Project Title:

Sentiment Analysis of Tweets

Project Members:

19BCE0249 (Kaustubh Dwivedi)

19BCE0276 (Harine A)

19BCE0253 (Nishma Avalon Rebello)

19BCE0200 (Soumyaraj Roy)

Final Project Review

Course Code: CSE3013 – AI

Slot: F1

Professor: Dr. W.B. Vasantha

1. Introduction

Sentiment analysis is a significant tool in social media monitoring and is often performed on textual data, as it allows us to gain an overview of the wider public opinion behind certain topics. Social media monitoring tools, for example, Brand watch Analytics make the process quicker and easier than ever before, because of Realtime monitoring capabilities. This project targets to enact on such a tool for all purpose utility. We have chosen twitter in this project for data collection, experimentation and analysis. Analysing the reactions on twitter, can give a true opinion analysis of majority of the people. We will check the success rate of different algorithms and implement an algorithm to our own understanding and incorporate an aspect that will allow us to judge sentiments in the tweets.

This project deals with Sentiment Analysis, also called opinion mining, which is a Natural Language Processing technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs.

We will use the concepts of Natural Language Processing (NLP), which is a field of Artificial Intelligence, in which computers are programmed how to process, analyze and understand large amounts of natural language data and hence derive meaning from human language in a smart and useful way.

Need of sentiment analysis

i) In Business:

In marketing field, the companies use it to develop their strategies or to understand customers' feelings towards products or brand, how people respond to their campaigns or product launches; and why do the consumers not buy some products.

ii) In Politics:

In the political field, it is used to keep track of political view and to detect consistency and inconsistency between statements and actions at the government level. It can also be used to predict election results.

iii) Public Actions:

Sentiment analysis can also be used to analyze and monitor social phenomena, for the spotting of potentially dangerous situations and determining the general mood of the blogosphere. By automatically sorting the sentiment behind reviews and social media conversations in open platform like Twitter, a company such as an E-Commerce company, can make faster and more accurate decisions for its market.

In this project we will perform the following tasks:

1. Firstly, Collection of Datasets from Kaggle.

2. Analyse the Data

3. Pre-process the data (transformation of data).

4. Train the data using python code.

5. Test the data.

6. Analyze the data and show result.

7. Analyze and show the level of accuracy of the prediction (analyze file)

2. Literature Survey

S.NO	Authors and Year (Reference)	Title (Study)	Concept / Theoretical model/ Framework	Methodology used/ Implementation	Dataset details/ Analysis	Relevant Finding	Limitations/ Future Research/ Gaps identified
1.	Erik Cambria and Soujanya Poria, Alexander Gelbukh, Mike Thelwall, 2017	Sentiment Analysis Is a Big Suitcase	This paper focuses more about deep learning and its implementation using NLP(natural language program)	This paper basically explains about the syntactic layer, and further breaks down to Microtext Normalization, Sentence Boundary Disambiguation, Part-of-Speech Tagging and their detailed	We can understand about the how sentimental analysis can be implemented through NLP, its structure and the break down and use of each	Sentiment analysis enormous bag of natural language processing (NLP) issues. sentimental analysis has for some time been confused with the undertaking of polarity detection. This,	Some NLP undertakings, in any case, require more than a simple data driven way to deal with accomplish human-like execution more pros and cons are to be discussed regarding the sentiment

				explanation . further more it gives us a more deep understanding of the semantics and the pragmatics layer.	process. Also the paper provides the problems to deal with in each process and alternatives.	notwithstanding, is only one of the numerous NLP issues that should be addressed to accomplish human-like execution in sentiment analysis.	approach towards NLP.
2.	Daniele Cenni, Paolo Nesi, Gianni Pantaleo, Imad Zaza, 2017	Twitter Vigilance: a Multi-User platform for Cross-Doma in Twitter Data Analytics, NLP and Sentiment Analysis	This paper proposes the twitter vigilance architecture, which is a cross-space, multi-client apparatus for gathering and dissecting Twitter information, giving aggregated measurements, dependent on the volume of	This journal first gives us an insight of what a social media analytics platform is, and also explains how sentiment analysis of social media can be influenced such as surveying customer sentiments, anticipating monetary and market results , anticipating political race results, giving	In order, to build this architecture, the important aspects concerned are data, NLP and Sentiment Analyses based metrics, API availability, User network analysis, real time	The extraction of Part-of-Speech (POS) labelled keywords and calculation of catchphrase event at various time goal. sentiment polarity extraction for each single tweet; this sort of data can be valuable to evaluate and appraise the overall notion of the Twitter local area in	This paper explains the architecture very well, with help of case study and understand the implementation of the sentiment analysis through NLP and social media analytics.

			tweets and retweets, clients' impact organization, Natural Language Processing and sentiment Analysis of textual content.	early recognition and cautioning for unfavourable technical issues as well as for disaster response surveillance systems.	analysis and full faceted search. The paper also gives a detailed analysis about the architecture.	regards to a particular channel or search. lower-level measurements are utilized to weight keyword events (registered as NLP-based measurements, as recently depicted) to assess the most powerful keywords for conclusion examination, just as distinguishing conceivable sources and motivations to clarify or decipher explicit supposition patterns.	

3.	S.Muthukumaran, Dr.P.Suresh 2018	Text Analysis for Product Reviews for Sentiment Analysis	This paper clarifies various strategies for sentiment analysis and displays a productive methodology. It likewise features	The proposed method they have analyzed various types of algorithms, for predicting semantic orientation. They utilized four-stage	This paper distinguishes solid pieces of information of subjectivity utilizing the consequences of a technique for bunching words	The paper mainly focuses on the implementation of the sentiment analysis. We can also understand more about opinion mining as well as sentiment analysis. The	As future work of this journal, we can refine rule set to extricate more reliance relations from datasets and that will assist with improving the precision and

