# Sentiment Analysis Applied to Stock Prediction

Proposal: https://review.udacity.com/#!/reviews/1079040

## I. Definition

### Project Overview

Sentiment Analysis refers to the use of machine learning to identify the emotional reaction to an event, document or topic [1]. One of the possible applications of sentiment analysis is for predicting stock market movements. The internet is full of sources that represent the public opinion and sentiment about current events. Studies in [2] shows that the aggregate public mood can be correlated with Dow Jones Industrial Average Index (DJIA).

### Problem Statement

This capstone seeks a model which uses the top daily news headlines from Reddit (/r/worldnews) to predict stock market movement. A dataset with 8 years of daily news headlines and their respective DJIAs is available in Kaggle [3]. The stock market movement will be modeled into a binary classification problem, where:

- 1 is when DJIA Adj Close value **rose or stayed as the same**
- 0 is when DJIA Adj Close value **decreased**.

### Metrics

The evaluation metric to be used is Area Under the Curve (AUC) which is a metric derived from receiver operating characteristic (ROC) curve [4]. The most recent two years of the dataset (about 20%), from 2015-01-02 to 2016-07-01, is going to be reserved for testing.

## II. Analysis

### Data Exploration

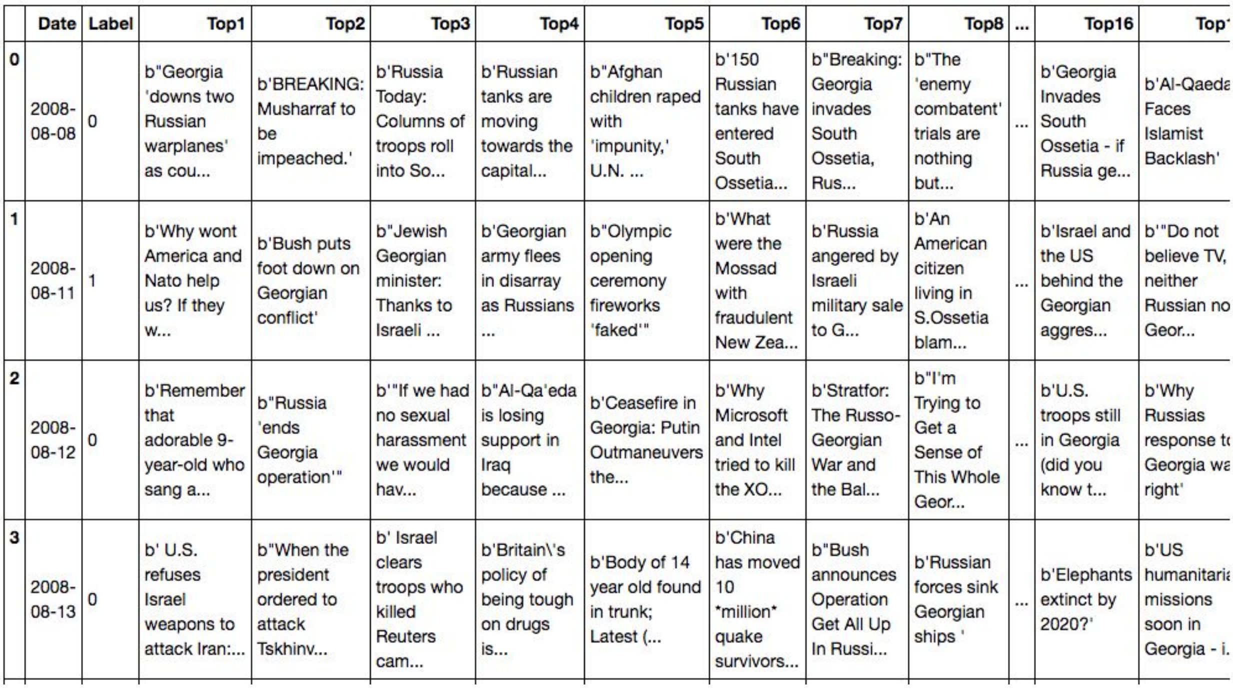
First some cursory investigation is computed:

- Total number of records: **1989**
- Number of records in which the index DJIA increased or stayed at the same: **1065**
- Number of records in which the index DJIA decreased: **924**
- Percent. of indexes which increased or stayed at the same: **53.54%**

Both classes are amost equally distributed (53% vs 47%) which is good since they don't suffer from imbalance. We can also check how the training and testing set are distributed:

- Class Balance in the Training set (first 6 years):
  - **1 -** 54%
  - **0 -** 46%
- Class Balance in the Test set (last 2 years):
  - **1** - 51%
  - **0** - 49%

