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

em lyon business school

Data Processing

LABELLED DATA

LOAD	EXTRACT	UNNESTED	TRANSFORM	FILL NA	MERGE
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	
	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

```python
def get_metadata_for_tweets(df, id_col, chunk_n=100, tweet_fields=TWEET_FIELDS,
                            expansions=EXPANSIONS, user_fields=USER_FIELDS,
                            poll_fields=POLL_FIELDS, place_fields=PLACE_FIELDS,
                            media_fields=MEDIA_FIELDS):
    """
    Gets metadata from the twitter API for a dataframe containing at minimum a
    list of tweet ids. By default divides dataframe into chunks of 100 which
    is max query for Twitter API. Accepts tweet_field and tweet_expansions
    as parameters. See Twitter API for a list of accepted fields and expansions.
    https://developer.twitter.com/en/docs/twitter-api/fields
    https://developer.twitter.com/en/docs/twitter-api/expansions

    param: df Pandas dataframe containing at minimum tweetids
    param: id_col Column containing tweet ids
    param: chunk_n Number of rows per chunk to divide dataframe into.
           by default 100. This corresponds to the max accepted by Twitter API
    param: tweet_fields List of extra fields to get from Twitter API
    param: expansions List of expansions to get from Twitter API
    """
    chunks = [df[i:i+chunk_n].copy() for i in range(0,df.shape[0],chunk_n)]
    joined_dfs = []

    for chunk in chunks:
        # Join ids by "," since query needs to be a string of ids separated by ","
        ids = ",".join(chunk[id_col].astype("str").values)

        response = client.get_tweets(ids, tweet_fields=tweet_fields, expansions=expansions,
                                     media_fields=media_fields, user_fields=user_fields,
                                     poll_fields=poll_fields, place_fields=place_fields)
        # Expand Tweepy response object into dataframe for easier merging
        tweets = pd.DataFrame(response.data)
        joined_dfs.append(chunk.merge(tweets, on="id", how="left"))

    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.

```python
def transform_references_binary(df, col_prefix, id_col=None):
    cols = [col for col in df.columns if col_prefix in col]
    if not id_col:
        id_col = f"{col_prefix}_id"
    print(id_col, cols)
    binar_col = (df.copy()
        .loc[:, id_col]
        .fillna(0)
        .apply(lambda x: np.where(x == 0, 0, 1))
        .rename(col_prefix)
        )

    transformed_df = (pd.concat([df, binar_col], axis=1)
                    .drop(cols, axis=1)
                    )

    return  transformed_df

unlabelled= (unlabelled.loc[:,~unlabelled.columns.duplicated()]
                    .pipe(transform_references_binary, "referenced_tweets")
                    .pipe(transform_references_binary, "in_reply_to_user")
                    .pipe(transform_references_binary, "attachment_poll", "attachment_poll.id")
                    .pipe(transform_references_binary, "attachment_media", "attachment_media")
                    )

```

transform_reference_binary()?

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

em lyon business school

Clustering

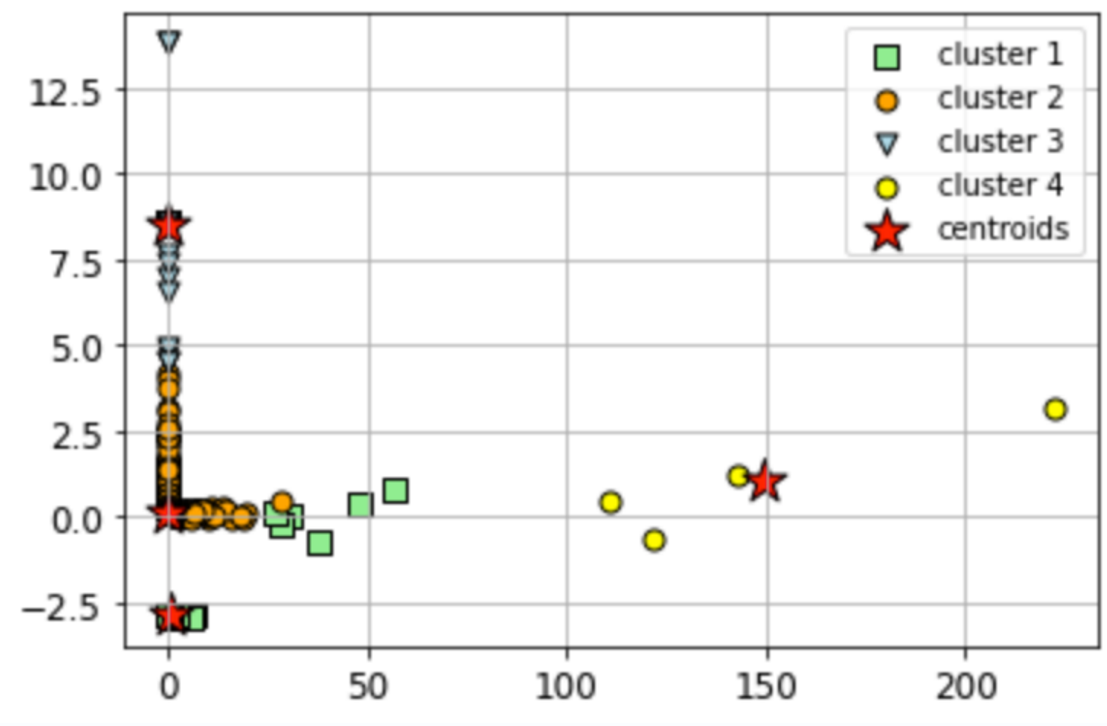
OUR GOAL:

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

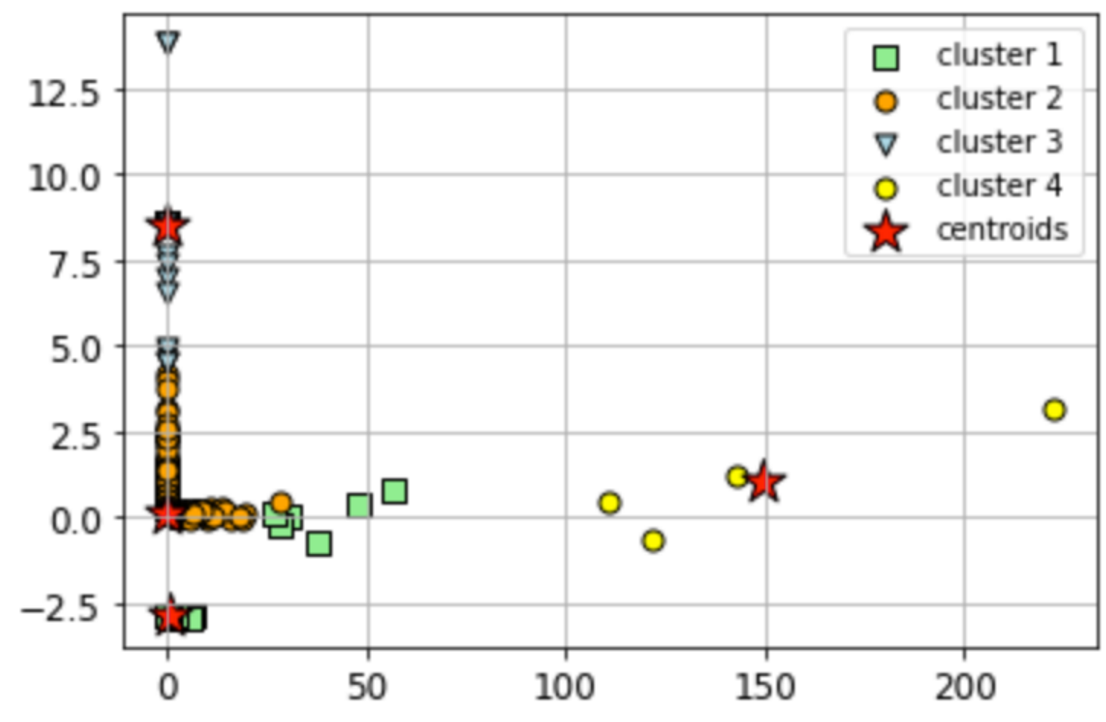
```python
from sklearn.cluster import KMeans

