Oeson
Inspiring generation

NLP FOR SENTIMENT ANALYSIS OF COVID TWITTER DATA

OESON INTERNSHIP

PALLAV CHAMOLI

CONTEXT

DATASET

A twitter dataset with the following fields is used for Sentiment Analysis.

1. UserName
2. ScreenName
3. Location
4. TweetAt
5. OriginalTweet
6. Sentiment

INTENT

1. Perform EDA for missing values on dataset
2. Analyse the dataset with plots
3. Pre-process the model for ML
4. Use ML models and compare perfromance
5. Use Deep learning for analysis
6. hypertune the model to increase model performance

WHAT ARE TOOLS USED?

For Analysis, cleaning and preparing the data for Machine Leanring and Deep leaning analysis, we used following libraries

Libraries : Numpy, Re, Pandas, Strings, Collections
Plots: Seasborn, WordCloud, Matplotlib
NLTK: WordNetLemmatizer, PorterStemmer
Sklearn: Linear SVC, logistic Regression, Train test split, tiff Vectoriser, confusion matrix
Keras: Sequential, Dense, Embedding, LSTM
Vader: Vader Sentiment, Sentiment Intensity Analyser

EDA: MISSING VALUE DETECTION

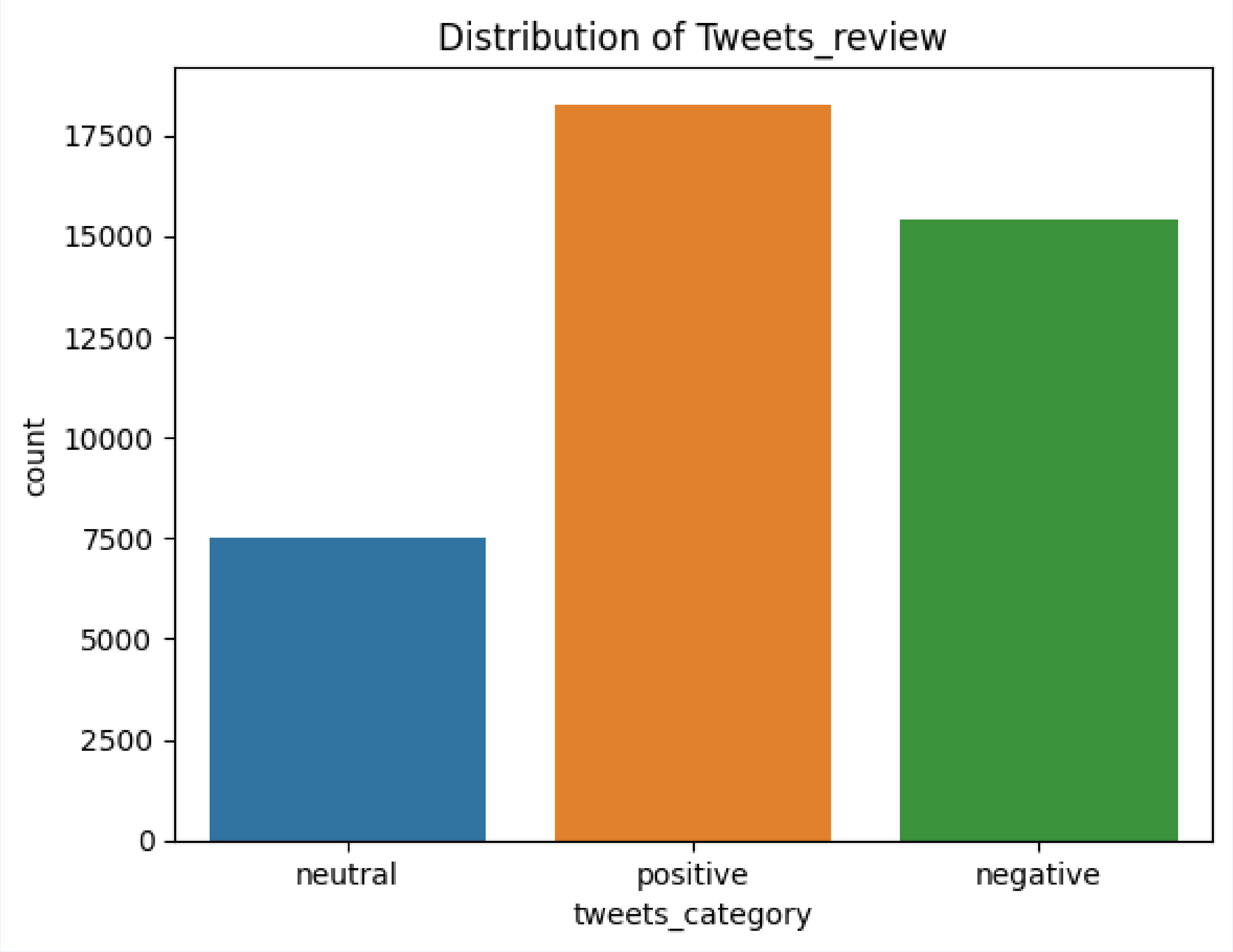
Q- what is exploratory data analysis & why is it important?