We can observe that the balance of the test set is slightly off from the original balance, but still can be representative since it is not too far away from the original balance. He header of the dataset is given as follows:



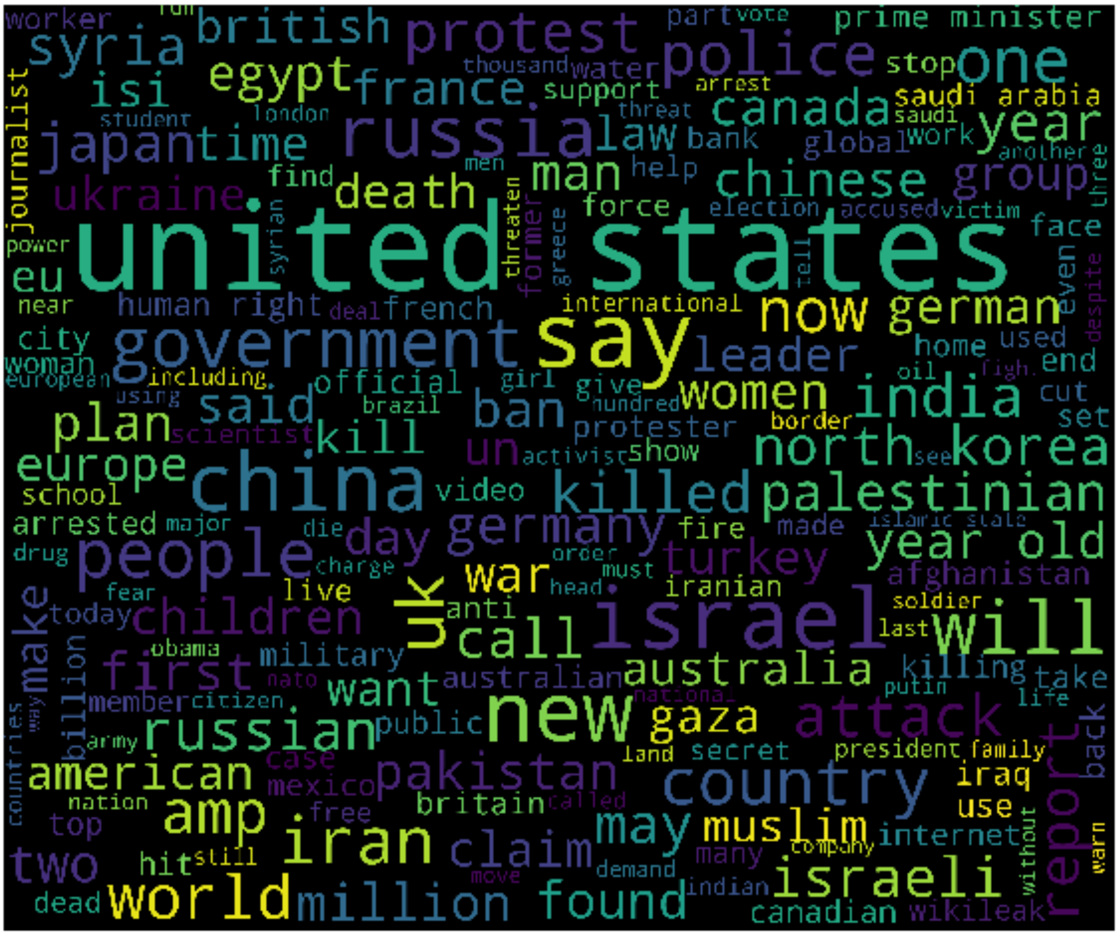
| | Date | Label | Top1 | Top2 | Top3 | Top4 | Top5 | Top6 | Top7 | Top8 | ... | Top16 | Top |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2008-08-08 | 0 | b"Georgia 'downs two Russian warplanes' as cou... | b'BREAKING: Musharraf to be impeached.' | b'Russia Today: Columns of troops roll into So... | b'Russian tanks are moving towards the capital... | b"Afghan children raped with 'impunity,' U.N. ... | b'150 Russian tanks have entered South Ossetia... | b"Breaking: Georgia invades South Ossetia, Rus... | b"The 'enemy combatent' trials are nothing but... | ... | b'Georgia Invades South Ossetia - if Russia ge... | b'Al-Qaeda Faces Islamist Backlash' |
| 1 | 2008-08-11 | 1 | b'Why wont America and Nato help us? If they w... | b'Bush puts foot down on Georgian conflict' | b"Jewish Georgian minister: Thanks to Israeli ... | b'Georgian army flees in disarray as Russians ... | b"Olympic opening ceremony fireworks 'faked'" | b'What were the Mossad with fraudulent New Zea... | b'Russia angered by Israeli military sale to G... | b'An American citizen living in S.Ossetia blam... | ... | b'Israel and the US behind the Georgian aggres... | b'"Do not believe TV, neither Russian no Geor... |
| 2 | 2008-08-12 | 0 | b'Remember that adorable 9-year-old who sang a... | b"Russia 'ends Georgia operation'" | b'"If we had no sexual harassment we would hav... | b"Al-Qa'eda is losing support in Iraq because ... | b'Ceasefire in Georgia: Putin Outmaneuvers the... | b'Why Microsoft and Intel tried to kill the XO... | b'Stratfor: The Russo-Georgian War and the Bal... | b"I'm Trying to Get a Sense of This Whole Geor... | ... | b'U.S. troops still in Georgia (did you know t... | b'Why Russias response t Georgia wa right' |
| 3 | 2008-08-13 | 0 | b' U.S. refuses Israel weapons to attack Iran:... | b"When the president ordered to attack Tskhinv... | b' Israel clears troops who killed Reuters cam... | b'Britain\\'s policy of being tough on drugs is... | b'Body of 14 year old found in trunk; Latest (... | b'China has moved 10 *million* quake survivors... | b"Bush announces Operation Get All Up In Russi... | b'Russian forces sink Georgian ships ' | ... | b'Elephants extinct by 2020?' | b'US humanitaria missions soon in Georgia - i. |