km_cluster = KMeans(
    n_clusters=4, init= 'random',
    n_init = 20, max_iter= 10_000,
    tol= 1e-04, random_state=0
)


y_km = km_cluster.fit_predict(labelled_X2D)

plt.scatter(
    labelled_X2D[y_km==0]['x'], labelled_X2D[y_km==0]['y'],
    s=50, c='lightgreen',
    marker='s', edgecolor='black',
    label='cluster 1'
)

plt.scatter(
    labelled_X2D[y_km == 1]['x'], labelled_X2D[y_km == 1]['y'],
    s=50, c='orange',
    marker='o', edgecolor='black',
    label='cluster 2'
)

plt.scatter(
    labelled_X2D[y_km == 2]['x'], labelled_X2D[y_km == 2]['y'],
    s=50, c='lightblue',
    marker='v', edgecolor='black',
    label='cluster 3'
)

plt.scatter(
    labelled_X2D[y_km == 3]['x'], labelled_X2D[y_km == 3]['y'],
    s=50, c='yellow',
    marker='o', edgecolor='black',
    label='cluster 4'
)

plt.scatter(
    km_cluster.cluster_centers_[:, 0], km_cluster.cluster_centers_[:, 1],
    s=250, marker="*",
    c='red', edgecolor='black',
    label= 'centroids'
)

