**NLP IA-1**

**Sentiment Analysis of Amazon review**

Aryan Balpande-16010120002

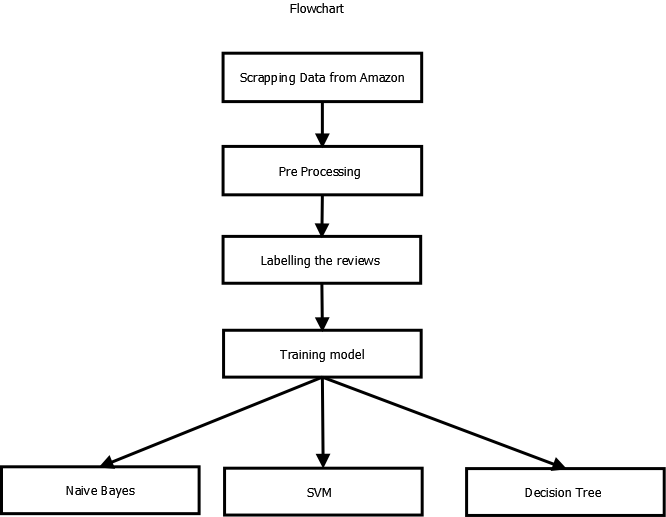
Vinod ghanchi-16010120016

Sneha Panditpautra-1601012036

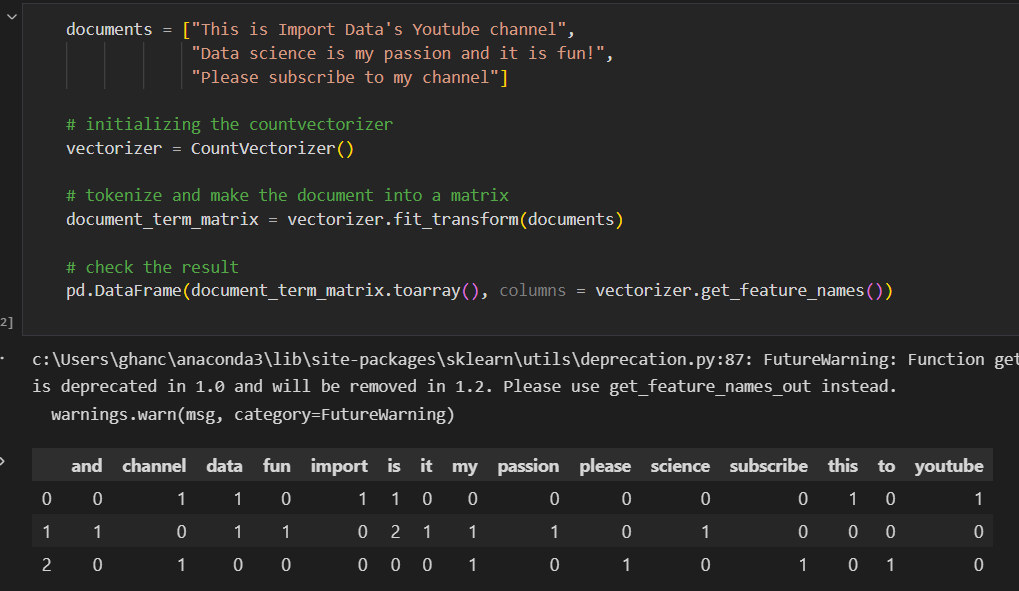
**Problem Statement**

To perform sentiment analysis on a dataset of Amazon school bag reviews scraped from the Amazon website. The goal is to develop a model that can accurately classify the sentiment of the reviews as positive, negative, or neutral.