**Top1** to **Top25** are the **features** and refers to the top 25 headlines on Reddit News for each day. **Label** is the **target variable**. We can observe that all the headings starts with a \`b\` tag. This tag and other non-word characters are removed with the function \`clean_text\`:

```
def clean_text(text):
    text = re.sub("'b", '', text) # Remove the 'b tag
    text = re.sub('"b', '', text) # Remove the "b tag
    text = re.sub("\n", '', text) # Remove the \n tag
    text = re.sub('[\W]+', ' ', text.lower()) # Remove all non-words
    return text
```

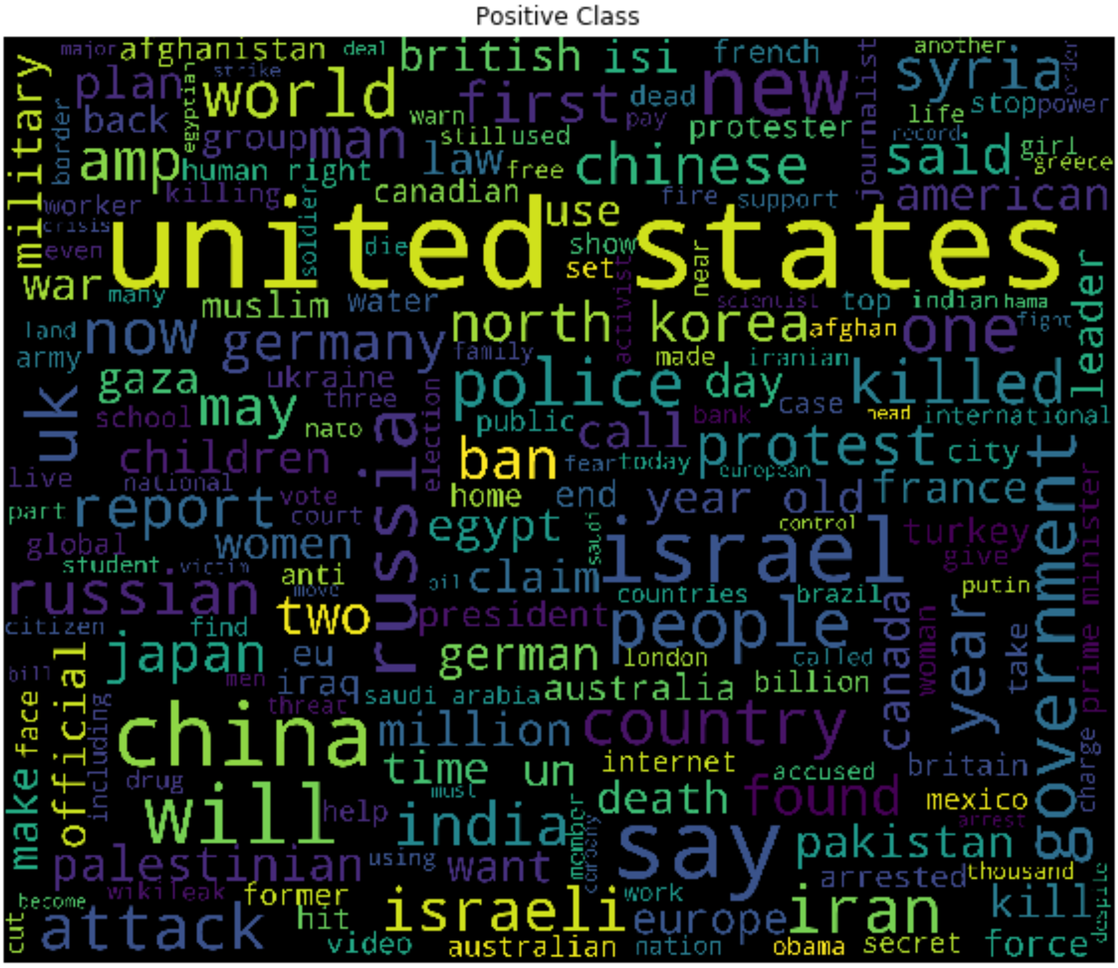
## Exploratory Visualization

Let’s perform the exploratory visualization using cloud of words. First, let’s check from all the headline’s content in the dataset:

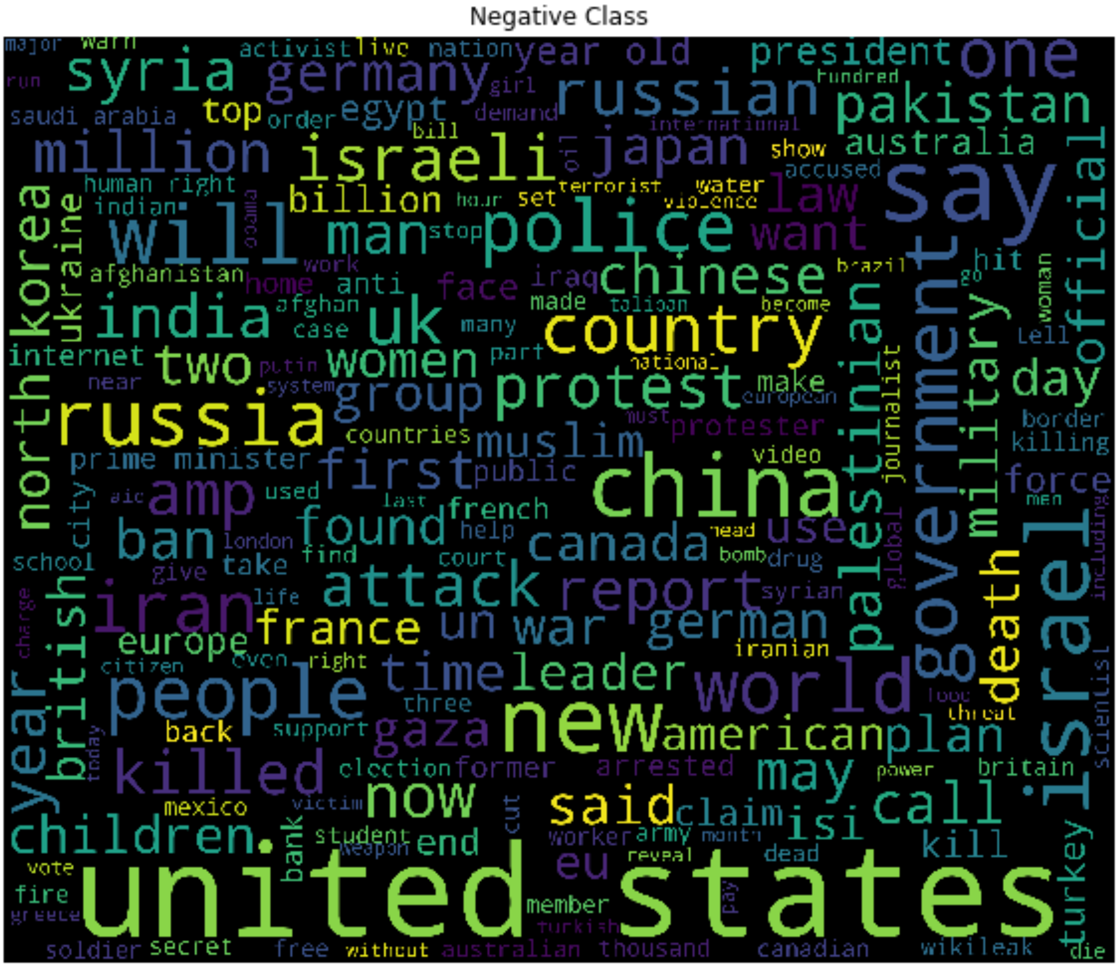


The package WordCloud was used for this visualization. Stop words were automatically removed from the visualization. We can observe that United States is the most frequent keywords in the news headline which is good since the DJIA is an American index. We can also check if there’s a difference between the cloud of words for the positive and negative class:

## Cloud of words for the positive class (Label = 1)



### Cloud of words for the negative class (Label = 0)



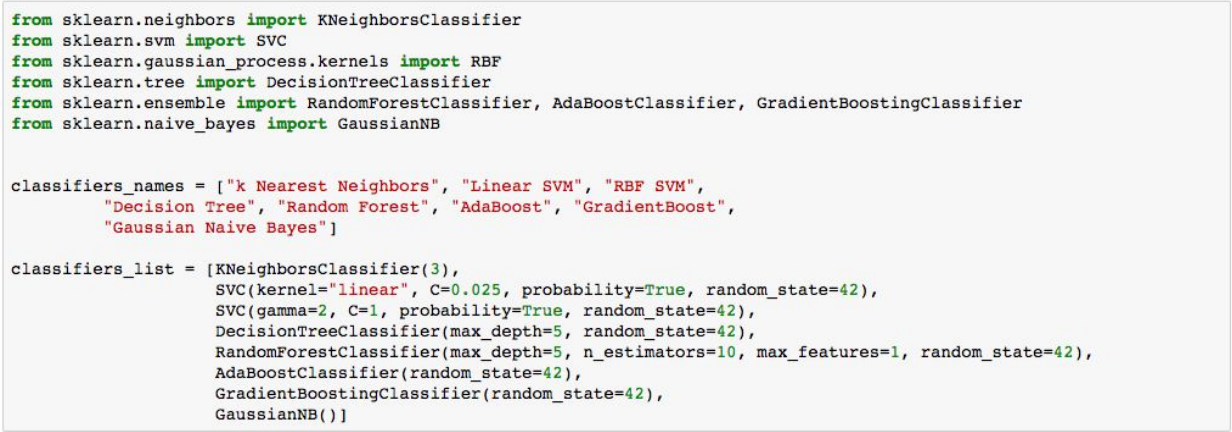
Negative Class

One of the words that is found in the negative class but not in the positive class is **attack** which might have a correlation.

## Algorithms and Techniques

First, the text data from the 25 features was cleaned (some HTML tags are still present in the original data). Next, the text was grouped and processed into feature vectors. The method **bag of words** [5] was used to represent the text as numerical feature vectors. The bag of words model created a vocabulary of **tokens** from the headlines data and then counted. Also, the relevancy of words was evaluated using the method **term frequency-inverse document frequency (tf-idf)** [6].

The learning algorithm Logistic Regression is initially employed since it fundamentally has probabilities as output (which is useful for the AUC metric). Later, most of the other algorithms from sklearn are also evaluated. Here’s the list of classifiers implemented:



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB


classifiers_names = ["k Nearest Neighbors", "Linear SVM", "RBF SVM",
         "Decision Tree", "Random Forest", "AdaBoost", "GradientBoost",
         "Gaussian Naive Bayes"]