plt.legend(scatterpoints=1)
plt.grid()
plt.show()

```

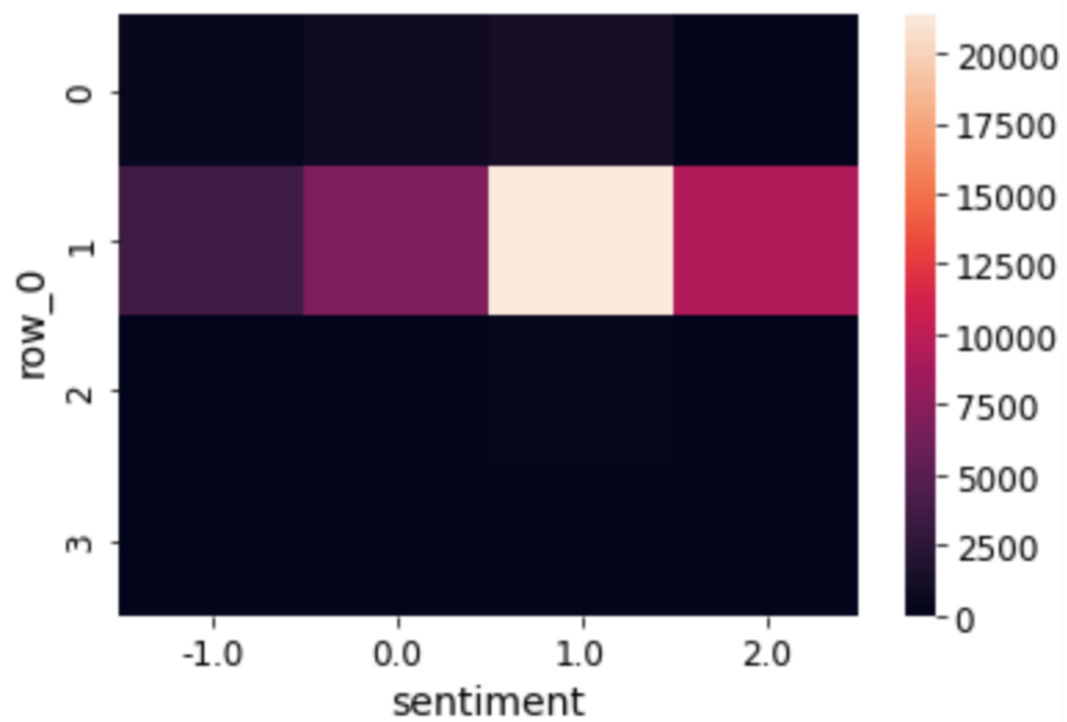
Clustering and plotting the cluster

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

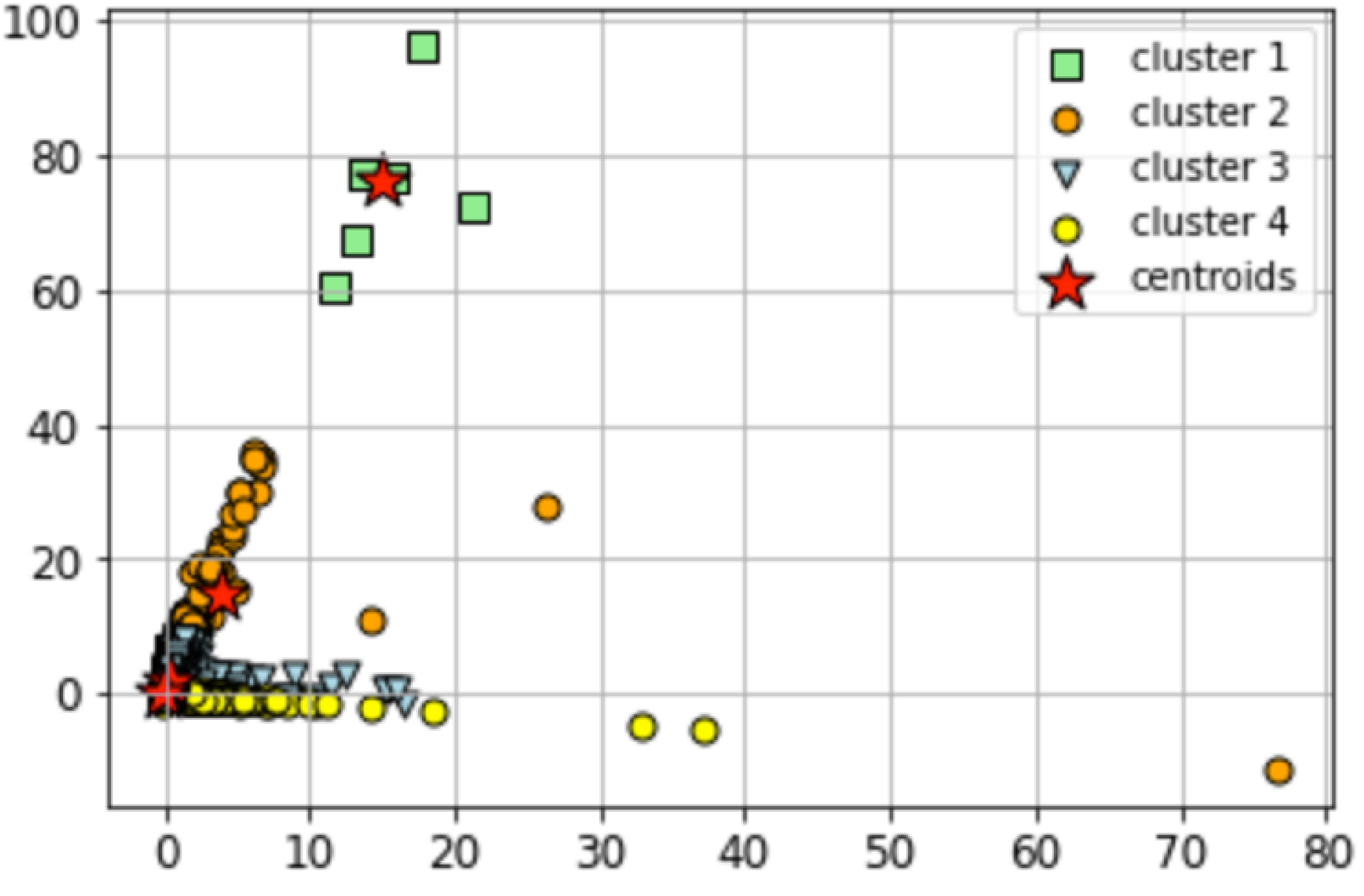


Labelled Dataset

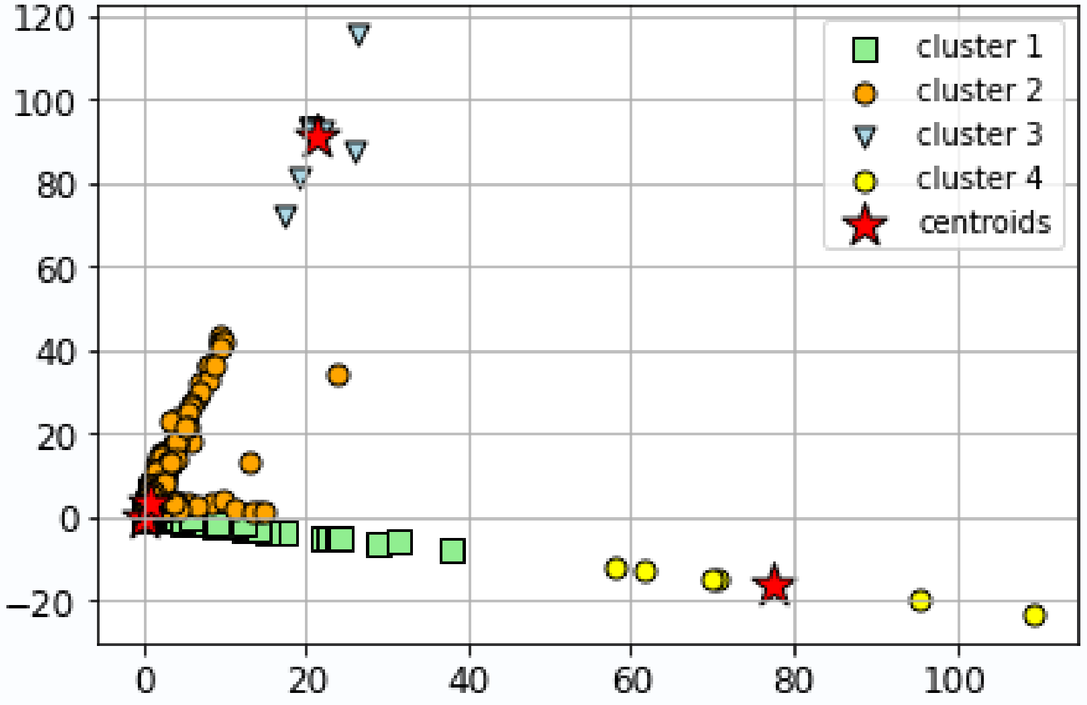


