<table>
<thead>
<tr>
<th></th>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>Background</td>
</tr>
<tr>
<td>3</td>
<td>Approach</td>
</tr>
<tr>
<td>4</td>
<td>Dataset</td>
</tr>
<tr>
<td>5</td>
<td>Evaluation</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
</tr>
<tr>
<td>7</td>
<td>Team Roles</td>
</tr>
<tr>
<td>8</td>
<td>References</td>
</tr>
</tbody>
</table>
INTRODUCTION

Given a large set of data, eg. a database of movie reviews in our case, the goal of Sentiment Analysis is to determine the emotional tone in each review and classify it as positive or negative. Since the dataset is huge, comprising of over ten thousand review files, we want to automate the process of sentiment analysis using machine learning and natural language processing techniques.

With the explosion of data in recent years, a lot of information is available to us that can be used to improve business strategies, help research and also solve social problems. Organizations can use sentiment analysis to know about customers’ reactions to their products and services. The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world.

From our research on past work done in sentiment analysis, there are a number of methods that have already been successfully used for the given problem of sentiment analysis. This has been discussed in more detail in the next page. We decided to use the Random Forest classifier. One of its main advantage is that it handles high dimensional spaces as well as large number of training examples very well and eliminates the problem of overfitting which is prevalent in such cases.
BACKGROUND

One research used three algorithms for sentiment classification into positive and negative categories - Naive Bayes, maximum entropy classification, and support vector machines. Features such as unigrams, bigrams and parts of speech (POS) were studied and compared. These algorithms do not perform as well on sentiment classification as they do on traditional topic-based categorization, although they do give reasonably good results. In terms of relative performance, Naive Bayes tends to do the worst and SVMs tend to do the best, although the differences aren’t very large.[1]

Another research performed sentiment analysis on confessions from the Experience Project, a collection of short, user-submitted posts reflecting the writer’s’ thoughts and actions. This required classifying the data into five categories: “Sorry, Hugs”, “You Rock”, “Teehee”, “I Understand” and “Wow, Just Wow”. The goal of the project was to perform two tasks: predict which label will receive the most votes (the “max label” task) and predict which labels will receive at least one vote (the “label presence” task). The feature models they used were ‘Bag of Words’, ‘WordnetSynsets’, ‘Sentiment Lexicon’, ‘TF-IDF’ and SVM. The results showed that Sentiment Lexicon worked best for “max label” task whereas SVM worked best for “label presence” task.[2]

Yet, another approach was taken by L.Mass to classify movie reviews. It captured both semantic and sentiment similarities among words. Semantic component of their model learned word vectors via an unsupervised probabilistic model of documents. It enabled a realization of how close the words ‘Wonderful’ and ‘Amazing’ were, however failed to capture the fact that they were words of positive extreme. The model was therefore extended with a supervised learning algorithm which used the vector representation of words to predict the sentiment annotations on contexts in which the words appeared i.e. similar vector representation for similar words.[3]

[3] Andrew, Raymond E. Daly, Peter T. Pham, Dan Huang, Christopher Potts. Stanford University
**APPROACH**

We were provided with a labelled dataset comprising of labelled 25,000 movie reviews with half of the reviews being positive and the other half being negative. We used this dataset to create a classification model that can classify the given test dataset comprising of 11,000 unlabelled movie reviews. A basic design of our model that takes in the train data and outputs the label of the test data is as follows.

We have coded for this project using the Python programming language. The first step is data cleaning and pre-processing. This entails removal of HTML markup, stop words, numbers and punctuation marks. The removal of HTML markup is done using the Beautiful Soup package. Beautiful Soup is a Python library for pulling data out of HTML and XML files. It saves some programming time by providing useful functionalities to deal with HTML markups. The removal of stop words and numbers is done using a package built-in with Python for dealing with regular expressions, called re. Using the re.sub function in this package we are able to get rid of all expressions that are not alphabets. Finally for the task of removing stop words, which are frequently occurring words that don't carry much meaning, we can use the Python Natural Language Toolkit(NLTK) which lets us import a list of common stop words in the English language.

