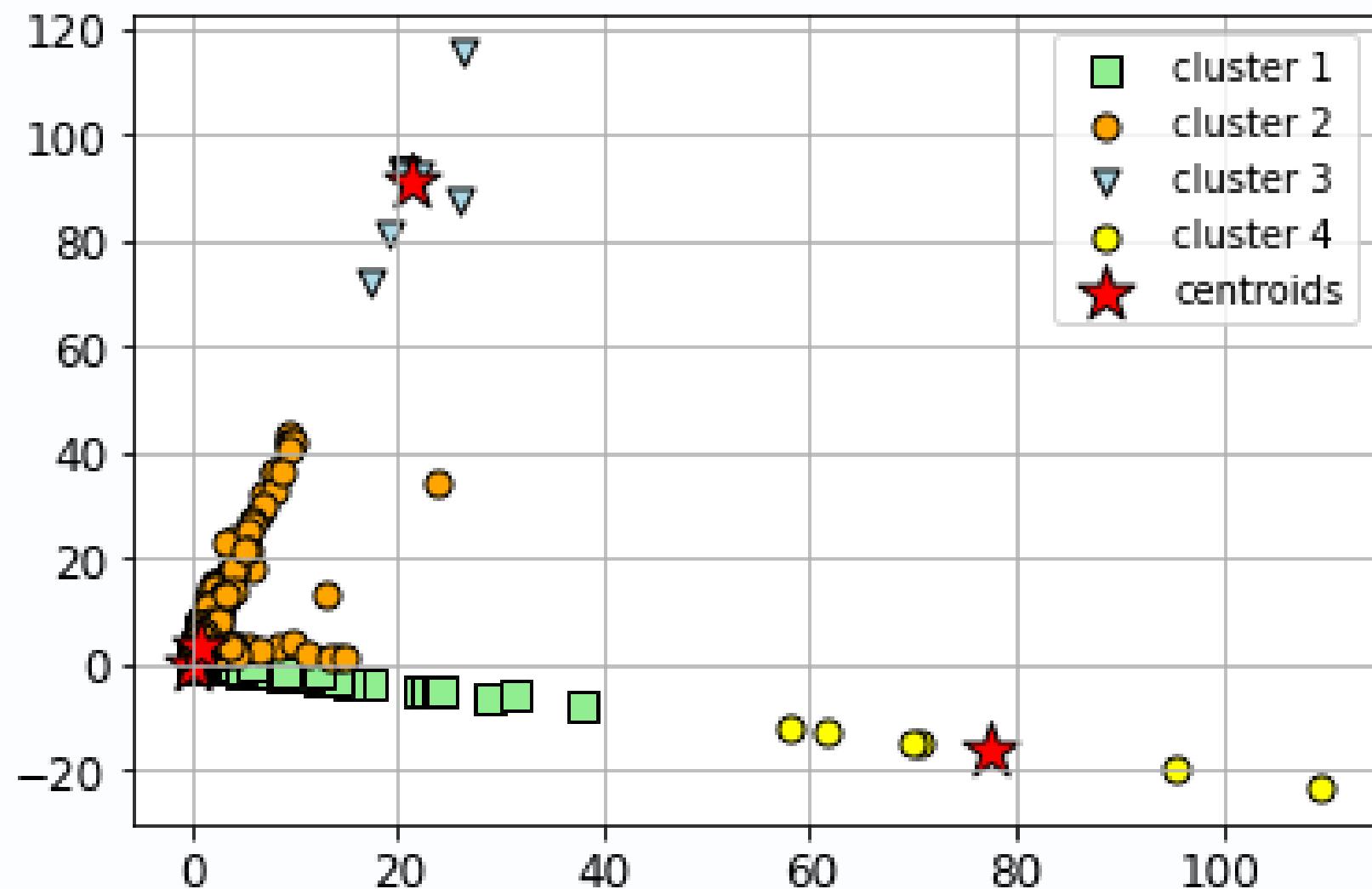
Labelled Dataset



Unlabeled Dataset



Semi-Labelled Dataset