			using NLP Methods	the significance the item surveys are of most extreme significance for the purchasers to choose depending on their interests with respect to item's different angles for instance a monitor, processor speed, memory.	supervised learning algorithm to derive the semantic direction of descriptive words from constraints on conjunctions. The texts are at that point tokenized into tokens and the stop-words are recognized and taken out. The audits for a couple of mainstream telephones have been gotten by building a web crawler. The web crawler has been written in Python utilizing a scraping library called Beautiful Soup. Alongside the survey text, some extra information bunches and the lexical semantic highlights are appeared to have higher exactness than.	as indicated by distributional similarity. Basically this paper uses Flip kart Reviews Database as dataset for this project . The main source of data used is the product reviews from Amazon. They utilize Dirichlet distribution and Bayesian Classification , that represents a supervised learning method as well as a statistical method for classifications of various words and their meaning.	architecture proposed utilizes a non-supervised sentiment order, approach for sentiment classification and it is assessed utilizing a dataset of online client surveys of cell phones. This paper shows that, the framework performs very well in opinion arrangement of client surveys with high exactness. implemented fuzzy functions to emulate the effect of various linguistic hedges such as dilators, concentrator and negation on opinionated phrases help the system to achieve more accuracy in sentiment classifications.	review estimations of the framework by characterizing algorithms. In the event that the framework ready to right all the spelling and syntactic blunders present in the survey reports in the pre-processing step itself that will improve the review estimation of the System execution.

4.	Alex Mordkovich, Kelly Veit, Daniel Zilber 2011	Detecting Emotion in Human Speech	This paper mainly focuses on detection of emotion in speech, they use a free software(praat) which is used to process the audio data and extract various statistics which includes voice report. The statistic in the report (pitch, pulses, voicing, jitter, and harmonicity) are determined through this report. It also emphasises about Mel-frequency cepstral coefficients (MFCCs) which are a common set of features used in voice processing algorithms.	In this paper in order to analyse various emotions of the speaker, the sound accounts and record information are pre-processed in a custom four-stage pipeline to produce an information document in which every expression is an information test addressed as a single line. This subsequent information record is stacked into MATLAB as a stylized design matrix. The chose K-Means clustering as their classification algorithm for simplicity purpose. Also they experimented with the SVM implementations in Liblinear, LibSVM, and the MATLAB-builtin	In this paper for preparation, cross-approval, and testing, they utilized the Emotional Prosody Speech and Transcripts acquired from the Linguistic Data Consortium. This information comprises accounts of expert actors discussing dates and numbers with different emotional intonations. The semantic content of the expressions is proposed to be sincerely unbiased, as a type of mental control in the examples.	Through this paper we could analyse the detection of emotion in the human speech by analysing various aspects and they concluded that we could take the sample inputs and classify them under 14 emotions, when trained on the complete training data set.	The feature selection was executed in two ways. The main route was to discover the mix of 1-3 features that limits training error. The subsequent methodology was to utilize a forward or in backward search heuristic. Neither one of the approaches improved outcomes fundamentally. These search algorithms for new features end up being slow. Basically, the main aim of this proposal was to analyse various emotions through the voice. Now they could actually classify under 14 emotions but it can be still improved to expand the classifications to improve the

				SVMClassify for our classification tasks.			accuracy.
5.	Milad Sharif, Soheil Norouzi 2011	Sentiment based model for Reputation system in Amazon	This paper mainly proposes that create tools to evaluate the semantics of item audits and determining the polarity of opinions. Also to assess the strength of an opinion is utilizing audits with numeric ratings and preparing (semi-)supervised learning calculations to arrange surveys as certain or negative	In this paper we could analyse that the semantic orientation and strength of a survey is anticipated by following the adjustments in the related financial factors of a dealer. the technique utilizes two diverse parallel classifiers (for example Innocent Bayes and semi-directed recursive auto-encoder) to foresee the exceptional cost of an item. The notion investigation calculation (for example RAE) was conveyed to acquire the semantics of the item surveys and gave a model to the exceptional costs	The data set incorporated in this paper details of the different transactions that occurred on Amazon.com for a wide range of software items. The data set gathered from freely accessible data at Amazon.com by utilizing Amazon Web Services. The data set incorporates two sections, transaction history and reputation data. The initial segment comprises of transaction IDs at every product and the cost at which the products were	Basically this journal helps us to analyse the semantic orientation and strength of a review incorporated in the amazon service d by tracing the changes in the associated economic variables of a merchant. Various algorithms and methods such as multivariate Bernoulli event model and Semi-supervised Auto-Encoder (RAE) architecture are well explained and utilized in their model. Basically, they conducted a survey in which they used different binary classification models to accurately predict the polarity of the premium price that a merchant gets based on the costumer	This paper provides us a very simple survey of analysing different classification model and determined the accurate one which can predict these distributions more accurately than other models. Further more research to be done on how its it better than other models and how the accuracy can be improved.

					sold. The second piece of data set incorporates the reputation history of every trader that had a product available to be purchased during the time frame which the data set was gathered.	reviews	

3. Objective of the project:

Sentiment analysis, which is otherwise called opinion mining, assessment extraction, sentiment mining or subjectivity analysis is the way toward dissecting if a piece of web-based composition (online media notices or blog entries or news locales, or some other piece) communicates positive, neutral or negative perspectives.

There are a lot of aspects in which the sentiment analysis can be done and it can be really helpful in various situations. In our project we have taken the airlines sentiment analysis.

The main objective of our project is to analyses and monitor social platforms accessed by various users from different backgrounds to detect consistency and inconsistency between statements and actions based on uses such as product development, governmental issues, and determining the general mood of the blogosphere.

As per the first source, A sentiment analysis work about the issues of each major U.S. carrier. Twitter information was scratched from February of 2015 and benefactors were asked to initially group positive, negative, and unbiased tweets, trailed by sorting negative reasons, (for example, "late flight" or "discourteous service").

Our objective is to investigate the sentiment and anticipate the class of the sentiment as "positive", "neutral" or "negative".

4. Innovation component in the project:

In our case we have taken the ‘twitter airlines sentiment analysis’ as the topic for our sentimental analysis. This data utilized for this analysis was taken from Kaggle.com.

In this project we have researched and analyzed the success rate of the sentimental analysis by testing through various algorithms such as

1. Logistic regression
2. 2. Naive bayes
3. 3. SVM
4. 4. Random Forest

To compute the general extremity of a post, we alluded to the past researches and added some worth by presenting new metrics. This sentiment Analysis Model which we have made can be utilized in various situations such as governance, public opinion predictions, e-commerce…etc

5. Work done and implementation

a. Methodology adapted

1. Use data sets from Kaggle (An online platform capable of providing users with datasets)

2. Analyse the data using various Visualization libraries

3. Pre-processing the data

Cleaning, normalization, transformation, feature extraction and selection, and so on are all part of the pre-processing. The result of pre-processing would be coherent and uniform data that can be used to improve the performance of the classifier.