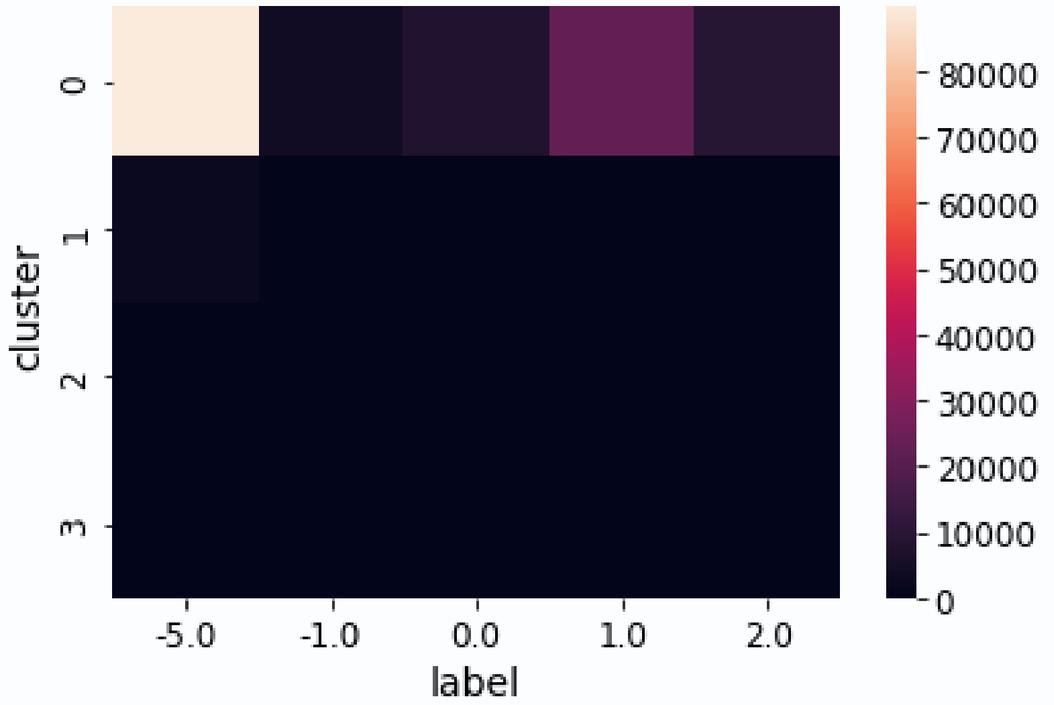


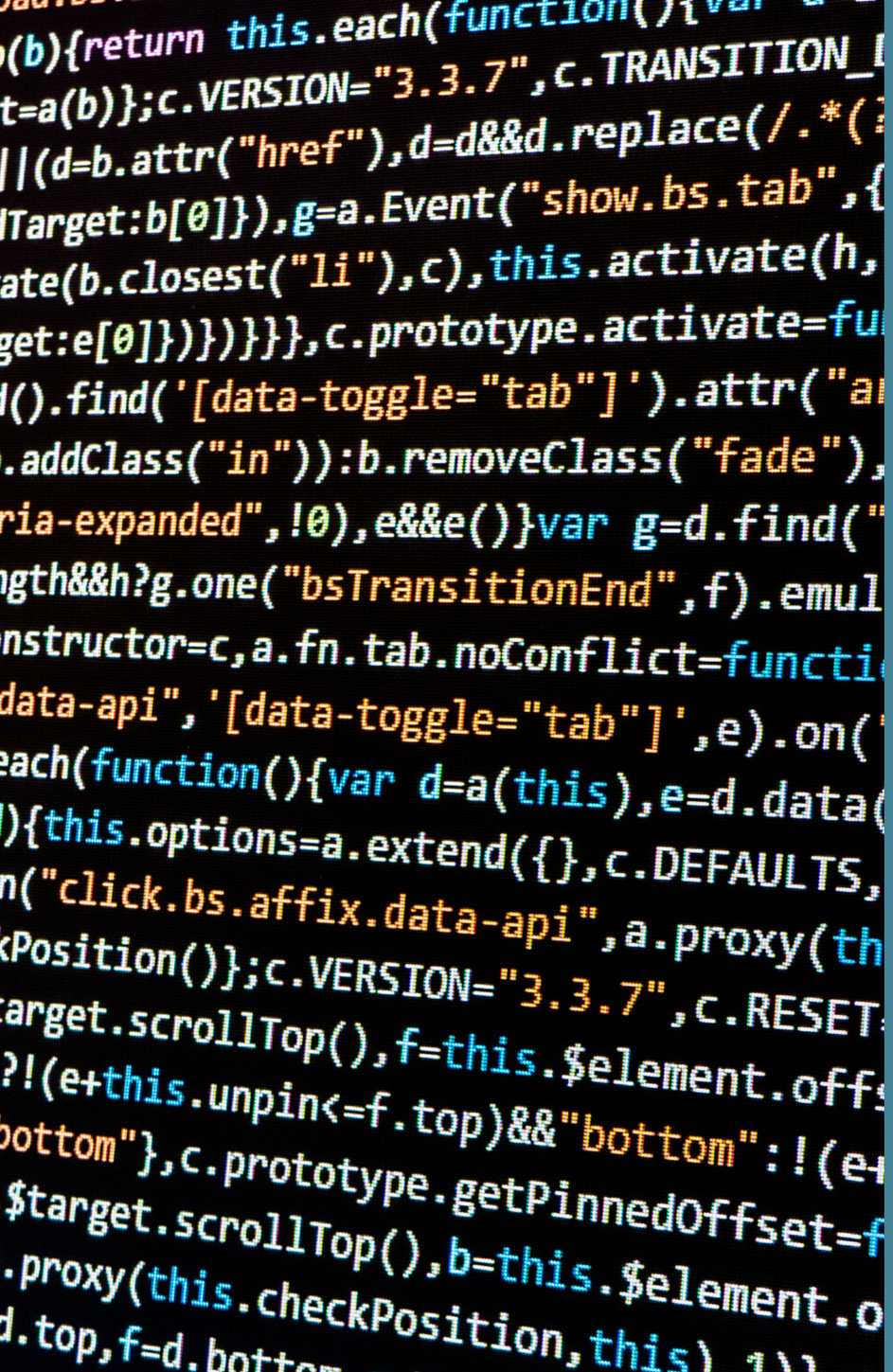
Unlabeled Dataset



Semi-Labelled Dataset







em lyon business school

Modelling

Model Selection

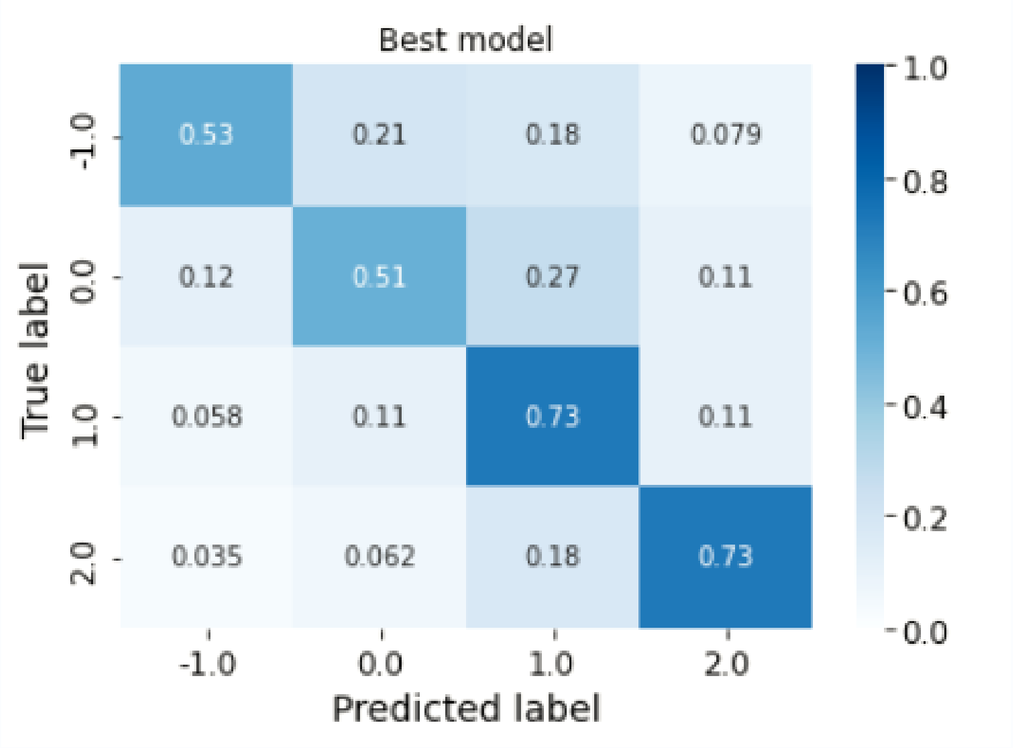
Models selected:

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