Modelling

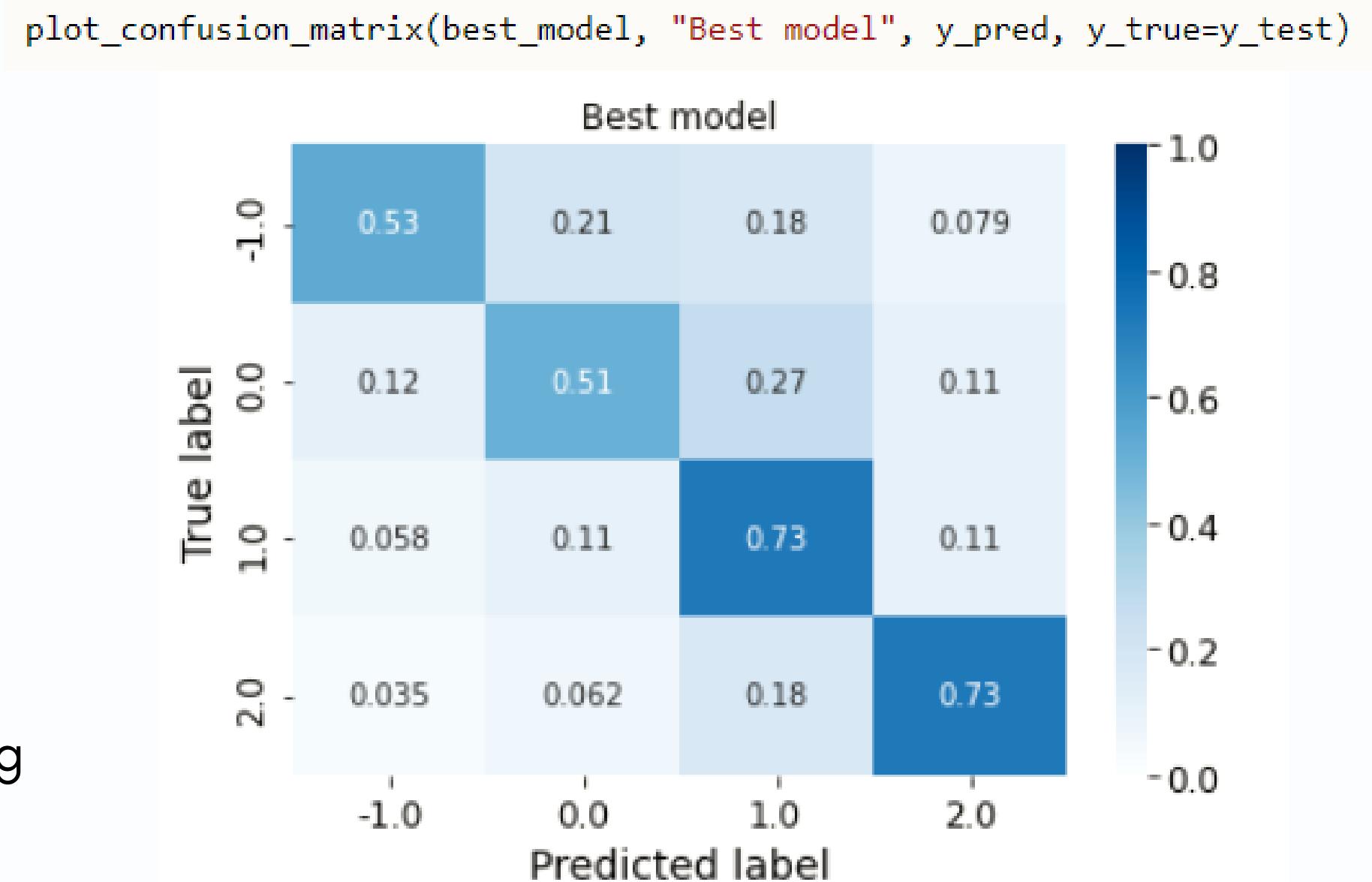
Model Selection

Models selected:

- i. Forest model
- ii. KNN model
- iii. Linear SVC model

Performance of each model

- i. F1-Score 0.36 (weighted average)
- ii. F1-Score 0.53 (weighted average)
- iii. F1-Score 0.59 (weighted average). 0.67 after tuning



```
1 from sklearn.pipeline import Pipeline
2 from sklearn.feature_extraction.text import TfidfVectorizer
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.ensemble import RandomForestClassifier
5 from sklearn.svm import LinearSVC
6 from sklearn.compose import ColumnTransformer
7 from sklearn.model_selection import cross_val_predict
8 from sklearn.metrics import balanced_accuracy_score, f1_score, precision_score, recall_score,
    confusion_matrix
9
10 transformer = ColumnTransformer(
11     [('vec', TfidfVectorizer(stop_words= 'english'), 'message')], # column should be a string or
    int
12     remainder='passthrough'
13 )
14
15 # Random Forest Classifier
16 rfc = Pipeline([('tfidf', transformer),
17                 ('scaler', StandardScaler(with_mean=False)),
18                 ('clf', RandomForestClassifier(max_depth=5,
19                                         n_estimators=100,
20                                         random_state=0))])
21
22 # K-NN Classifier
23 knn = Pipeline([('tfidf', transformer),
24                 ('scaler', StandardScaler(with_mean=False)),
25                 ('clf', KNeighborsClassifier(n_neighbors=5,
26                                         metric='minkowski',
27                                         p=2))])
28
29 # Linear SVC:
30 lsvc = Pipeline([('tfidf', transformer),
31                 ('scaler', StandardScaler(with_mean=False)),
32                 ('clf', LinearSVC(class_weight='balanced',
33                                     random_state=0))])
34
35 # Random forest
36 rfc.fit(X_train, y_train)
37 y_pred_rf = rfc.predict(X_test)
38
39 # K - nearest neighbors
40 knn.fit(X_train, y_train)
41 y_pred_knn = knn.predict(X_test)
42
43 # Linear SVC
44 lsvc.fit(X_train, y_train)
45 y_pred_lsvc = lsvc.predict(X_test)
```