A- Exploratory data analysis refers to cleaning the data/extracting vital features before training model on the dataset. It is important because the missing values or exisiting outliers can affect the accuracy and performance of the model in predicting new data.

Finding missing value and substituting them with null /pre-determined values:

This was accomplished with : df.fillna(method=’fill’, axis=1)

This will fill all the null values row-wise and hence would replace ‘Location’ values with their corresponding values



PLOT

Distribution of Sentiment in Dataset

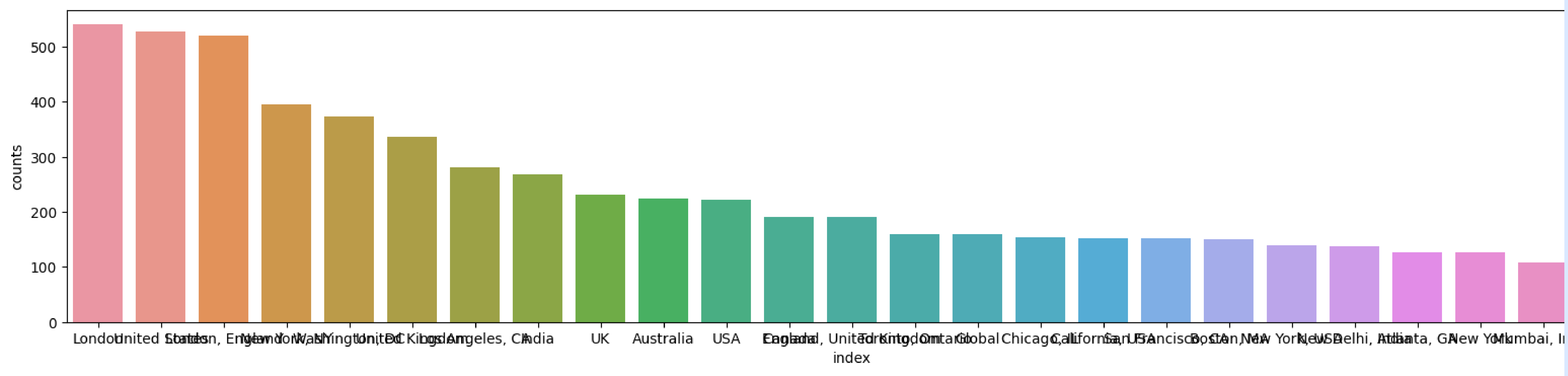
A histogram corresponding to each Sentiment value can be plotted using: sns.countplot(data=df, x='tweets_category').set_title ("Distribution of Tweets_review")

PLOT

Tweet Count by Location

A simple graph of the total number of tweets by Location can be done using:

```
fig, ax = plt.subplots(figsize=(20, 4))
sns.barplot(x='index', y='counts',
data=tweets_per_country, width=0.8)
```



PREPROCESSING FOR ANALYSIS

Analysis to plot common words and their word clouds

1. change labels of ‘ExtremelyPositive’&’ExtremelyNegative’ to ‘Positive’&’Negative’
2. Define stopwords list and clean the dataset
3. Remove punctuations, repeating characters, URL’s,Numbers,Special Characters
4. Cread Word Stemmer and Word Lemmatizer fuctions and apply them to dataset

Word Stemmer: A word Stemmer is natural language processing which reduces words to their bases or root form to improve computational efficiency and enable better analysis of text.

Word Lemmatizer: A word Lemmatizer is natural language processing which reduces words to their canoniocal or dictionary form to ensure better semantic analysis and maintain interpretability.