4. Use python code to train the data set

The machine learning modules which we plan to use include SVM, Logistic regression, random forest, Naïve Bayes.

One of the most basic text classification algorithms is the Naive Bayes classifier. It's a simple classifier based on the Bayes theorem that makes naive assumptions about the feature variables' independence.

Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model

Support vector machines so called as SVM is a supervised learning algorithm. The Ideology behind SVM is to find a hyperplane that best separates the features into different domains.

5. Testing phase

6. Analyze the data and display results

Perform Sentiment Analysis on Tweets After gathering and cleaning our data set, we are ready to execute the sentiment analysis algorithm on each tweet. Then, we will calculate an average score for all the tweets combined.

7. Visualization of results using graphs and charts

We plan to use Pyplot to display figures. matplotlib.pyplot is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc

8. Analyze as well as show the accuracy-level of the prediction (analyses file).

Tf-Idf

Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

Bag of words (BoW) converts the text into a feature vector by counting the occurrence of words in a document. It is not considering the importance of words. Term frequency — Inverse document frequency (TFIDF) is based on the Bag of Words (BoW) model, which contains insights about the less relevant and more relevant words in a document. The importance of a word in the text is of great significance in information retrieval.

Example — If you search something on the search engine, with the help of TFIDF values, search engines can give us the most relevant documents related to our search.

Term Frequency (TF)

It is a measure of the frequency of a word (w) in a document (d). TF is defined as the ratio of a word's occurrence in a document to the total number of words in a document. The denominator term in the formula is to normalize since all the corpus documents are of different lengths.

$$TF(w,d) = \frac{occurences\ of\ w\ in\ document\ d}{total\ number\ of\ words\ in\ document\ d}$$

Inverse Document Frequency (IDF)

It is the measure of the importance of a word. Term frequency (TF) does not consider the importance of words. Some words such as' of', 'and', etc. can be most frequently present but are of little significance. IDF provides weightage to each word based on its frequency in the corpus.

$$IDF(w,D) = \ln(\frac{Total\ number\ of\ documents\ (N)\ in\ corpus\ D}{number\ of\ documents\ containing\ w})$$

Term Frequency — Inverse Document Frequency (TFIDF)

It is the product of TF and IDF.

TFIDF gives more weightage to the word that is rare in the corpus (all the documents).

TFIDF provides more importance to the word that is more frequent in the document.

$$TFIDF\ (w,d,D) = TF(w,d) * IDF(w,D)$$

Term Frequency — Inverse Document Frequency (TFIDF) is a technique for text vectorization based on the Bag of words (BoW) model. It performs better than the BoW model as it considers the importance of the word in a document into consideration. The main limitation is that it does not capture the semantic meaning of the words. This limitation of TFIDF can be overcome by more advanced techniques such as word2Vec.

Parameters used:

input: text document

lowercase : bool(Default-True). Convert all characters to lowercase before tokenizing.

stop_words : Remove the defined words from resulting vocabulary.

ngram_range : The lower and upper boundary of the range of n-values for different n-grams to be extracted.

max_df : Ignore the term that has a document frequency higher than a threshold.

min_df : Ignore the term that has a document frequency lower than a threshold.

max_features : Build a vocabulary that only considers top max_features ordered by word occurrence.

norm : ‘l1’, ‘l2’ or ‘None’ (Default-‘l2’)

use_idf : boolean (default=True). Enable inverse-document-frequency reweighting.

smooth_idf : boolean (default=True). Smooth idf weights by adding one to document frequencies, as if an extra document was seen containing every term in the collection exactly once. Prevents zero divisions.

sublinear_tf : boolean (default=False). Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).

Machine Learning Model Used:

1. Support Vector Machine: Support vector machines so called as SVM is a supervised learning algorithm which can be used for classification and regression problems as support vector classification (SVC) and support vector regression (SVR).

2. Logistic Regression: Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist.

3. Random Forest Classifier: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

4. Multinomial NB: The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts.

Major Libraries Used:

1. Pandas : Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures.

2. Numpy : NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.

3. Matplotlib: Matplotlib is one of the most popular Python packages used for data visualization.

4. Seaborn: Seaborn is a library mostly used for statistical plotting in Python.

5. Sklearn Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python.

HARDWARE AND SOFTWARE REQUIREMENTS:

SOFTWARE REQUIREMENT:

1. Operating System: Windows 7, Windows XP, Windows Vista or higher versions

2. Programming Language: Python 3

3. Coding Platform: Any Python Platform such as Anaconda, Spyder or Jupyter Notebook

4. Modern Web Browser: Preferably Chrome or Firefox or Edge

5. Kaggle Data Set

HARDWARE REQUIREMENT:

1. RAM: 1GB or more

2. Processor: Any Intel Processor

3. Hard Disk: 6GB or more

4. Speed: Min 1 GHz

5. No additional hardware components are required.

b. Dataset used:

a. The dataset used is Twittern Airlines sentiment Analysis.

The data is collected from-

https://www.kaggle.com/crowdflower/twitter-airline-sentiment

According to the original source "A sentiment analysis job about the problems of each major U.S. airline. Twitter data was scraped from February of 2015 and contributors were asked to first classify positive, negative, and neutral tweets, followed by categorizing negative reasons (such as "late flight" or "rude service")".

Our job is to analyze the sentiment and predict the category of the sentiment as "positive","negative","neutral". The data embodies the relationship mapping tweets to their author's sentiments: positive or negative. The tweets have been extracted using the twitter api.

Tools Used: We have used Jupyter Notebook as a tool for running our ML Models.

b. Reference paper we are taking in consideration is

Sailunaz, K. (2018). Emotion and Sentiment Analysis from Twitter Text (Unpublished master's thesis), University of Calgary, Calgary.

The link for the following project is mentioned below:

https://prism.ucalgary.ca/handle/1880/107533

c. Our project differs the above research paper that we are analyzing the tweets and using Machine Learning models and Natural Language Processing concepts to predict whether a tweet tweeted by the user is positive negative or neutral and analyze the other trends with highest accuracy. Our is a practical hands-on approach by using NLP and ML concepts.

d. Screenshot and Demo:

1. First step is to import all the libraries that are required in this project.

We imported pandas to manipulate our data, converting the csv file to data frame so that we can use it for Models. Pyplot is used to plot graph in python, we will use it to plot a bar graph for comparison of the models.