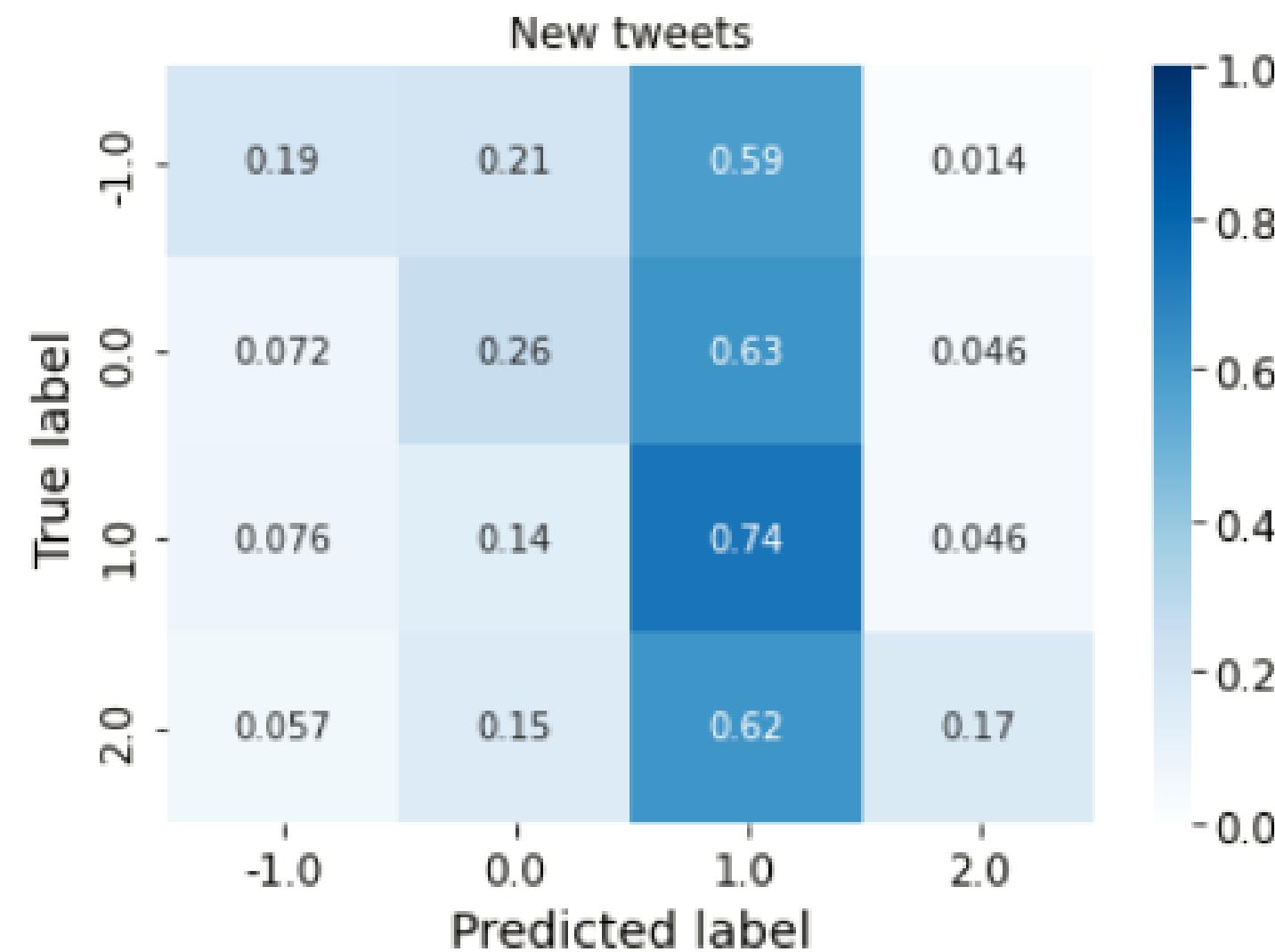
Create a pipeline to boost replication

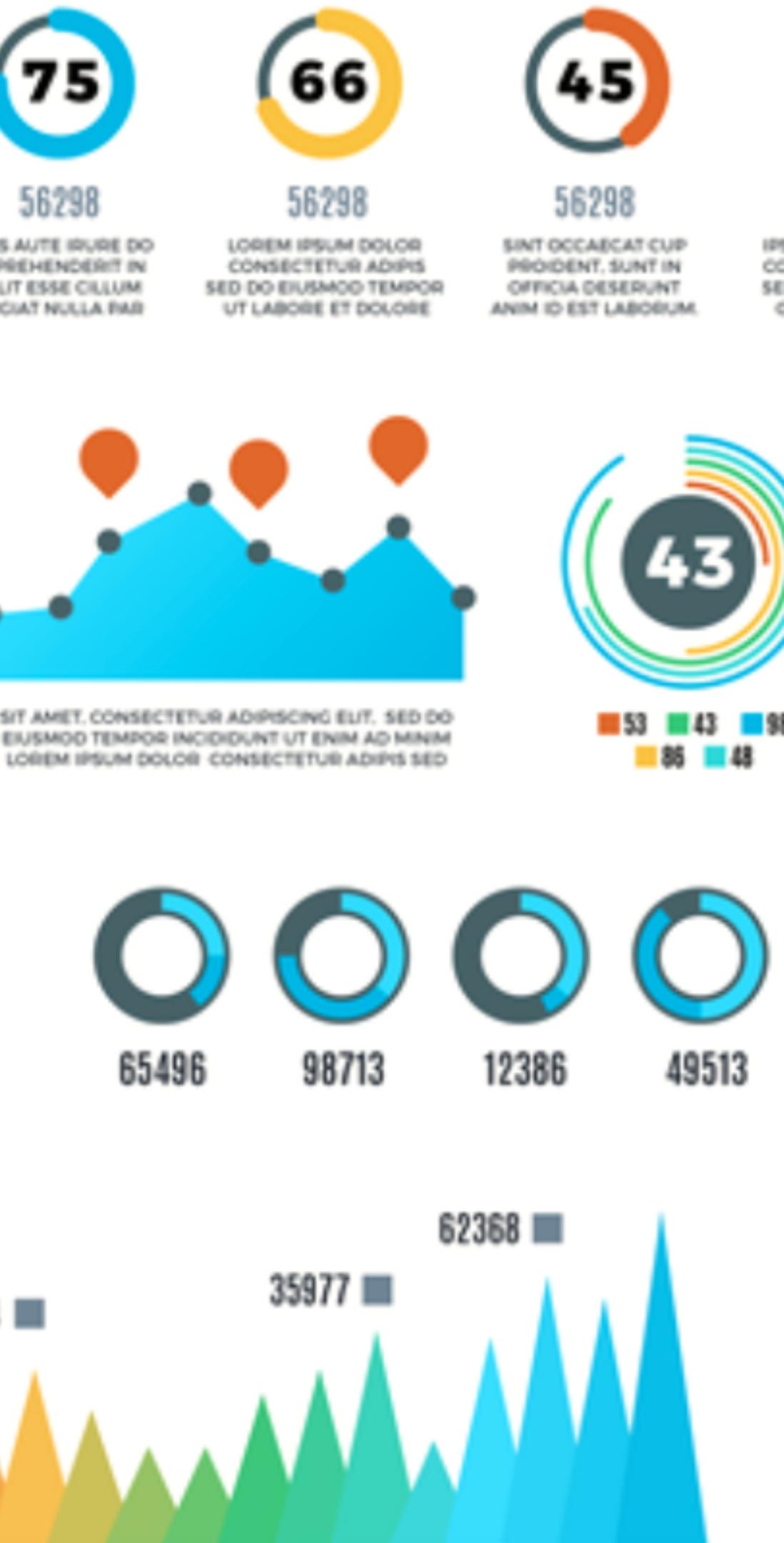
This code chunk is to create a pipeline to train models with TFIDF, StandardScaler and use default parameters. We create 3 models:

- Random Forest
- KNNs
- LinearSVC

Model Performance on New Tweets

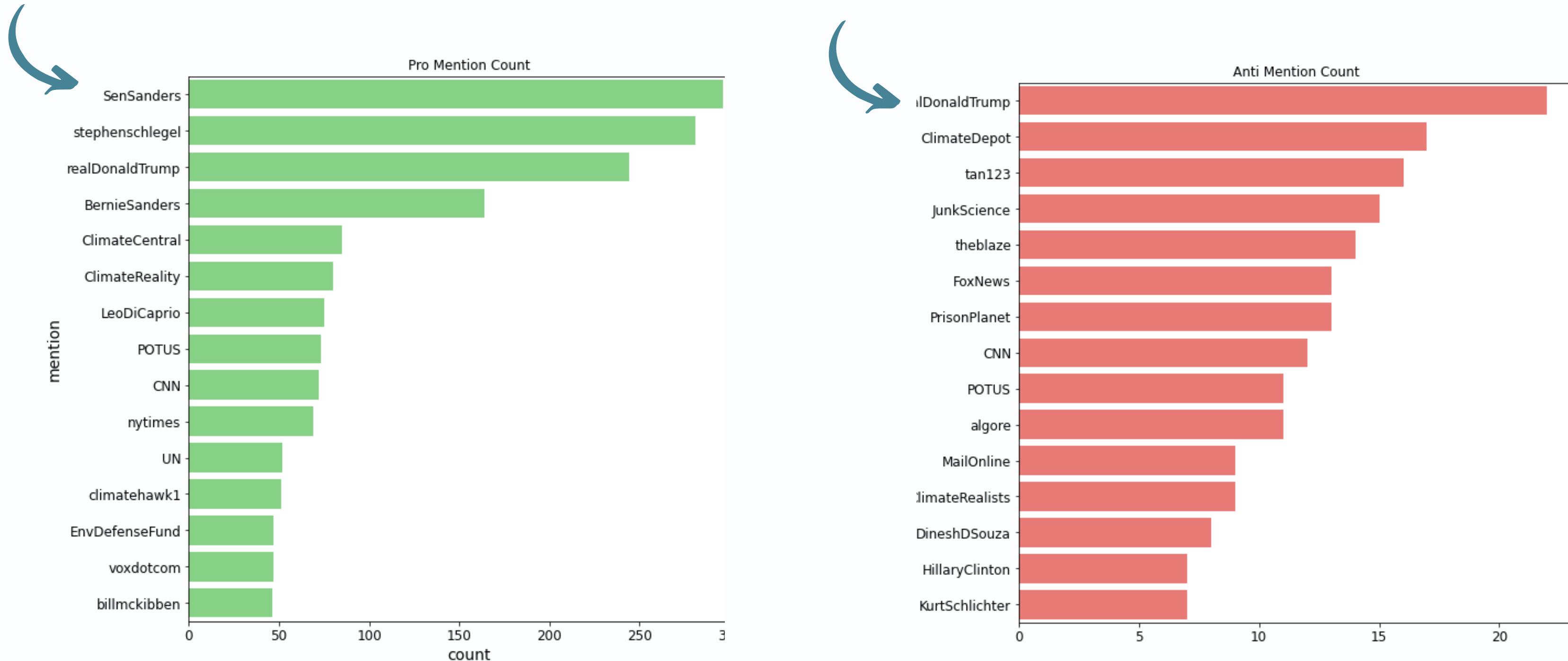
```
plot_confusion_matrix(best_model, "New tweets", new_tweets_pred, y_true=new_y)
```