**Flowchart of the project/task:**



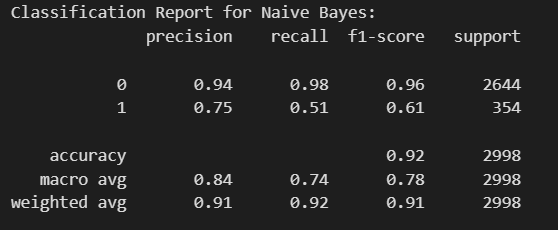
**Steps involved in completing this task:**

1. **Scrapping the data**: We scrapped the data from amazon.com using beautiful soup in python. In this step we created different function for extracting the single page data using data asin number(unique identification number) of products. Using this data-asin number, we were able to extract all the reviews of a particular product and also similar products from all the pages available. In this manner we extracted all the reviews of all the products from all the pages and stored them into a form of list. We converted this list into dataframe using pandas and then converted it to excel and csv.
2. **Preprocessing the data**: The data we extracted was not clean data. We need to clean the data first before we could apply any classification techniques on it. For this used “spacy” module. We checked for all the null values, and replaced it with empty string “”. Firstly we converted all the reviews from the reviews column to lowercase and stored it into a new column new\_reviews. Next, we remove punctuations, then emojis, pictographs and stopwords like a, the, etc and many more. We then lemmatized the words into their root words. This is the cleaned dataset on which we can work on.
3. **Labelling the reviews dataset**: For this step we imported modules “vader” and “nltk”. Using vader module, we calculated the polarity scores of each rows in reviews column. We got the values of positive, negative, neutral and compound. Depending on the compound values, we can label the reviews as having positive and negative polarity. For the compound values more than -0.5, we treated them as positive and labelled it as 0 and for the values less than -0.5, we treated them as negative review and labelled it as 1.
4. **Naïve Bayes model**: Using sklearn module, we applied naïve bayes classifier on our labelled data.Firstly we splitted the dataset into train and test set as 70% and 30%. Using the count vectorizer function, we tokenized the document into a matrix. We have also entered a user defined list to see the result of this function 

As shown CountVectorizer, it tokenize each word in the list and shows us the frequency. We applied the same to our train and test dataset.

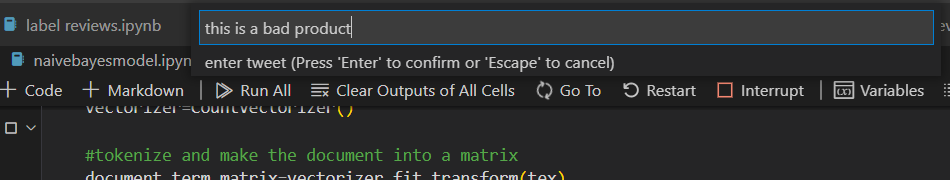
For this we used multinomial naïve bayes classifier model. After fitting the vector to the model, we got the following confusion matrix.

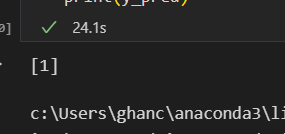




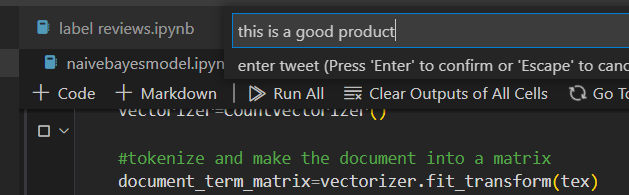
The accuracy score using this model was 92%.

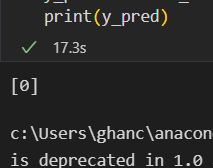
Using pickle module, we deployed our model or project in python to a .pkl format file.





As we can see it showed us the expected result i.e. negative sentiment.

