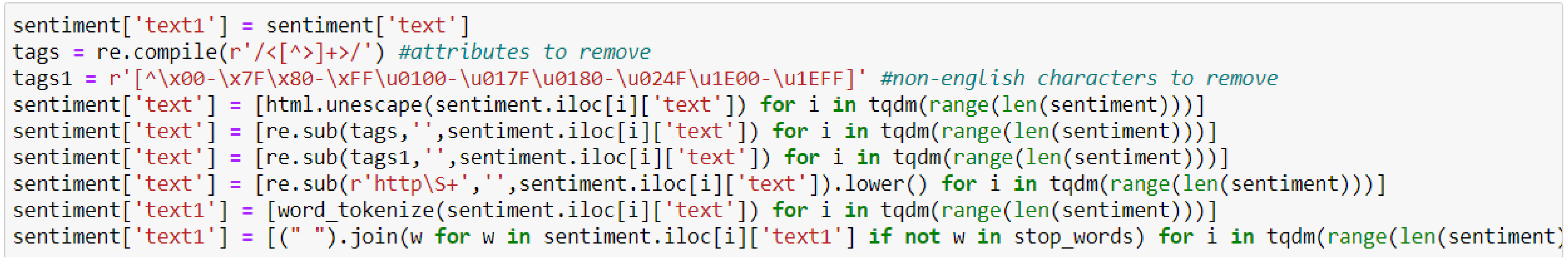
Sentiment analysis of twitter data

Topic introduction

- The exercise involves analyses of two datasets – one containing a set of generic tweets, and the other containing tweets specific to 2020 US elections
- The first objective is to train the generic tweets using different ML algorithms. This trained model will be used to predict sentiment of the election tweets.
- The second objective is to train the negative tweets of the elections dataset using different ML algorithms and then predict the reasons of the negative tweets.