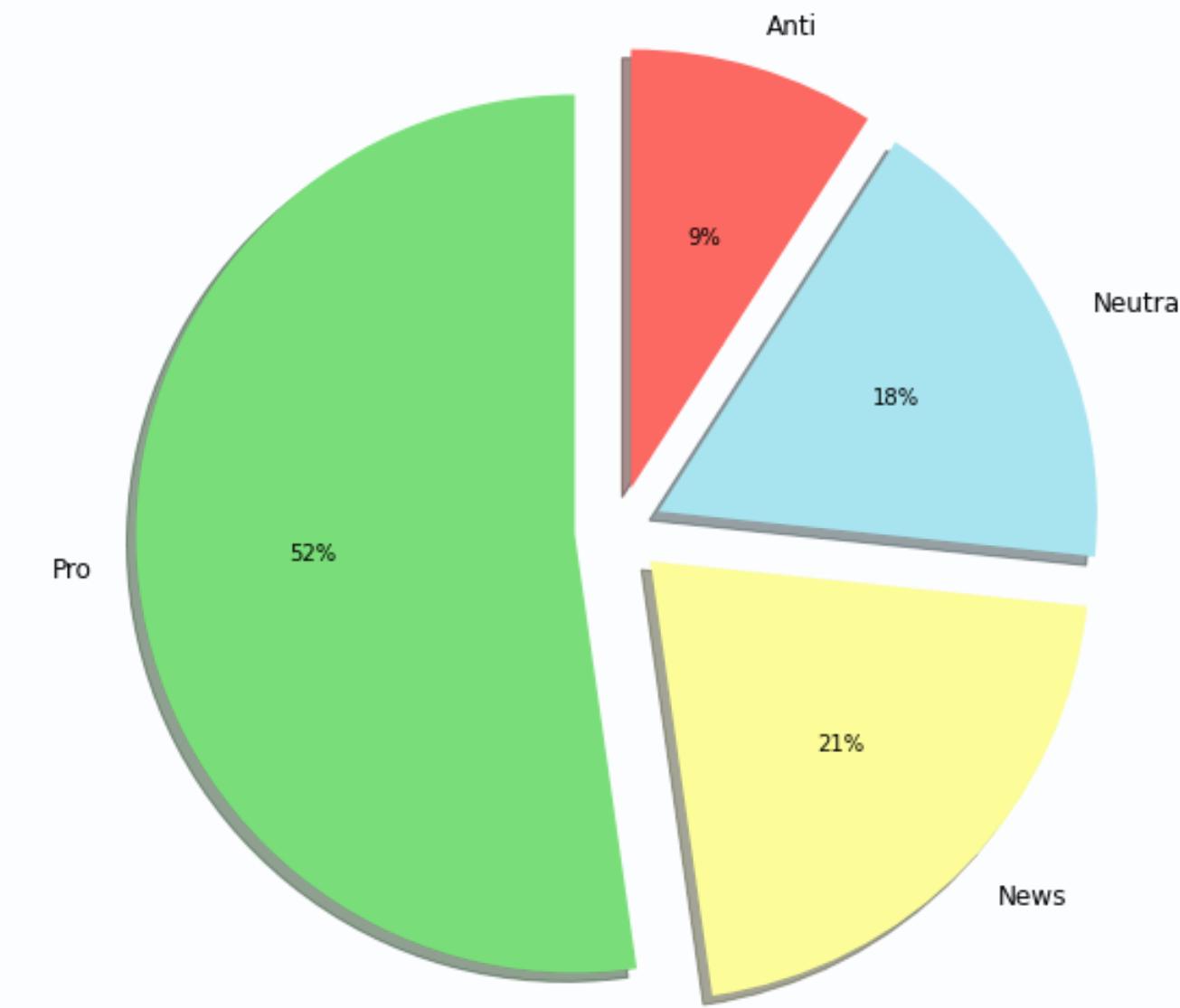
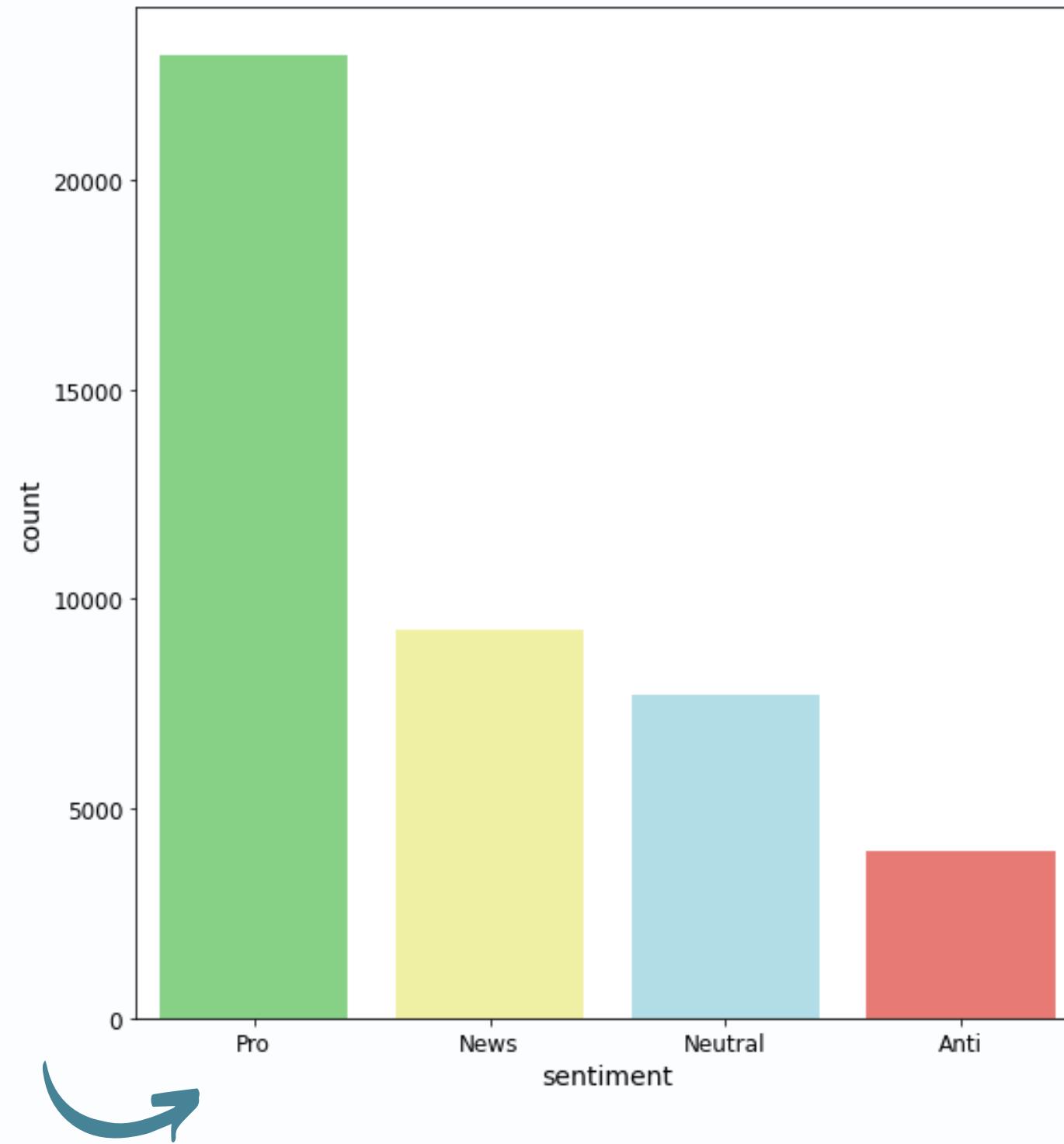
Visual Insights