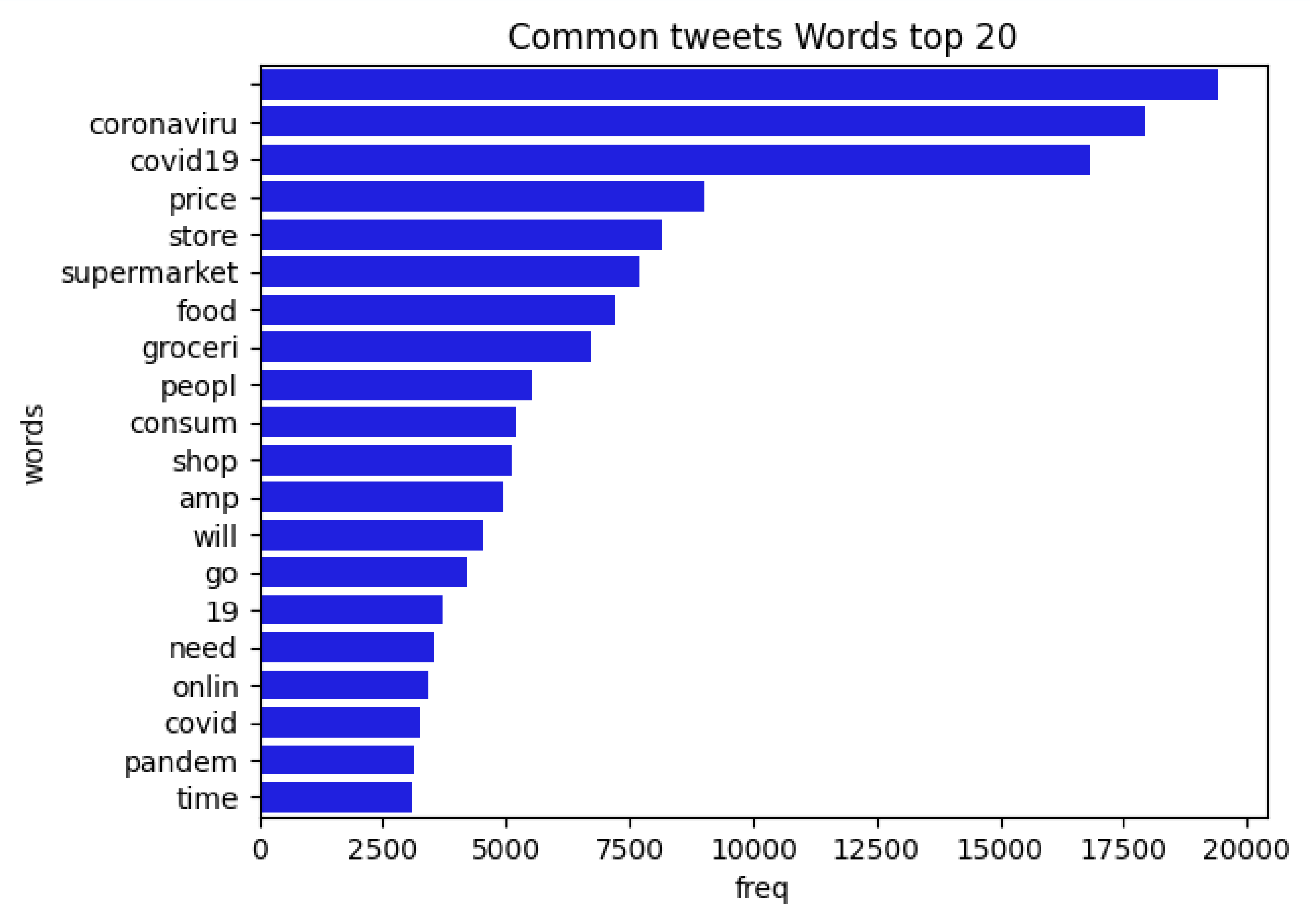
Let's explore different ways we can represent data!