Sklearn is the most important library we will use here for importing various models that we will use to train our data and compare their accuracy using classification report, accuracy score, confusion matrix, etc. We will also import TfidfVectorizer.

A Scikit-Learn provides the implementation of the TfidfVectorizer.

```
In [2]: from __future__ import print_function
import sys
%matplotlib inline
import pandas as pd
import itertools
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
import re
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction import text
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix, roc_curve
from sklearn.feature_extraction.text import TfidfVectorizer
```

Read the csv file

```
In [4]: input_df= pd.read_csv("../Tweets.csv")
```

Check first few rows of the table

```
In [5]: input_df.head()
```

Out[5]:

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdi
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardi
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonr
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardi
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardi

2. Then we will read our dataset using pandas read csv function which reads a csv file and returns a dataframe.

We will store that dataset in our variable input_df .

Head function is used to view top 5 rows of our dataset.

Analysing the data and get some insights

The shape of the table: rows=14640, columns=15. Each row corresponds to one twitter user.

In [6]:
```
input_df.shape, input_df.columns
```

Out[6]:
```
((14640, 15),
 Index(['tweet_id', 'airline_sentiment', 'airline_sentiment_confidence',
        'negativereason', 'negativereason_confidence', 'airline',
        'airline_sentiment_gold', 'name', 'negativereason_gold',
        'retweet_count', 'text', 'tweet_coord', 'tweet_created',
        'tweet_location', 'user_timezone'],
       dtype='object'))
```

Total number of passengers grouped by airlines

In [7]:
```
input_df["airline"].value_counts()
```

Out[7]:
```
United            3822
US Airways        2913
American          2759
Southwest         2420
Delta             2222
Virgin America     504
Name: airline, dtype: int64
```

There are 3 different category of sentiments -"positive","negative","neutral"

In [8]:
```
input_df["airline_sentiment"].value_counts()
```

Out[8]:
```
negative    9178
neutral     3099
positive    2363
Name: airline_sentiment, dtype: int64
```

Checking if there any null entry. Most importantly we care if the "text" entry is blank. As foolows there is no empty "text" entry. Good for us :). We do not need to deal with missing entry problem.

3. The next part is to do some data analysis on our dataframe using various pandas function. We will count the various values of different airways present in our dataset. Then we will count the number of sentiments available for each type that is positive, negative and neutral.

```
In [9]: input_df.isnull().sum()
```

```
Out[9]: tweet_id                          0
        airline_sentiment                 0
        airline_sentiment_confidence      0
        negativereason                 5462
        negativereason_confidence      4118
        airline                           0
        airline_sentiment_gold        14600
        name                              0
        negativereason_gold           14608
        retweet_count                     0
        text                              0
        tweet_coord                   13621
        tweet_created                     0
        tweet_location                 4733
        user_timezone                  4820
        dtype: int64
```

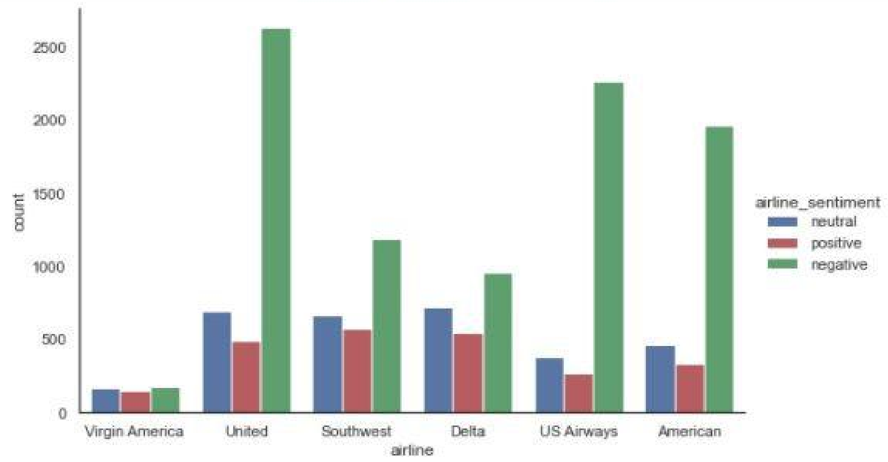
No of "postive", "negative" and "neutral" sentiments per airline

```
In [10]: sentiments_per_airline=pd.crosstab(input_df["airline_sentiment"],input_df["airline"])
         sentiments_per_airline
```

Out[10]:

airline	American	Delta	Southwest	US Airways	United	Virgin America
airline_sentiment						
negative	1960	955	1186	2263	2633	181
neutral	463	723	664	381	697	171
positive	336	544	570	269	492	152

```
In [11]: palette ={"positive":"C3","negative":"C2","neutral":"C0"}
         with sns.axes_style('white'):
             sns.factorplot("airline",data=input_df, aspect=1.5,\
                            hue='airline_sentiment', kind='count',palette=palette)
```



We see from the table "sentiments_per_airline" is not a good way of measuring sentiment rating as there are different no of passengers per airline. For example: Virgin America has only 504 total passengers where United has 3822 passengers.

A better way to compare ratings is to compute the percent of positive, negative, and neutral ratings.

```
In [12]: p=sentiments_per_airline.apply(lambda x: x/ x.sum() * 100.)
         p
```

Out[12]:

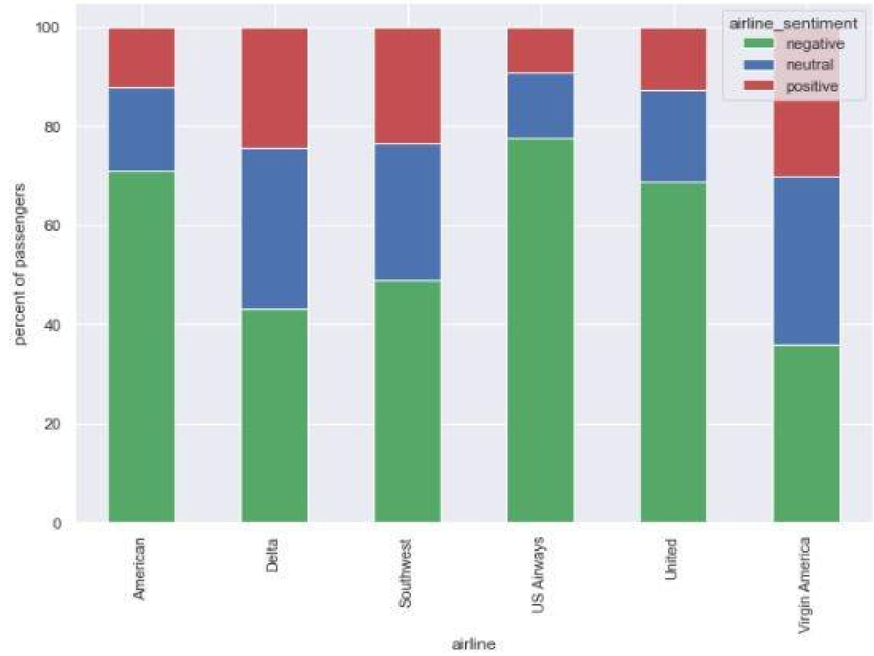
airline	American	Delta	Southwest	US Airways	United	Virgin America
airline_sentiment						
negative	71.040232	42.979298	49.008264	77.686234	68.890633	35.912698
neutral	16.781443	32.538254	27.438017	13.079300	18.236525	33.928571
positive	12.178325	24.482448	23.553719	9.234466	12.872841	30.158730

4. After counting the different sentiments we will use seaborn library to plot the three sentiments against the different airlines to get the insight of data. Then we will find the percentage of the data for each airlines for different kind of sentiment.

We observe that the maximum negative tweets came for the United Airlines (71%) while the positive tweets came from the southwest airlines.(25%).

In [13]:
```
my_colors = 'gbred'
p.T.plot(kind='bar',figsize=(10, 7),stacked=True,color = my_colors)
plt.ylabel("percent of passengers")
```

Out[13]: Text(0, 0.5, 'percent of passengers')



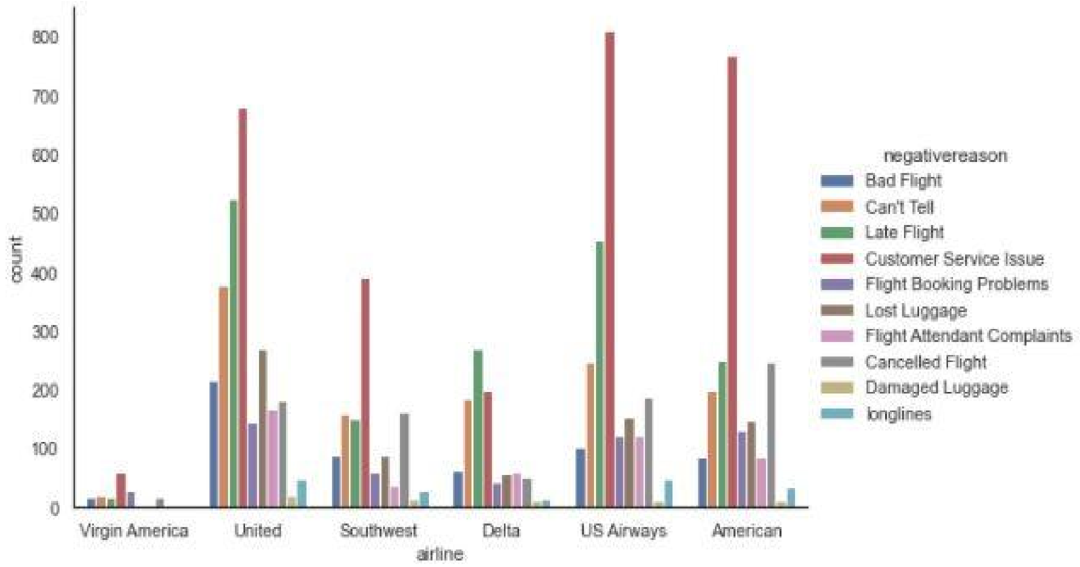
Let's analyze the reason for giving negative rating. Looking at the following numbers and the plot we can infer that the "Customer Service Issue" and "Late Flight" play vital role.

In [14]:
```
input_df["negativereason"].value_counts()
```

Out[14]:
```
Customer Service Issue         2910
Late Flight                    1665
Can't Tell                     1190
Cancelled Flight                847
Lost Luggage                    724
Bad Flight                      580
Flight Booking Problems         529
Flight Attendant Complaints     481
longlines                       178
Damaged Luggage                  74
Name: negativereason, dtype: int64
```

5. Then we have represented the same information of the previous percentage in the form of bar graph. Then We have found out the various reasons for negative tweets by the users. We found out that customer service issues and late flight are the two main reasons for negative tweets.

In [15]:
```
df=input_df.groupby(['airline','negativereason'])
df = df["airline_sentiment"].value_counts()
with sns.axes_style('white'):
     sns.factorplot("airline",data=input_df, aspect=1.5,\
                    hue='negativereason', kind='count')
```



In [16]:
```
input_df["negativereason"].isnull().sum()
```

Out[16]: 5462

Around 40 % = (5462/14640)*100 of the "negativereason" rows are empty. Therefore we will use only "text" column for our model as X.

6. Then we have presented the same information in the form of visualization using bar graph for different airlines : how much a reason is responsible for a negative tweet for various flight.