Old tweets: mention count

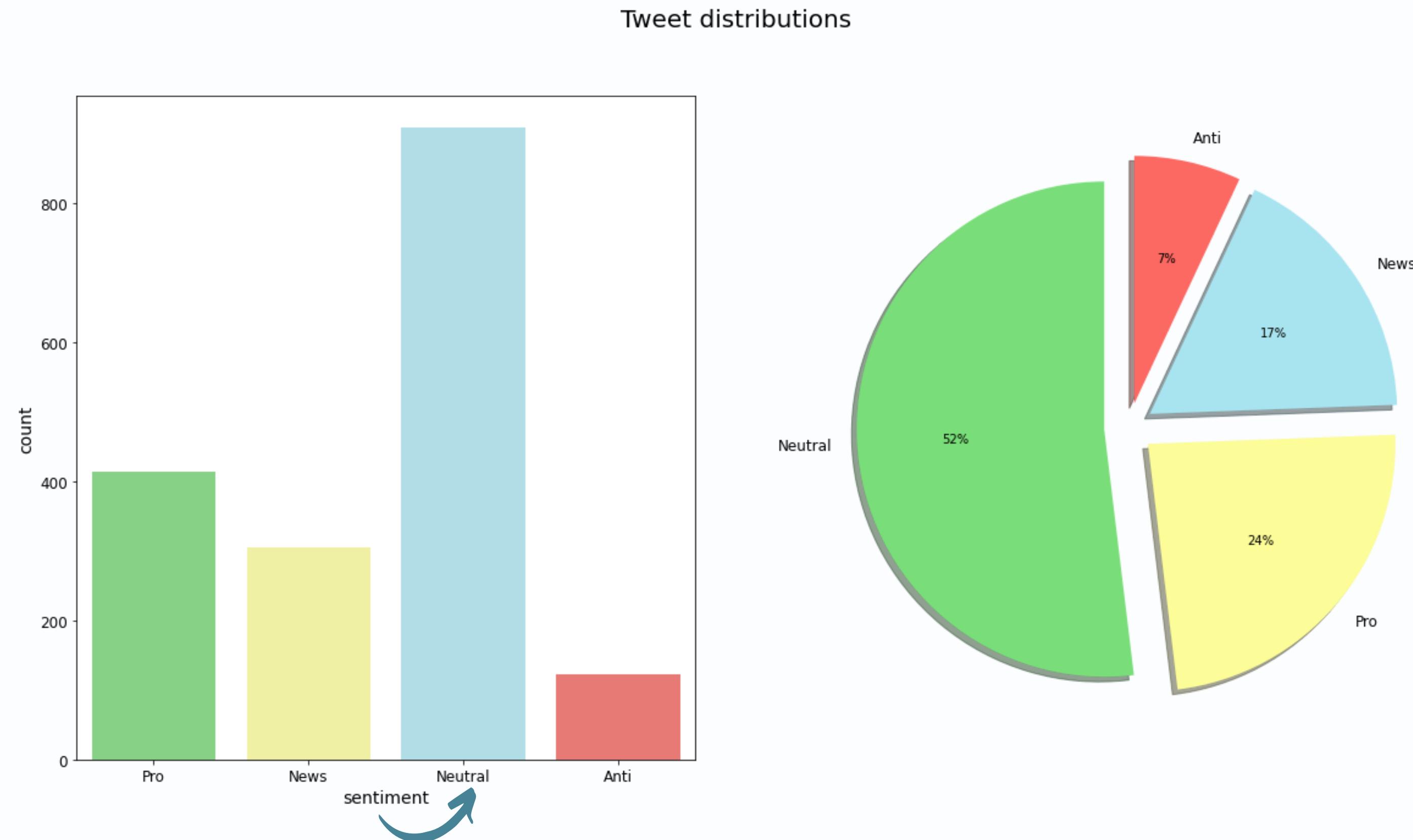


Old Tweets Distribution as per Category

Tweet distributions



New Tweets Distribution as per Category



Conclusion

Old tweets: Pro-climate change or news (2015-2018)

- Strong political influence of the label of the tweets with Donald Trump being mentioned many times in anti-climate tweets
- Bernie Sanders being mentioned many times in pro climate change tweets
- The Linear SVC model performed the best:
 - f1 score of 0.59
 - After tuning the hyperparameters, there was an increase in the performance by 2%

Linear SVC model performed the best with a f1 score of 0.59

New tweets: Neutral (February 2022)

- Model preformed poorly

Conclusion



Due to the vast nature of our dataset, it was difficult to reach a single conclusion for our questions.

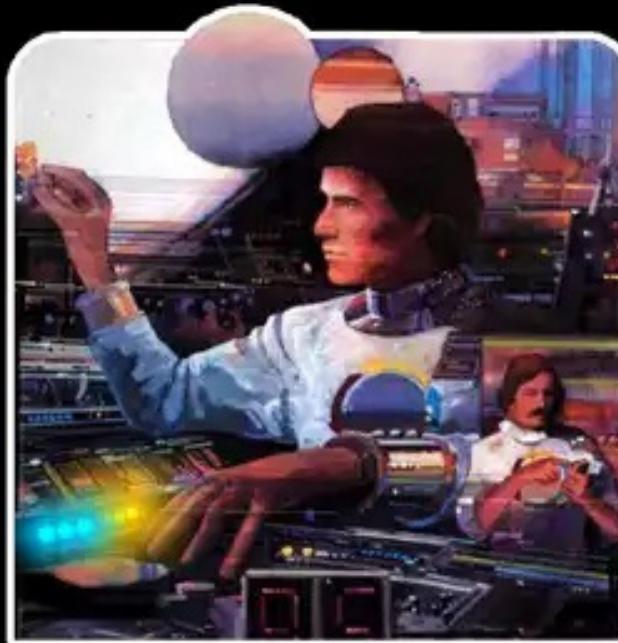
Based on the exploratory analysis there could have been a change in the general sentiment of climate change on Twitter moving towards being more neutral.

However, Neural Network still remains a promising model. It would be interesting to explore this further by gathering more data, doing more feature engineering, and training more models.

THANK YOU



THE TWO STATES OF EVERY PROGRAMMER



I AM A GOD.



I HAVE NO IDEA
WHAT I'M DOING.

QUESTIONS & ANSWERS