classifiers_list = [KNeighborsClassifier(3),
                    SVC(kernel="linear", C=0.025, probability=True, random_state=42),
                    SVC(gamma=2, C=1, probability=True, random_state=42),
                    DecisionTreeClassifier(max_depth=5, random_state=42),
                    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1, random_state=42),
                    AdaBoostClassifier(random_state=42),
                    GradientBoostingClassifier(random_state=42),
                    GaussianNB()]
```

## Benchmark

Since this dataset is from a Kaggle kernel, there is no ‘official’ benchmark available. Below are the scores from some Kaggle users who used the very same metric and test set that is going to be implemented in this project (AUC metric and 2 last years as test set):

| Kaggle User | Method | AUC score | Reference |
|---|---|---|---|
| Aaron7sun* | CNN and LSTM | 62-63% | Link |
| Kate | Bernoulii Naive Bayes | 59% | Link |
| Dan Offer | Unknown | 56% | Link |
| Kate | Logistic Regression | 49% | Link |

* This was the user who provided the original database.

However, this benchmark will be used as a secondary benchmark. The following list to be considered as **primary benchmark**:

1. A ‘dummy’ classifier with random output (**test result = 49.13%**)
2. A logistic regression and a simple vectorizer with both using default parameters (**test esult = 41.36%**)

The logistic regression with default parameters is performing even worse than the dummy classifier due to overfitting in the training set. The overfitting is due to a very high number of featues (about 20 times the number of samples). The vectorizer parameter **max_depth** played a major role here in order to limit the number of features and handle the overfitting.

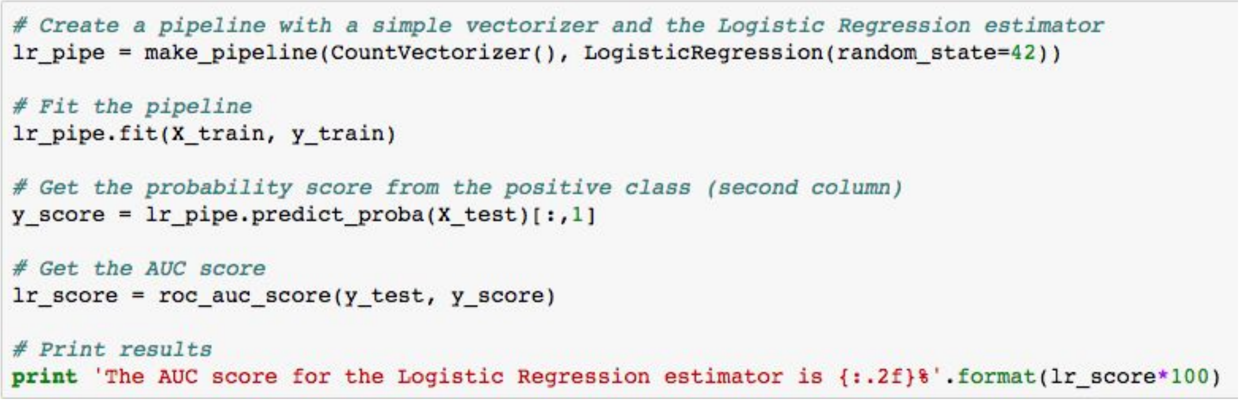
# III. Methodology

## Data Preprocessing

The preprocessing went as follows:

1. The text from the features was cleaned and HTML or terminal tags (such as `\n`) were removed using the function **clear_text.**
2. Next, all the 25 headlines where joined and processed into feature vectors using the model **bag of words**.
    a. First CountVectorizer() from sklearn was employed
    b. Later, the transformation tf-idf was also evaluated

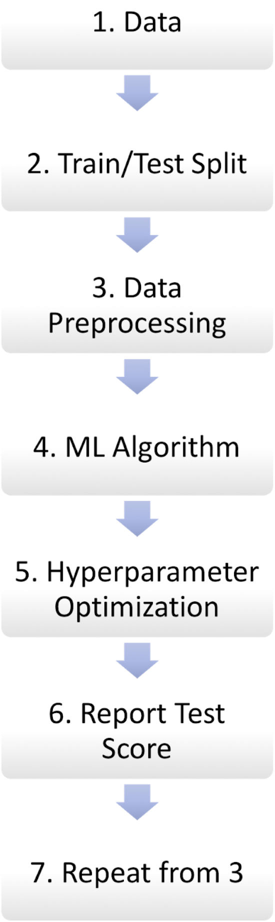
The preprocessing was performed using Pipelines, here’s an example:



```
# Create a pipeline with a simple vectorizer and the Logistic Regression estimator
lr_pipe = make_pipeline(CountVectorizer(), LogisticRegression(random_state=42))