```
plot_confusion_matrix(best_model, "Best model", y_pred, y_true=y_test)
```



```python
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import balanced_accuracy_score, f1_score, precision_score, recall_score,
confusion_matrix

transformer = ColumnTransformer(
    [('vec', TfidfVectorizer(stop_words= 'english'), 'message')],   # column should be a string or
int
    remainder='passthrough'
)

# Random Forest Classifier
rfc = Pipeline([('tfidf', transformer),
                ('scaler', StandardScaler(with_mean=False)),
               ('clf', RandomForestClassifier(max_depth=5,
                                              n_estimators=100,
                                              random_state=0))])

# K-NN Classifier
knn = Pipeline([('tfidf', transformer),
                ('scaler', StandardScaler(with_mean=False)),
                ('clf', KNeighborsClassifier(n_neighbors=5,
                                             metric='minkowski',
                                             p=2))])

# Linear SVC:
lsvc = Pipeline([('tfidf', transformer),
                 ('scaler', StandardScaler(with_mean=False)),
                 ('clf', LinearSVC(class_weight='balanced',
                                   random_state=0))])

# Random forest
rfc.fit(X_train, y_train)
y_pred_rf = rfc.predict(X_test)

# K - nearest neighbors
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

# Linear SVC
lsvc.fit(X_train, y_train)
y_pred_lsvc = lsvc.predict(X_test)
```

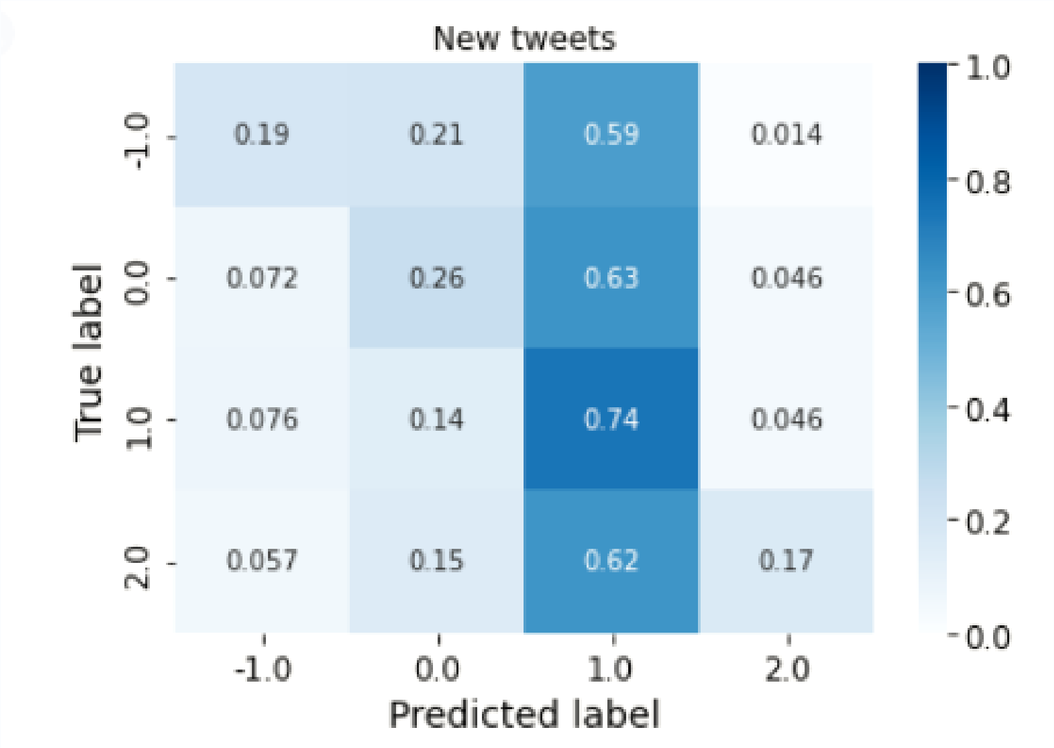
Create a pipeline to boost replication

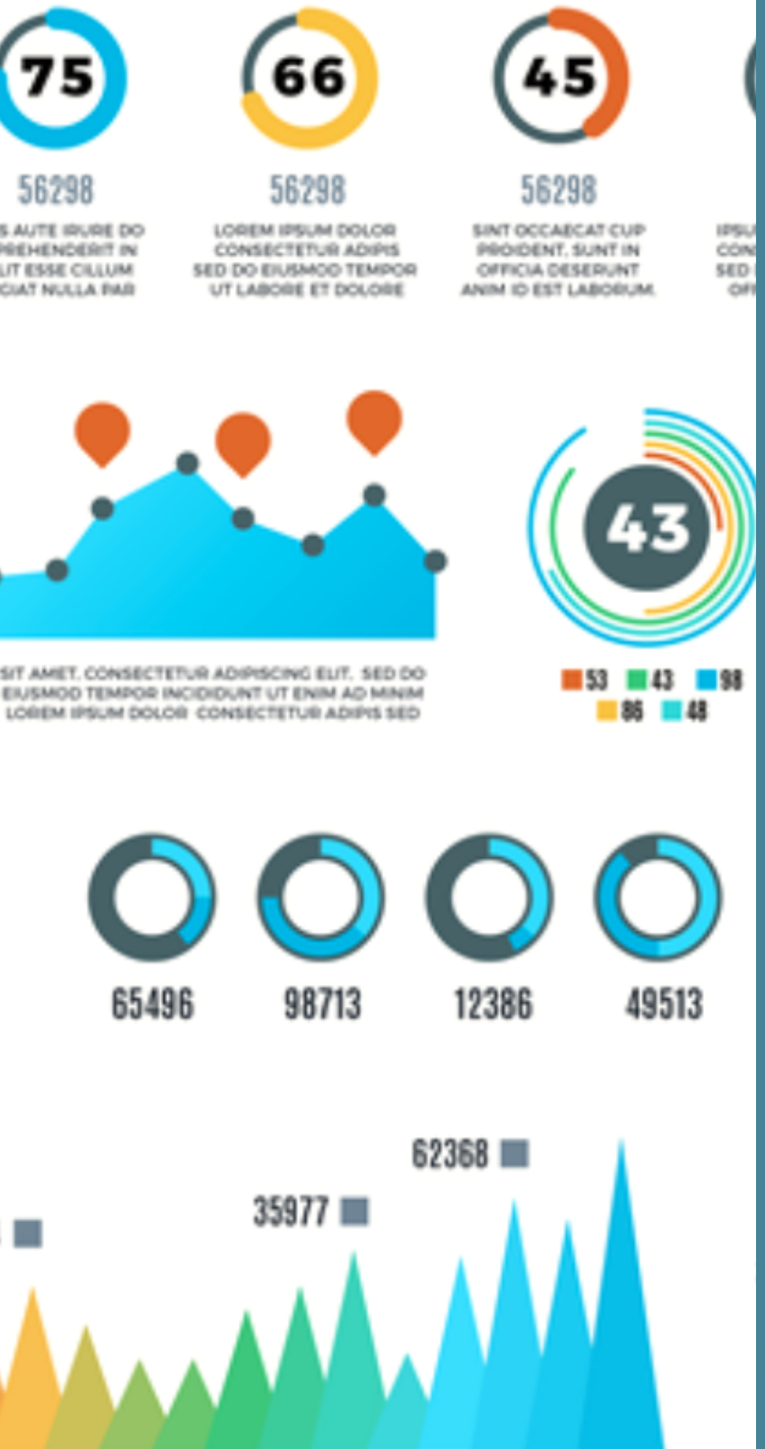
This code chunk is to create a pipeline to train models with TFIDF, StandardScaler and use default parameters. We create 3 models:

- Random Forest
- KNNs
- LinearSVC

Model Performance on New Tweets

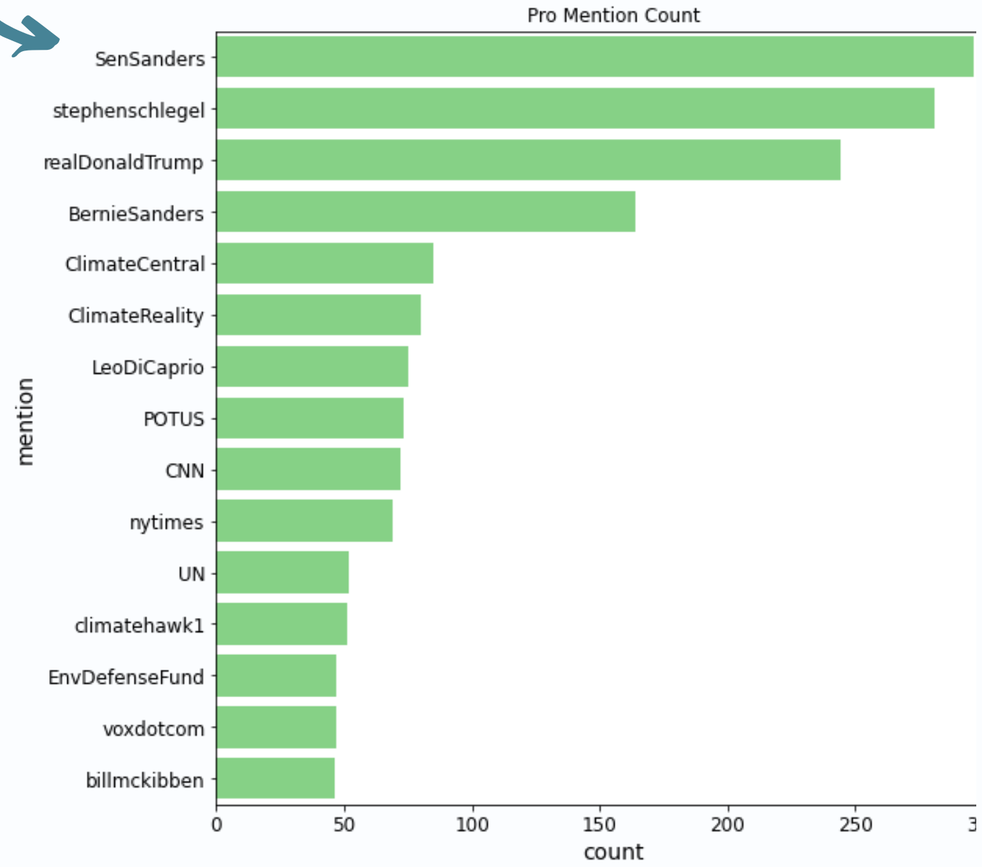
```