PLOT

20 MOST COMMON WORDS USING VADER

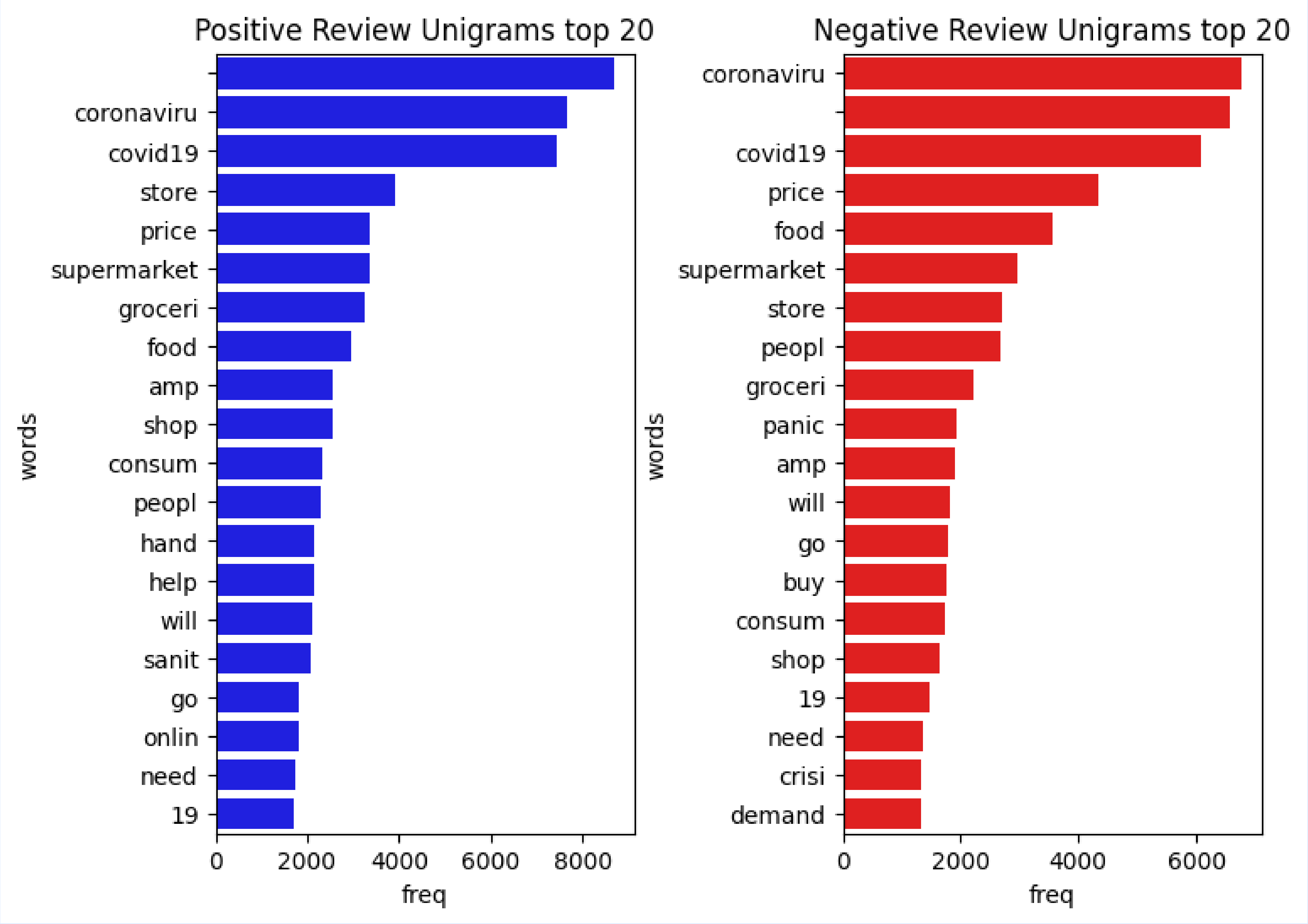
Most common words in dataset can be plotted using Vader

1. Using Porter Stemmer to clean the text and tokenize each review
2. Using Sentiment Intensity Anlyzer to calcuate polarity
3. Using Frequency Counter on Negative and Positive subsets based on polarity score



PLOT

comparision of Most common positive & Negative words in Dataset



Common words in negative tweets

Common words in positive tweets

WORD CLOUDS

word Clouds can be easily plotted using ‘WordCloud’ Library

```
wordcloud = WordCloud(height=800, width=1600,
background_color='black')
wordcloud = wordcloud.generate('
'.join(df.loc[df['tweets_category']=='positive','cleaned_
tweets'].tolist()))
plt.imshow(wordcloud)
```

```
wordcloud = WordCloud(height=800, width=1600,
background_color='black')
wordcloud = wordcloud.generate('
'.join(df.loc[df['tweets_category']=='negative','cleaned
_tweets'].tolist()))
plt.imshow(wordcloud)
```

Let's explore different Machine learning ways we can analysis data!