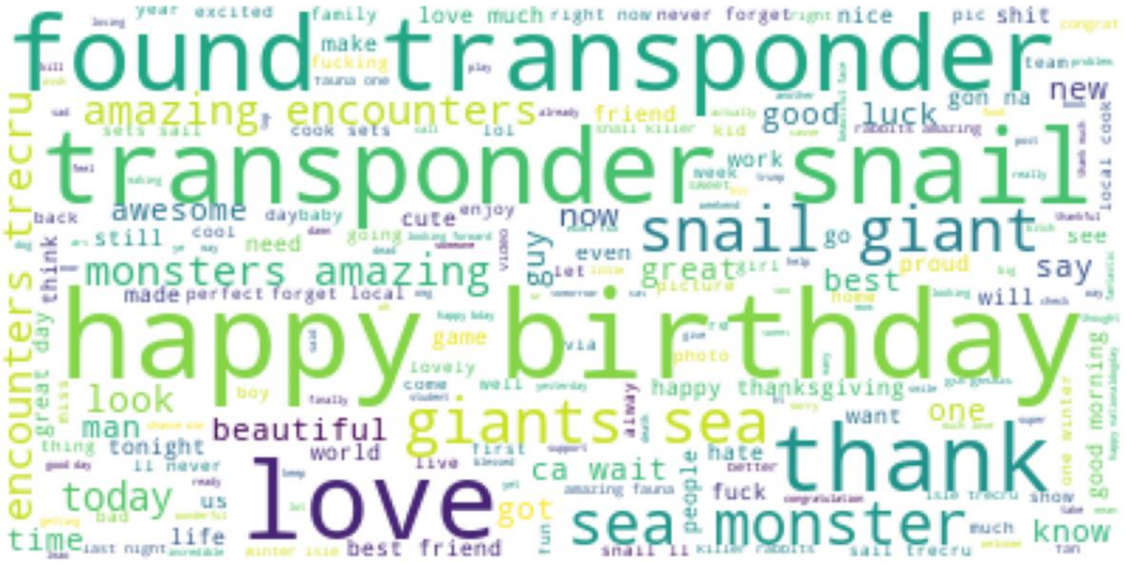
Data cleaning



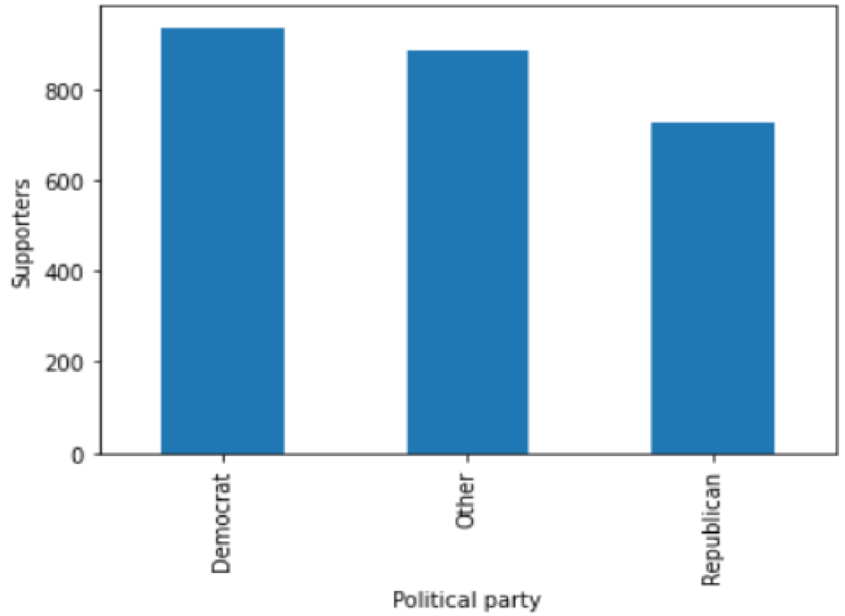
```python
sentiment['text1'] = sentiment['text']
tags = re.compile(r'/<[^>]+>/') #attributes to remove
tags1 = r'[^\x00-\x7F\x80-\xFF\u0100-\u017F\u0180-\u024F\u1E00-\u1EFF]' #non-english characters to remove
sentiment['text'] = [html.unescape(sentiment.iloc[i]['text']) for i in tqdm(range(len(sentiment)))]
sentiment['text'] = [re.sub(tags,'',sentiment.iloc[i]['text']) for i in tqdm(range(len(sentiment)))]
sentiment['text'] = [re.sub(tags1,'',sentiment.iloc[i]['text']) for i in tqdm(range(len(sentiment)))]
sentiment['text'] = [re.sub(r'http\S+','',sentiment.iloc[i]['text']).lower() for i in tqdm(range(len(sentiment)))]
sentiment['text1'] = [word_tokenize(sentiment.iloc[i]['text']) for i in tqdm(range(len(sentiment)))]
sentiment['text1'] = [(" ").join(w for w in sentiment.iloc[i]['text1'] if not w in stop_words) for i in tqdm(range(len(sentiment)
```

- First step is to clean the tweets of both the datasets to use it for further analysis.
- HTML tags & attributes, URLs and stop words are removed from the tweets. Also, HTML character codes are replaced with ASCII equivalents and all the characters are converted to lower case.