The next step is called Feature-Extraction. We want to convert the set of movie reviews provided as paragraphs in English into a set of words, which can be used as ‘features’ to train our classification model. This is done using a well-known approach for dealing with textual data, called the Bag of Words. The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. For example, the following sentences
Sentence 1: "The cat sat on the hat"
Sentence 2: "The dog ate the cat and the hat"
can be converted into a feature count set as follows
Features: { the, cat, sat, on, hat, dog, ate, and }
Sentence 1: { 2, 1, 1, 1, 1, 0, 0, 0 }
Sentence 2: { 3, 1, 0, 0, 1, 1, 1, 1}
Since the large number of movie reviews provided will result in a massive vocabulary, we chose to limit the size of the feature vectors to the 5000 most frequent words.

The next step actually trains the classification model, by feeding the feature count set of all the movie reviews in the training dataset, as obtained from the previous step. We have used the Random Forest learning model. According to Wikipedia, Random Forest is “an ensemble learning method for classification, regression and other tasks, which operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.” The main advantage of this approach and the reason for us choosing this model is that it eliminates the problem of overfitting which becomes a major concern when using a large dataset. Since our dataset is not that large, as per machine learning standards, we also tried implementing the Naïve Bayes classification algorithm since it is easy to write and the classifier model is fast to build. Eventually comparing the accuracy obtained from both of these classification models we decided to go ahead with Random Forests.

Once the classification model is trained, we perform the same pre-processing and feature extraction steps on the given test data and then feed this feature count set to the learnt classification model, which gives us a predicted label (positive/negative) for each movie review.
### Dataset:
In order to make our classifier learn and predict the sentiment of a given review, we need to work with data. We were given two types of datasets:
- Training dataset
- Test dataset

### Test Dataset:
This is the dataset that we should use to test our algorithm. The test set contains 11,000 unlabelled reviews. Our goal is to feed this test set to our classifier and label accurately these reviews as positive (1) or negative (0).

### Training Dataset:
The training dataset was used to help the algorithm learn: obtain features from the dataset and form a classification logic based on these extracted features in order to classify a given test review as a positive or negative review.

### Format of the Training Dataset:
There are 25,000 reviews in the training dataset out of which 12,500 are positive and 12,500 are negative. The training dataset is available in the following format:
**Evaluation:**

We need to evaluate our algorithm to understand how it's performance. In order to do this, we calculate the prediction accuracy. Prediction accuracy is nothing but the percentage of labels that were predicted correctly. The higher the prediction accuracy, the better is our algorithm.

**Feature Extraction:**

Feature extraction is the process of extracting informative and non redundant values from a given dataset. These values are used for the learning procedure of machine learning algorithm in order to produce a classifier model.

Once we cleaned up the 25,000 reviews from the train set, we created a vocabulary using the Bag of Words model. The bag of words model computes the frequency of occurrence of each of these words in the form of features. And these features are used to train the classifier. We used a Sci-kit Learn feature extraction module to perform this action. This module extracts numerical features from the given movie reviews which are in text format in the following way:
- each string is converted into a 'token'
- each of these tokens are given token IDs
- the frequency of occurrence of each of these tokens is calculated
- then these tokens are organized based on how often they occur

While dealing with 25,000 reviews, we could possibly obtain a very large number of features. We cannot use all the features that are extracted. We need to select a certain number of feature vectors that we want to use. Getting this number right plays an important role in the resulting prediction accuracy. Choosing too less feature vectors obviously means that our prediction model would be at a disadvantage when it has to deal with a feature vector that wasn't used for training our model. And if this particular feature is quite frequent, it will result in poor prediction accuracy. Of course, missing out a single feature vector is not going to impact the accuracy much, but when the number of missed out important features are in hundreds or thousands, our accuracy is definitely going to drop. Similarly, selecting a large number of features is not a good strategy either as it results in the problem of over fitting, which again results in poor prediction accuracy. So it is quite essential to select the right number of features for your data.

For our program, we chose to limit our feature vectors to 5000. Upon testing and experimenting, we observed that selecting << 5000 or >> 5000 features was resulting in poor prediction accuracy.