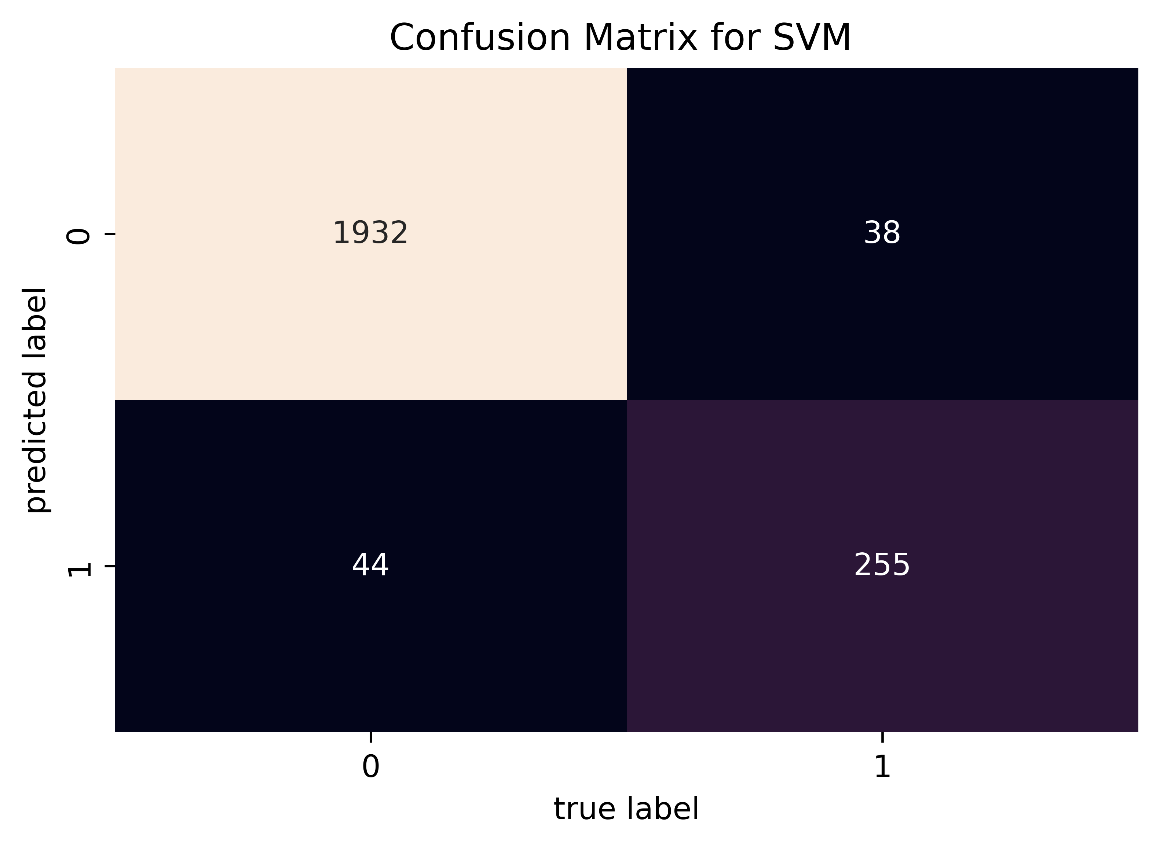


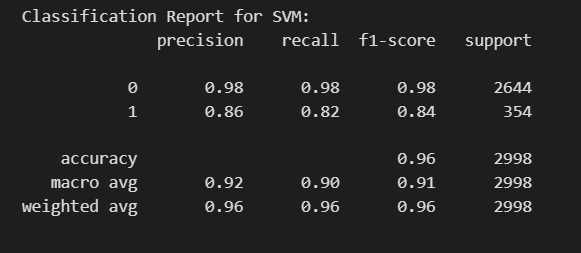


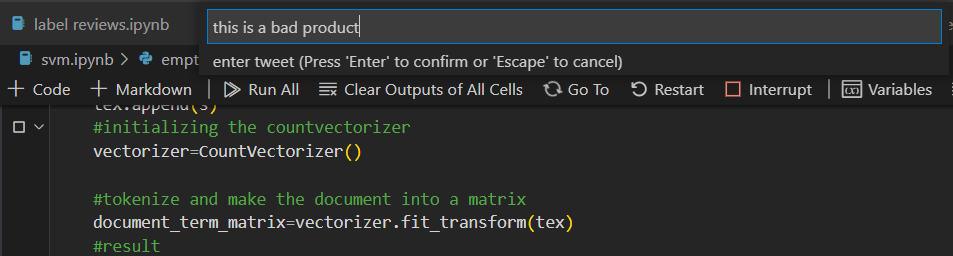
As we can see it showed us the expected result i.e. positive sentiment.