plot_confusion_matrix(best_model, "New tweets", new_tweets_pred, y_true=new_y)
```

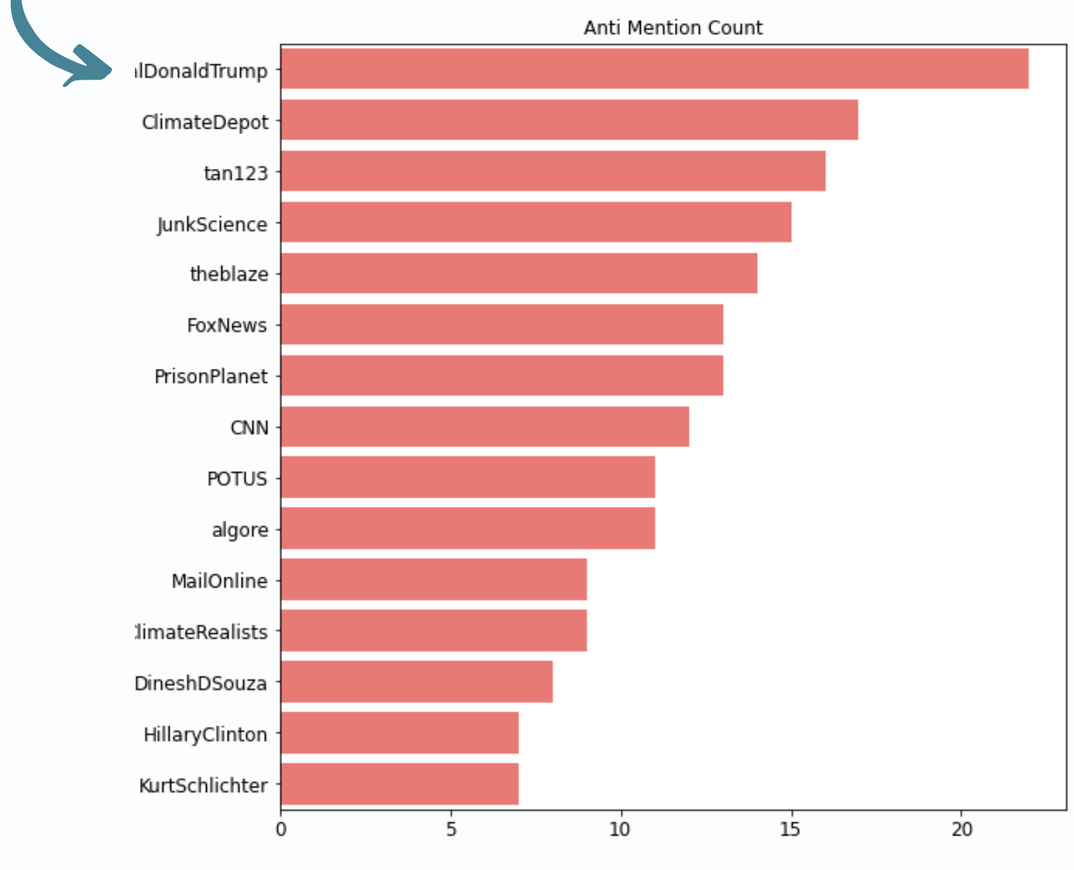




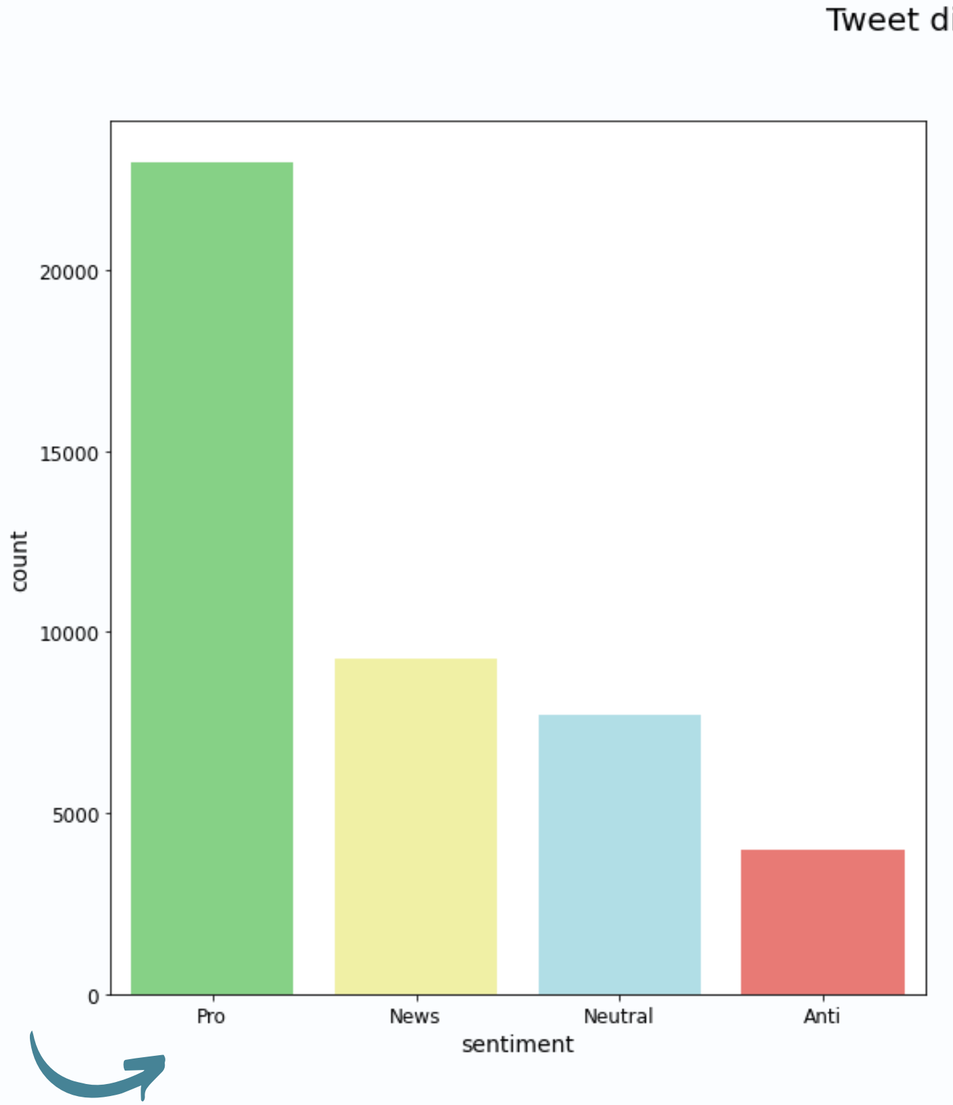
Visual Insights

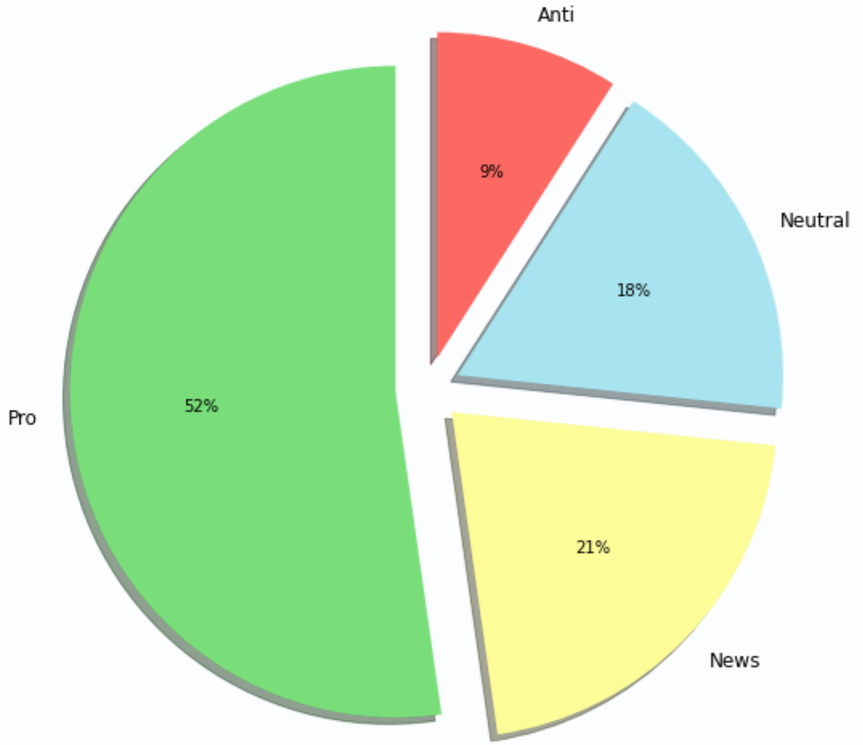
Old tweets: mention count



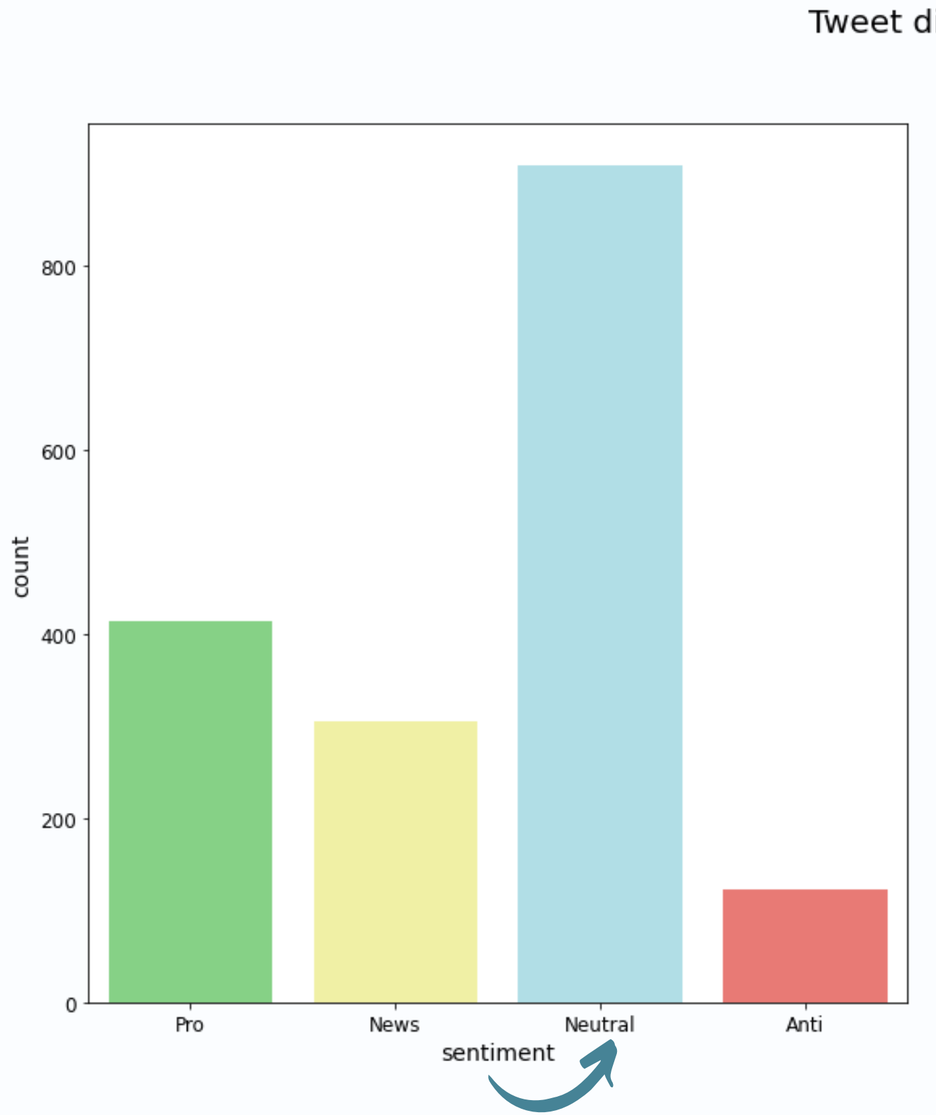


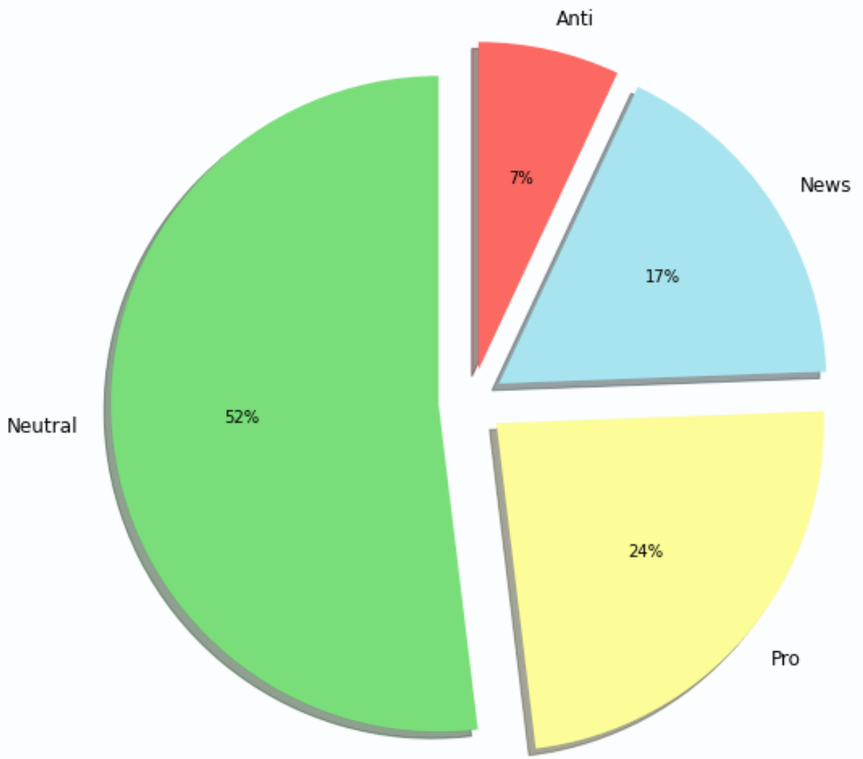
Old Tweets Distribution as per Category





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%

New tweets: Neutral (Feburary 2022)

- Model preformed poorly

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

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

em lyon business school

QUESTIONS & ANSWERS