We then created a final array of 25,000 reviews in rows and 5000 features.
Experiment Setup:
Our entire program was written in Python. Python is often preferred while dealing with large datasets. Most importantly, Python has some core libraries such as NumPy, SciPy and pandas which were very relevant and useful for our project.
In order to use these libraries, we need to explicitly install the following Python packages. We did all these installations using pip:
- pandas
- numpy
- scipy
- NLTK
- cython
- scikit learn
- beautiful soup
These are all the packages that need to be installed in order to run the code.

Results:
Using the random forest classifier we achieved a final prediction accuracy of 87.691%. This means that when we fed the 11,000 reviews of the test dataset to our code, we were able to classify about 8,770 of them correctly.

When we used <5000 features we got a prediction accuracy of ~84% and when we used >>5000 features, we got a prediction accuracy of ~83%
Results of some of the procedures:

Cleaning and parsing the movie reviews in the training set:

In [1]: %cd "C:\Users\admin\Desktop"
C:\Users\admin\Desktop

In [2]: %run "C:\Users\admin\Desktop\myBagOfWords.py"

story man unnatural feelings pig starts opening scene terrific example absurd
comedy formal orchestra audience turned insane violent mob crazy chantings
singers unfortunately stays absurd whole time general narrative eventually
making putting even era turned cryptic dialogue would make shakespeare seem
easy third grader technical level better might think good cinematography
future great vilmos zsigmond future stars sally kirkland frederic forest seen
briefly
Cleaning and parsing the training set movie reviews...

Review 1000 of 20000

Review 2000 of 20000

Review 3000 of 20000

Features:

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>accident</td>
<td>217</td>
</tr>
<tr>
<td>accidentally</td>
<td>139</td>
</tr>
<tr>
<td>accompanied</td>
<td>69</td>
</tr>
<tr>
<td>accomplish</td>
<td>61</td>
</tr>
<tr>
<td>accomplished</td>
<td>89</td>
</tr>
<tr>
<td>according</td>
<td>212</td>
</tr>
<tr>
<td>account</td>
<td>148</td>
</tr>
<tr>
<td>accuracy</td>
<td>64</td>
</tr>
<tr>
<td>accurate</td>
<td>221</td>
</tr>
<tr>
<td>accused</td>
<td>90</td>
</tr>
<tr>
<td>achieve</td>
<td>136</td>
</tr>
<tr>
<td>achieved</td>
<td>110</td>
</tr>
<tr>
<td>achievement</td>
<td>97</td>
</tr>
<tr>
<td>across</td>
<td>695</td>
</tr>
<tr>
<td>act</td>
<td>906</td>
</tr>
<tr>
<td>acted</td>
<td>501</td>
</tr>
<tr>
<td>acting</td>
<td>4735</td>
</tr>
<tr>
<td>action</td>
<td>2525</td>
</tr>
<tr>
<td>actions</td>
<td>232</td>
</tr>
<tr>
<td>activities</td>
<td>65</td>
</tr>
<tr>
<td>actor</td>
<td>1808</td>
</tr>
</tbody>
</table>

Cleaning and parsing the movie reviews in the test set:
Conclusion:
The random forest classifier algorithm gave use an accuracy of about 87%. We presume that some of the reasons for not getting a higher accuracy while predicting the sentiments of the review could be because the following type of misclassifications: when the person writing the review talks more about the plot/characters than about his opinion about the movie. This can potentially be misclassified. Another such case is when a review contains a lot of quotes. In such a case when the quotations are stripped off in the data preprocessing stage, the quotes are considered as part of the reviewer's opinion. This could result in a lot false positives and false negatives leading to lower prediction accuracy.

Team members' roles:
1. Anirban - Feature extraction and Random Forest Classifier
2. Murtaza - Initial format conversion of text files and Preprocessing
3. Swarna - Tried implementing Naive Bayes and finding precision, recall, f1-score

References:
1. Learning Word Vectors for Sentiment Analysis by Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts
2. Thumbs up? Sentiment Classification using Machine Learning Techniques by Bo Pang and Lillian Lee, Shivakumar Vaithyanathan
3. Machine Learning for Sentiment Analysis on the Experience Project by Raymond Hsu, Bozhi See and Alan Wu
5. Tsutsumi K, Shimada K, Nedo T. Movie Review Classification Based on a Multiple Classifier.