# Fit the pipeline
lr_pipe.fit(X_train, y_train)

# Get the probability score from the positive class (second column)
y_score = lr_pipe.predict_proba(X_test)[:,1]

# Get the AUC score
lr_score = roc_auc_score(y_test, y_score)

# Print results
print 'The AUC score for the Logistic Regression estimator is {:.2f}%'.format(lr_score*100)
```

## Implementation

The implementation is summarized as follows:



Next some more details:

- A dataset with 1989 samples and 25 news headlines (which is later processed into vectors) was loaded as input.
- The most recent two years were used as the test set and the previous 6 years of data as the training set.
- Those 25 features are later joined and processed into feature vector. The preprocessing went as follows:
    a. The text from the features was cleaned and HTML or terminal tags (such as `\n`) were removed
    b. Next, the text was grouped and processed into feature vectors using the model **bag of words**.
- The AUC metric is used. As ML algorithms, the learning algorithm Logistic Regression is first evaluated since it fundamentally returns probabilities as output (which is necessary for the AUC metric). Later, other

learning algorithms are also evaluated, more specifically the following list: "k Nearest Neighbors", "Linear SVM", "RBF SVM", "Decision Tree", "Random Forest", "AdaBoost", "GradientBoost", "Gaussian Naive Bayes".

- The hyperparameter optimization was performed using a pipeline which evaluates:
    a. A pipeline with features from the vectorizer method (**TfidfTransformer**) for example, `stop_words`, `tokenizer` (a simple one or the PorterStemmer), `ngram_range`, `norm` and `use_idf`.
    b. Also parameters from the learning algorithm are also evaluated, for example `penalty` and `C` for Logistic Regression and n_estimators for RandomForest.
- The model with the highest validation score is going to be used to evaluate and report the test score.

## Refinement

The AUC test score of the **dummy estimator** was **49%**. Next, the AUC test score for **Logistic Regression** model using **default parameters** was **41.36%**. The logistic regression with default parameters is performing worse than even the dummy classifier due to **overfitting in the training set (the training score was 100%)**. The overfitting occoured due to a very high number of featues. The number of features is near 33000 which is more than 20 times higher than the number of samples (about 1600 samples). The vectorizer parameter **max_depth** played a major role here in order to limit the number of features and handle the overfitting problem. Here's some score when changing the value of **max_depth** in CountVectorizer and using Logistic Regression:

```
Using max_features = 1
The AUC score for the training set is 51.14%
The AUC score for the test set is 57.97%
==============================================================================
Using max_features = 5
The AUC score for the training set is 53.19%
The AUC score for the test set is 57.73%
==============================================================================
Using max_features = 10
The AUC score for the training set is 54.49%
The AUC score for the test set is 53.95%
==============================================================================
Using max_features = 20
The AUC score for the training set is 55.17%
The AUC score for the test set is 53.61%
==============================================================================
Using max_features = 25
The AUC score for the training set is 56.76%
The AUC score for the test set is 51.43%
==============================================================================
Using max_features = 30
The AUC score for the training set is 57.27%
The AUC score for the test set is 52.67%
==============================================================================
Using max_features = 40
The AUC score for the training set is 57.82%
The AUC score for the test set is 52.77%
==============================================================================
Using max_features = 50
The AUC score for the training set is 58.48%
The AUC score for the test set is 54.09%
==============================================================================
Using max_features = 60
```