PREPROCESSING FOR MACHINE LEARNING

The Dataset has to be pre processed for training Machine Learning models using the following steps:

1. Split the dataset into training and test Datasets.
2. Apply TfidVectoriser on the 'X' values.

Tfidvectoriser: A tfidVectorizer natural language processing (NLP) converts a collection of raw text document into matrix of TF-IDF features, enabling text analyusis and information retrieval.

ML MODEL

Logistic Regression

	precision	recall	f1-score	support
Negative	0.80	0.81	0.81	3028
Neutral	0.67	0.67	0.67	1541
Positive	0.82	0.81	0.82	3663
accuracy			0.78	8232
macro avg	0.76	0.76	0.76	8232
weighted avg	0.79	0.78	0.78	8232

Training a Logistic Regression Model on the dataset has the accuracy of 78% on test dataset

ML MODEL

Support Vector Machine

	precision	recall	f1-score	support
Negative	0.78	0.79	0.79	3057
Neutral	0.65	0.68	0.66	1457
Positive	0.82	0.79	0.80	3718
accuracy			0.77	8232
macro avg	0.75	0.76	0.75	8232
weighted avg	0.77	0.77	0.77	8232

The SVC Model had the Second higest accuracy at 77% on the test dataset

ML MODEL

Random Forest

	precision	recall	f1-score	support
Negative	0.76	0.78	0.77	2988
Neutral	0.64	0.71	0.67	1402
Positive	0.82	0.77	0.79	3842
accuracy			0.76	8232
macro avg	0.74	0.75	0.75	8232
weighted avg	0.77	0.76	0.76	8232

Training a Random Forest Model on the dataset has the accuracy of 76% on test dataset

ML MODEL

Multinomial Naive Bayes

	precision	recall	f1-score	support
Negative	0.74	0.67	0.71	3406
Neutral	0.19	0.73	0.30	405
Positive	0.81	0.66	0.72	4421
accuracy			0.67	8232
macro avg	0.58	0.69	0.58	8232
weighted avg	0.75	0.67	0.70	8232

Training a Multinomial Naive Bayes Model on the dataset has the accuracy of 67% on test dataset

COMPARISION OF MODEL PERFROMANCE

The Dataset has to be pre processed for training Machine Learning models using the following steps:

1. Split the dataset into training and test Datasets.
2. Apply TfidVectoriser on the ‘X’ values.

	Model	Test accuracy
0	Logistic Regression	0.784864
1	Support vector Machine	0.772352
2	Random Forest	0.763362
3	Naive Bayes	0.667274

Let's explore Deep learning Model for better Performance

Q- will it helps to improve the perforamnce of the models?

DEEP LEARNING MODELS

In order to train Deep Learning Models on the given Datasets
All fuctions was calcualted using ‘SparseCategoricalCrossentropy’ with an ‘Adam’ optimiser.
All deep learning Models tested for the given dataset with Sequential layer > Embedding & various number of Dense and Bidirectional layers.

Hence the strcture for models are

Input > Sequential > Embedding > ANN/CNN/RNN/LSTM .... > Dense > Output

```
Test Loss: 0.5695905685424805
Test Accuracy: 0.7921525835990906
CNN
Test Loss: 0.7647427916526794
Test Accuracy: 0.7838921546936035
RNN
Test Loss: 1.1772931814193726
Test Accuracy: 0.3741496503353119
LSTM
Test Loss: 0.973524808883667
Test Accuracy: 0.5651117563247681
```

MODELS

Multinomial Naive Bayes

The deep learning models performed as follows:

1. ANN : 79% accuracy
2. CNN : 78% accuracy
3. RNN : 37% accuracy
4. LSTM : 56% accuracy

CONCLUSION

- The dataset only has missing values in the non-essential fields like ‘Location’ & ‘TweetAt’. these were fixed using EDA
- London was the most common tweet location, followed USA & then New York
- The Model performance of Machine learning models performed at 78% by Logistic Regression, 77% Support Vector, Random forest 76%, & Naive Bayes 66%
- We analysed different type of Deep learning models ANN, CNN, RNN, LSTM.
- the initial model demonstarated an accuracy around 76%
- however, hypertunning the model resulted in increament of ~2%
- this was achived by using more features & embedding high dense layers of LSTM as well as CNN layers.