Data Preprocessing

In [17]:
```
X = input_df['text']
y = input_df['airline_sentiment']
```

Split data into train, test

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42)
X_train.shape, y_train.shape
```

Out[18]: ((7320,), (7320,))

We will use scikit-learn's feature extraction tool: Term Frequency times Inverse Document Frequency (tf-idf) For more details please see -https://en.wikipedia.org/wiki/Tf–idf

For improving performance of our model we shrink the vocabulary by removing stopwords from the "english" library which will remove some redunadant words "a", "the", "about"

In [19]:
```
def plot_confusion_matrix(cm,target_names,title='Confusion matrix',normalize=True):
    cmap = plt.get_cmap('Reds')
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names)
        plt.yticks(tick_marks, target_names)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white"
                     if cm[i, j] > thresh else "black")

        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()
```

7. Here is the important step of our project i.e. Data PreProcessing

We have divided the data into X (our features - text) and Y (our target which we have to predict - airline-sentiment). After that we have used train test split function of scikit learn library for dividing the whole dataset into training and testing set.We have divided the dataset into two equal parts for testing and training sets.

After that we have defined our function plot_confusion_matrix for plotting the classification report that we will get from testing of different models.

Different Models

a. Our first model **Multinomial Naive Bayes Classifier** that we have used for predicting values on our test dataset. On trying various values for alpha, we have found it that it has given best result when the value of alpha is set to 1. For tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix.

The accuracy that we get after using this model is around 68% which is very low and can be more improved by using various other models.After that we have called our function that we have defined before plot_confusion_matrix and plotted our results for visual representation of our data.

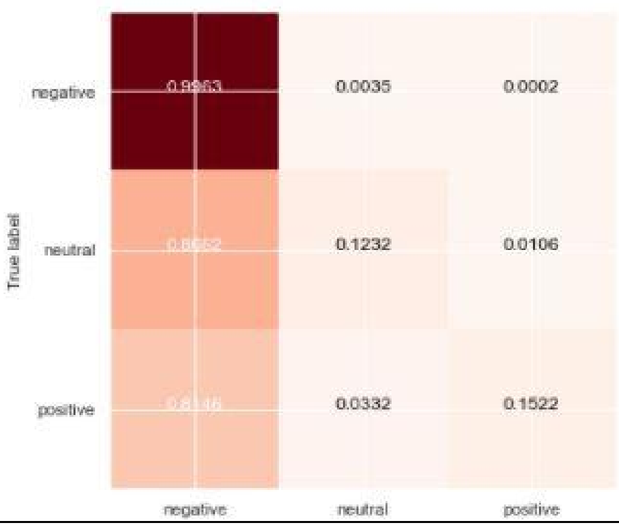
1. Naive Bayes Classifier

In [20]:
```
from sklearn.naive_bayes import MultinomialNB

parameters = {"alpha" : [1.0, 5.0, 10.0, 50.0, 100.0]}
tfidf=TfidfVectorizer(stop_words='english')
clf=GridSearchCV(MultinomialNB(),parameters,cv=2,refit=True)
model= make_pipeline(tfidf,clf)
model.fit(X_train, y_train)
```

Out[20]:
```
Pipeline(steps=[('tfidfvectorizer', TfidfVectorizer(stop_words='english')),
                ('gridsearchcv',
                 GridSearchCV(cv=2, estimator=MultinomialNB(),
                              param_grid={'alpha': [1.0, 5.0, 10.0, 50.0,
                                                    100.0]}))])
```

In [22]:
```
print("Best parameters set:",clf.best_params_)
print("Grid scores on every set of parameters:")
print()
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
# for mean, std, params in zip(means, stds, clf.cv_results_['params']):
#     print("%0.3f (+/-%0.04f) for %r"
#           % (mean, std * 2, params))

print()
print("Classification report:")
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Test accuracy:",accuracy_score(y_test, y_pred))
labels = model.classes_
matrix = confusion_matrix(y_test,y_pred)
print(pd.DataFrame(matrix,columns=labels, index=labels))
plot_confusion_matrix(matrix,labels)
```

```
Best parameters set: {'alpha': 1.0}
Grid scores on every set of parameters:


Classification report:
              precision    recall  f1-score   support

    negative       0.67      1.00      0.80      4634
     neutral       0.77      0.12      0.21      1510
    positive       0.91      0.15      0.26      1176

    accuracy                           0.68      7320
   macro avg       0.79      0.42      0.43      7320
weighted avg       0.73      0.68      0.59      7320

Test accuracy: 0.6806010928961749
          negative  neutral  positive
negative      4617       16         1
neutral       1308      186        16
positive       958       39       179
```



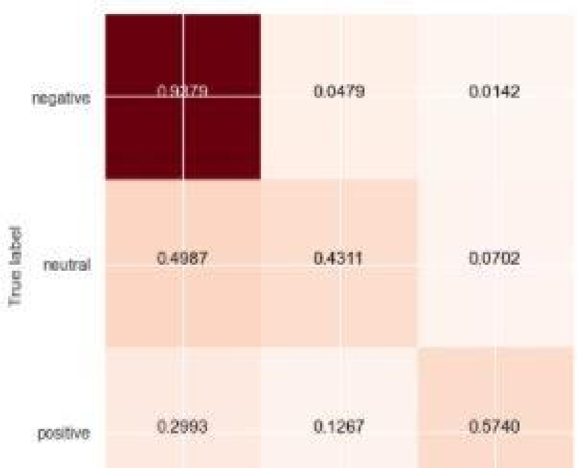
b. The second model we have used is the **logistic regression model** used mainly for binary classification but can also be used for multiclass classification. On trying various parameters we found out that it gives optimum result when l1_ratio is 0.2 , solver is ‘saga’ and tolerance is 0.01 and when fit_intercept is true. For tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix.

Logistic Regression has given a huge improvement as compared to Naive Bayes Classifier on the accuracy as our accuracy reached around 77%. After that we have plotted the graph of our result predictions using our function plot confusion matrix which we have defined before.

Logistic Regression

```
In [37]: from sklearn.linear_model import LogisticRegression

         parameters = {"penalty" : ['l1', 'l2', 'elasticnet', 'none'], "tol" : [1e-2, 1e-3, 1e-4], "C" : [ 1.0], "l1_ratio" : [0.2, 0.4, 0.6, 0.
         8],
                       "fit_intercept" : [True], "solver" : ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'] }
         tfidf=TfidfVectorizer(stop_words='english')
         clf=GridSearchCV(LogisticRegression(),parameters,cv=2,refit=True)
         model= make_pipeline(tfidf,clf)
         model.fit(X_train, y_train)

Out[37]: Pipeline(steps=[('tfidfvectorizer', TfidfVectorizer(stop_words='english')),
                         ('gridsearchcv',
                          GridSearchCV(cv=2, estimator=LogisticRegression(),
                                       param_grid={'C': [1.0], 'fit_intercept': [True],
                                                   'l1_ratio': [0.2, 0.4, 0.6, 0.8],
                                                   'penalty': ['l1', 'l2', 'elasticnet',
                                                               'none'],
                                                   'solver': ['newton-cg', 'lbfgs',
                                                              'liblinear', 'sag',
                                                              'saga'],
                                                   'tol': [0.01, 0.001, 0.0001]}))])
```