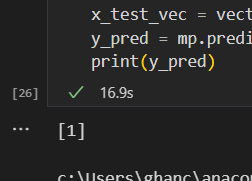
1. **SVM model**: The second model that we used was svm model. All the function and the code till the model selection is the same as the naïve bayes model code. Using sklearn we selected svm linear model. The accuracy using this model came out to be 96.38%.

After fitting the vector to the model, we got the following confusion matrix.

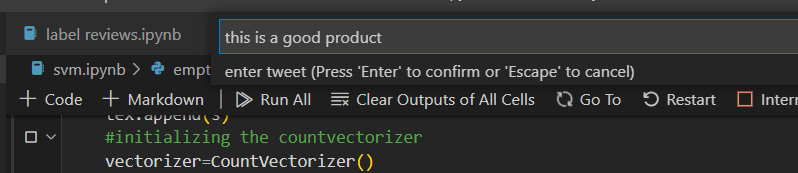


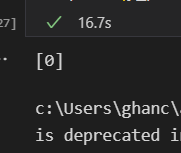






As we can see it showed us the expected result i.e. negative sentiment.

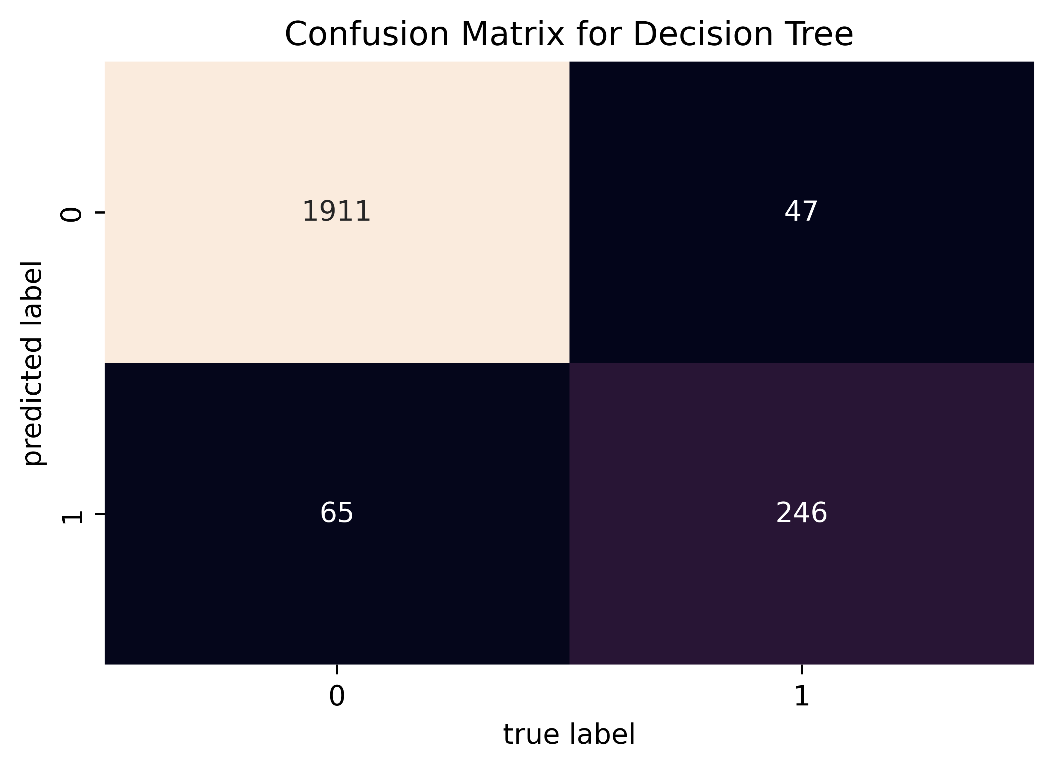


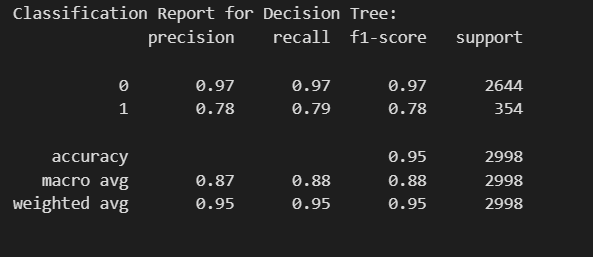


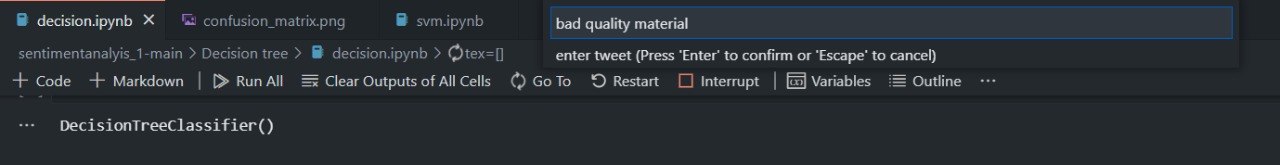
As we can see it showed us the expected result i.e. positive sentiment.

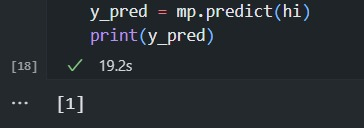
1. **Decision tree model**: The last model that we used was decision tree model. All the function and the code till the model selection is the same as the naïve bayes model code. Using sklearn we selected decision tree model and set the criterion to entropy. The accuracy using this model came out to be 95.01%.

The confusion matrix of using this model is as below:

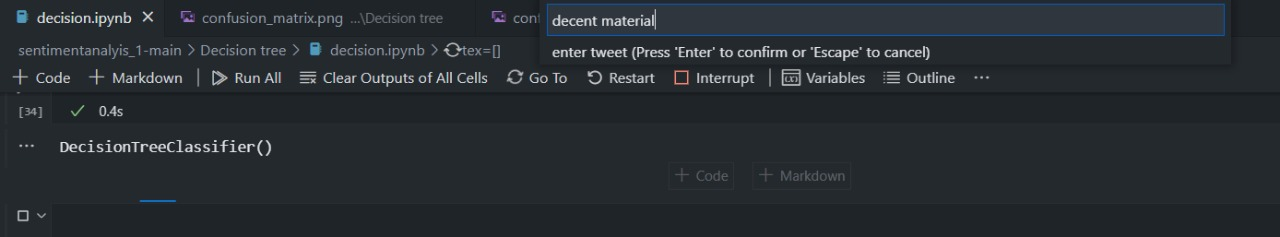


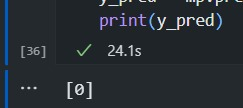


`



As we can see it showed us the expected result i.e. negative sentiment.





As we can see it showed us the expected result i.e. positive sentiment.

**Evaluation**

Comparison of the three models:

**Conclusion**

In conclusion, SVM's robustness to outliers and noisy data makes it a superior choice for sentiment analysis compared to Naive Bayes and Decision Trees. Sentiment analysis often involves dealing with sarcastic or ironic expressions, which can be considered as outliers in the dataset. While these outliers can significantly impact the performance of Naive Bayes and Decision Trees, SVM remains less sensitive to their presence. By focusing on maximizing the margin between the sentiment classes rather than being influenced by individual data points, SVM can effectively handle outliers and noisy data. This robustness ensures that SVM can capture the underlying sentiment patterns accurately, even in the presence of challenging and ambiguous expressions. As a result, SVM delivers more reliable and consistent results in sentiment analysis tasks, making it a preferred choice over Naive Bayes and Decision Trees.