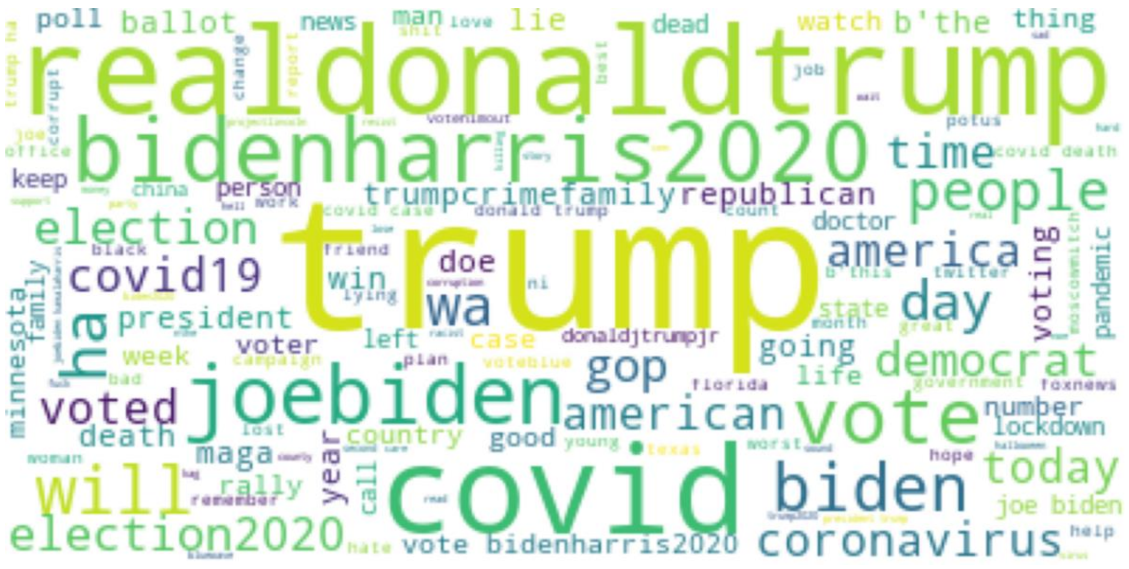
Exploratory analysis



Word Cloud of generic tweets



Distribution of political affiliation

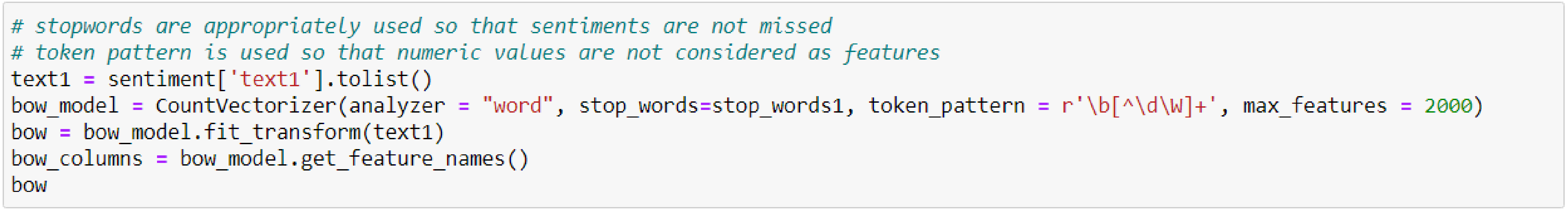


Word Cloud of election tweets

- The elections dataset shows that there are more supporters for the democratic party than the republican party. A large number of the tweeters are neither democrats/republicans.
- From the election word cloud we can see that words like 'trump', 'joebiden', 'covid', 'election2020', etc. are the most frequently used words. We can infer that it relates to the US 2020 presidential elections. Similarly, key words in the generic tweets can also be found in its word cloud.

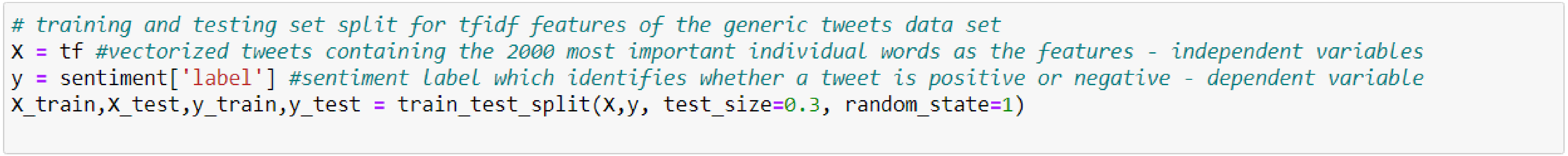
Model preparation

- The cleaned tweets from both datasets are vectorised into features using Bag of Words and Tfidf vectorization models
- Then number of features is limited to 2000 for both datasets



```
# stopwords are appropriately used so that sentiments are not missed
# token pattern is used so that numeric values are not considered as features
text1 = sentiment['text1'].tolist()
bow_model = CountVectorizer(analyzer = "word", stop_words=stop_words1, token_pattern = r'\b[^\d\W]+', max_features = 2000)
bow = bow_model.fit_transform(text1)
bow_columns = bow_model.get_feature_names()
bow
```

The vectorised BoW and Tfidf features are split into training and testing sets to apply various ML algorithms for prediction



```
# training and testing set split for tfidf features of the generic tweets data set
X = tf #vectorized tweets containing the 2000 most important individual words as the features - independent variables
y = sentiment['label'] #sentiment label which identifies whether a tweet is positive or negative - dependent variable
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.3, random_state=1)
```

Model implementation and tuning

ML Algorithm	Prediction accuracy	
	BoW features	**Tfidf Features**
Logistic regression	93.24%	93.28%
Decision tree	91.82%	91.84%
XG-Boost	90.02%	90.01%
Naïve Bayes	89.98%	90.77%
Random forest	93.06%	92.93%
kNN	84.52%	90.63%
Support Vector Machine	93.94%	93.68%
Evaluating the best performing model (SVM) on election dataset		
Support Vector Machine	62.81%	63.79%

- The BoW features and Tfidf features are used to train 7 different ML algorithms (as shown in the table), to predict the sentiment of the generic tweets
- SVM algorithm gave the best accuracy for both the feature sets
- Same BoW and Tfidf features are used to predict the elections dataset sentiment using the SVM algorithm
- For both the feature sets (BoW and Tfidf), the accuracies of prediction are moderate.
- This is because the features specific to elections are not used to predict the elections sentiment.
- This can be improved by vectorizing the elections dataset and training it using the ML algorithms, and then using it to predict the results. This could give better prediction accuracy.

Predicting negative reasons

First the election dataset is vectorised using tfidf model, then 3 ML algorithms are used to predict negative reason of the tweets. Then the hyperparameters are tuned for each algorithm.

ML Algorithm	Prediction Accuracy – Untuned model	Prediction Accuracy – Hyperparameter tuned	Best Hyperparameters
Logistic Regression	38.68%	38.68%	'C': 1, 'solver': 'newton-cg'
Decision Tree	32.63%	31.58%	'criterion': 'gini', 'max_depth': 8.0, 'max_features': 1.0, 'min_samples_leaf': 0.02
K-NN	27.11%	37.63%	'n_neighbors': 16, 'p': 1, 'weights': 'distance'

Example tweet and reason:

```
Original tweet : b'@brikeilarcnn @socalmom the truth is that the #covid19 deaths, the children separated from parents, dead &
Original sentiment : corruption
```

- All three models perform poorly in predicting the negative reasons
- This may be because the Tfidf features used to predict the negative reasons are not sufficient to do accurate prediction.
- The Tfidf features do not capture the real intent, quality, relevance and context of words used in the sentence.
- Another reason could be inconsistent and unreliable classification of the original reasons (as seen in the example tweet)
- Accuracy could be improved by manually coding the correct reasons on the training set and using it to predict the test set