```
The AUC score for the training set is 61.00%
The AUC score for the test set is 51.97%
================================================================================
Using max_features = 80
The AUC score for the training set is 64.42%
The AUC score for the test set is 49.87%
================================================================================
Using max_features = 100
The AUC score for the training set is 65.35%
The AUC score for the test set is 48.84%
================================================================================
Using max_features = 150
The AUC score for the training set is 67.95%
The AUC score for the test set is 48.37%
================================================================================
Using max_features = 200
The AUC score for the training set is 70.27%
The AUC score for the test set is 47.53%
================================================================================
```

The best test score was with max_features = 1, however the training score was lower than the test score which does not make sense - a model performing better with unseen data than with the data that it was training. The most reasonable value is max_features = 50 which led to the highest test score and still has the training score higher than the test score:

- **Test score for Logistic Regression (with max_features = 50)** = 54.09%

Also, some other learning algorithms were evaluated as well. Here's a summary of test scores:

- **Dummy Estimator** = 49.13%
- **Logistic Regression (with max_features = 50)** = 54.09%
- **Naive Bayes (with max_features = 50)** = 54.53%
- **Random Forest (with max_features = 50 and stop words)** = 56.24%

# IV. Results

## Model Evaluation and Validation

The Random Forest was first chosen for GridSearchCV due to the highest performance in the initial analysis. However, GridSearchCV didn't help to improve the best score of **56.24%** in the test set. The best test score with hyperparameter optimization was **51.73%** with RandomForest. Perhaps this is a more representative score since the parameters were optimized to the test set but to the validation set which avoids overfititng to the test set due to the choice of hyperparameters.

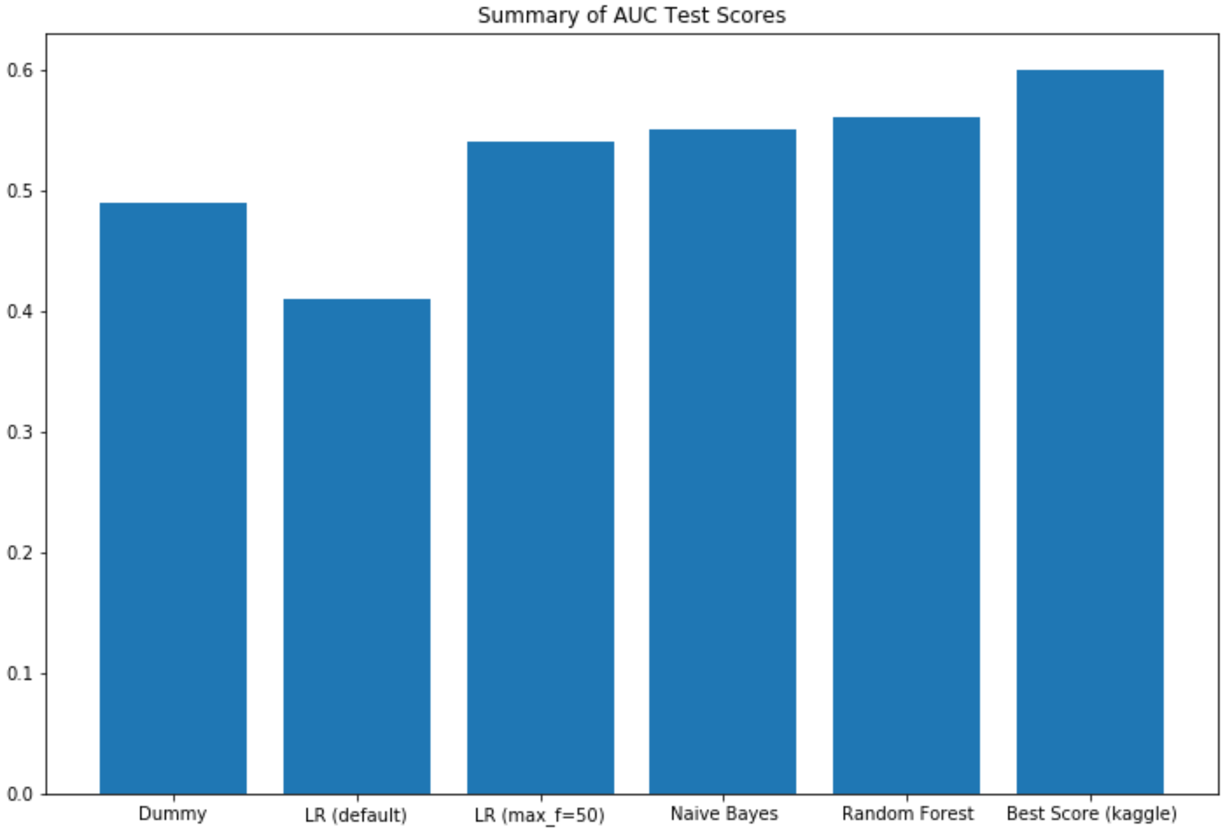
## Justification

The model chosen as final solution is **Random Forest** with the parameters max_depth=5, n_estimators=10, max_features=1. As vectorizer method it is chosen CountVectorizer with **max_features = 50** and using **stop words**. It reached the highest score in the test set of **56.24%**.

# V. Conclusion

## Free-Form Visualization

The following plot summarizes the best test scores obtained:



The best score was with Random Forest with 56.24% in the test set. The best reported score on Kaggle was 62%.

## Reflection

This capstone seeks a model which uses the top daily news headlines from Reddit (/r/worldnews) to predict stock market movement. First a Dummy estimator was implemented to serve as baseline for comparison. The AUC test score was 49%. Next, Logistic Regression with the default parameters was implemented. The score of 41% was obtained which is low due to overfitting. The CountVectorizer parameter max_depth was evaluated in order to handle the overfitting problem. The best test score with Logistic Regression was of 54% when using max_features = 50. Finally, other models were evaluated as well. The highest test score was obtained with RandomForest of 56%. When compared to the benchmark is 7% higher than the Dummy estimation and 15% higher than Logistic Regression with default parameters.

## Improvement

The best test score is still lower than two reported scores on Kaggle (59% and 62%), so still there's room for improvement. Also, no significant improvement was obtained with GridSearchCV. A further investigation is necessary with more hyperparameters and also checking if the highest obtained test score has parameters that are really representative for unseen data.

# References

1. https://en.wikipedia.org/wiki/Sentiment_analysis
2. https://arxiv.org/pdf/1610.09225.pdf
3. https://www.kaggle.com/aaron7sun/stocknews
4. https://en.wikipedia.org/wiki/Receiver_operating_characteristic
5. https://en.wikipedia.org/wiki/Bag-of-words_model
6. https://en.wikipedia.org/wiki/Tf%E2%80%93idf