```
In [39]: print("Best parameters set:",clf.best_params_)
         print("Grid scores on every set of parameters:")
         print()
         means = clf.cv_results_['mean_test_score']
         stds = clf.cv_results_['std_test_score']
         # for mean, std, params in zip(means, stds, clf.cv_results_['params']):
         #     print("%0.3f (+/-%0.04f) for %r"
         #                % (mean, std * 2, params))


         print()
         print("Classification report:")
         y_pred = model.predict(X_test)
         print(classification_report(y_test, y_pred))
         print("Test accuracy:",accuracy_score(y_test, y_pred))
         labels = model.classes_
         matrix = confusion_matrix(y_test,y_pred)
         print(pd.DataFrame(matrix,columns=labels, index=labels))
         plot_confusion_matrix(matrix,labels)

         Best parameters set: {'C': 1.0, 'fit_intercept': True, 'l1_ratio': 0.2, 'penalty': 'l2', 'solver': 'saga', 'tol': 0.01}
         Grid scores on every set of parameters:


         Classification report:
                       precision    recall  f1-score   support

             negative       0.80      0.94      0.86      4634
              neutral       0.64      0.43      0.51      1510
             positive       0.80      0.57      0.67      1176

             accuracy                           0.77      7320
            macro avg       0.74      0.65      0.68      7320
         weighted avg       0.76      0.77      0.76      7320

         Test accuracy: 0.774863387978142
                   negative  neutral  positive
         negative      4346      222        66
         neutral        753      651       106
         positive       352      149       675
```



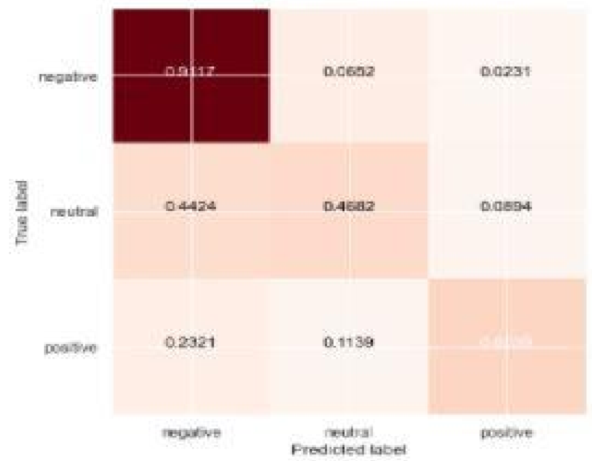
c. **Support Vector Machines** is the third type of classifier is the third type of classifier we have used for predicting sentiment. For tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix. We have tried two different kernels 'linear' and 'rbf'. Linear kernel has given the more optimum result with a test accuracy of 78% which is an improvement but not much as compared to previous Logistic Regression. The we have plotted our result using the function plot_confusion_matrix defined before.

Support Vector Machine

In [40]:

```
parameters = [{'kernel': ['linear'], 'C': [1, 10, 100]},\
              {'kernel': ['rbf'], 'gamma': [1e-2,1e-3, 1e-4],'C': [1, 10, 100, 1000]}]

tfidf=TfidfVectorizer(stop_words='english')
clf=GridSearchCV(SVC(),parameters,cv=2,refit=True)
model= make_pipeline(tfidf,clf)
model.fit(X_train, y_train)


print("Best parameters set:",clf.best_params_)
print("Grid scores on every set of parameters:")
print()
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    print("%0.3f (+/-%0.04f) for %r"
              % (mean, std * 2, params))


print()
print("Classification report:")
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Test accuracy:",accuracy_score(y_test, y_pred))
labels = model.classes_
matrix = confusion_matrix(y_test,y_pred)
print(pd.DataFrame(matrix,columns=labels, index=labels))
plot_confusion_matrix(matrix,labels)
```

```
Best parameters set: {'C': 1, 'kernel': 'linear'}
Grid scores on every set of parameters:

0.754 (+/-0.0077) for {'C': 1, 'kernel': 'linear'}
0.726 (+/-0.0046) for {'C': 10, 'kernel': 'linear'}
0.708 (+/-0.0216) for {'C': 100, 'kernel': 'linear'}
0.621 (+/-0.0000) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
0.621 (+/-0.0000) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.621 (+/-0.0000) for {'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
0.672 (+/-0.0046) for {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
0.621 (+/-0.0000) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.621 (+/-0.0000) for {'C': 10, 'gamma': 0.0001, 'kernel': 'rbf'}
0.747 (+/-0.0041) for {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
0.673 (+/-0.0055) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.621 (+/-0.0000) for {'C': 100, 'gamma': 0.0001, 'kernel': 'rbf'}
0.717 (+/-0.0063) for {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
0.747 (+/-0.0052) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.673 (+/-0.0057) for {'C': 1000, 'gamma': 0.0001, 'kernel': 'rbf'}

Classification report:
              precision    recall  f1-score   support

    negative       0.82      0.91      0.86      4634
     neutral       0.62      0.47      0.53      1510
    positive       0.76      0.65      0.70      1176

    accuracy                           0.78      7320
   macro avg       0.73      0.68      0.70      7320
weighted avg       0.77      0.78      0.77      7320

Test accuracy: 0.7788251366120219
          negative  neutral  positive
negative      4225      302       107
neutral        668      707       135
positive       273      134       769
```



d. The last model we have used is **Random Forest** for testing and classifying our test dataset. We have tried different values of max_features and different values for estimators. Best result is given when max_features are 10 and n_estimators are 250. For tf-idf we are training the data in it and calculating the accuracy score with the confusion matrix.

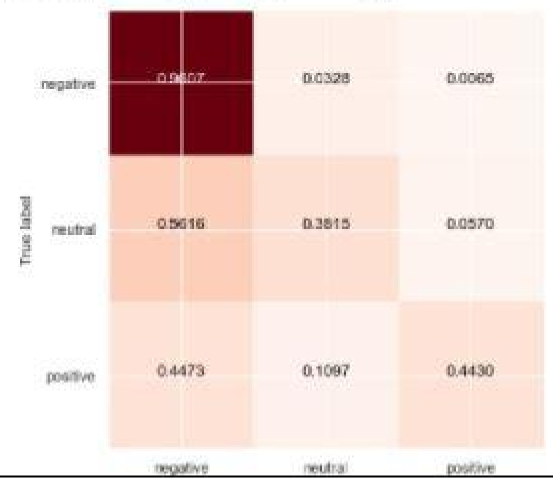
We got an accuracy of 76% on our test data set which is not an improvement from the previous two models having accuracy of 77% and 78%. But it performs much better than Naive Bayes Classifier.Then we have plotted the data using our function plot_confusion_matrix defined before.

Random Forest Classifier

In [41]:

```
parameters = {"max_depth": [None],"max_features": [1,3,10],
              "criterion": ["gini", "entropy"],"n_estimators":[50,250,300]}
tfidf=TfidfVectorizer(stop_words='english')
clf=GridSearchCV(RandomForestClassifier(),param_grid=parameters,cv=2,refit=True)
model= make_pipeline(tfidf,clf)
model.fit(X_train, y_train)



print("Best parameters set:",clf.best_params_)
print("Grid scores on every set of parameters:")
print()
means = clf.cv_results_['mean_test_score']
stds = clf.cv_results_['std_test_score']
for mean, std, params in zip(means, stds, clf.cv_results_['params']):
    print("%0.3f (+/-%0.03f) for %r"
          % (mean, std * 2, params))


print()
print("Classification report:")
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Test accuracy:",accuracy_score(y_test, y_pred))
labels = model.classes_
matrix = confusion_matrix(y_test,y_pred)
print(pd.DataFrame(matrix,columns=labels, index=labels))
plot_confusion_matrix(matrix,labels)
```

```
Best parameters set: {'criterion': 'entropy', 'max_depth': None, 'max_features': 10, 'n_estimators': 250}
Grid scores on every set of parameters:

0.716 (+/-0.002) for {'criterion': 'gini', 'max_depth': None, 'max_features': 1, 'n_estimators': 50}
0.721 (+/-0.002) for {'criterion': 'gini', 'max_depth': None, 'max_features': 1, 'n_estimators': 250}
0.723 (+/-0.005) for {'criterion': 'gini', 'max_depth': None, 'max_features': 1, 'n_estimators': 300}
0.714 (+/-0.001) for {'criterion': 'gini', 'max_depth': None, 'max_features': 3, 'n_estimators': 50}
0.723 (+/-0.003) for {'criterion': 'gini', 'max_depth': None, 'max_features': 3, 'n_estimators': 250}
0.719 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'max_features': 3, 'n_estimators': 300}
0.721 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'max_features': 10, 'n_estimators': 50}
0.722 (+/-0.004) for {'criterion': 'gini', 'max_depth': None, 'max_features': 10, 'n_estimators': 250}
0.723 (+/-0.007) for {'criterion': 'gini', 'max_depth': None, 'max_features': 10, 'n_estimators': 300}
0.717 (+/-0.004) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 1, 'n_estimators': 50}
0.720 (+/-0.005) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 1, 'n_estimators': 250}
0.720 (+/-0.003) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 1, 'n_estimators': 300}
0.724 (+/-0.016) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 3, 'n_estimators': 50}
0.721 (+/-0.004) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 3, 'n_estimators': 250}
0.722 (+/-0.003) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 3, 'n_estimators': 300}
0.720 (+/-0.007) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 10, 'n_estimators': 50}
0.724 (+/-0.007) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 10, 'n_estimators': 250}
0.724 (+/-0.007) for {'criterion': 'entropy', 'max_depth': None, 'max_features': 10, 'n_estimators': 300}

Classification report:
              precision    recall  f1-score   support

    negative       0.76      0.96      0.85      4634
     neutral       0.67      0.38      0.49      1510
    positive       0.82      0.44      0.57      1176

    accuracy                           0.76      7320
   macro avg       0.75      0.60      0.64      7320
weighted avg       0.75      0.76      0.73      7320

Test accuracy: 0.7580601092896175
          negative  neutral  positive
negative      4452      152        30
neutral        848      576        86
positive       526      129       521
```



6. Results and discussion

We have used 4 different machine learning models for predicting sentiment of tweets on our test dataset. These models have given different accuracies on our testing dataset. They are as follows :

1. Naive Bayes Classifier : 68%

2. Logistic Regression : 77%

3. Support Vector Machine : 78%

4. Random Forest Classifier : 76%

The most accurate model with 78 % accuracy is SVM with TFIDFVectorizer.

The Naive Bayes Classifier gives us the least accuracy of 68% so we can't use that model for the prediction purpose. The other three models have given pretty good accuracy >75% so they can be used for sentiment prediction purpose.

The prediction process can be furthurly improved in future to achieve an accuracy greater than 80% by using different classifiers or just modifying and tuning the hyper-parameters of the existing models more finely. We can also try different other techniques to improve the performance of our existing model.

1. Logistic Regression :

Logistic Regression has given an accuracy of 77% when its parameters are set as follows :

C : 1.0,

fit_intercept : True,

L1_ratio : 0.2,

Penalty : L2,

Solver : saga,

Tolerance : 0.01,

In case of Logistic Regression with the above tuned parameters, the F1 score of negative, neutral and positive tweets are 0.86, 0.51 and 0.67 respectively.

2. Random Forest Classifier :

Random Forest Classifier has given a slightly better accuracy of 78% when its parameters are set as follows :

Criterion : entropy,

Max_depth : None,

Max_features : 10,

N_estimators : 250,

The Random Forest Classifier with the above tuned parameters has the F1 score for negative, neutral and positive tweets as .85, .49 and .57 respectively.

3. Support Vector Classifier :

The Support Vector Classifier has given the best accuracy of 78% when its parameters are set as follows :

C : 1,

Kernel : Linear

The Support Vector Classifier with the above tuned parameters has the F1 score for negative, neutral and positive tweets as .86, .53 and .70 respectively.

Online Link of our Project

The complete link of the project along with the dataset can be find out at the following GitHub repository of our team member Kaustubh Dwivedi (Github Username : onlykingKD).

https://github.com/onlykingKD/Twittern-Airlines-sentiment-Analysis

7. References:

[1] E. Cambria, S. Poria, A. Gelbukh and M. Thelwall, "Sentiment Analysis Is a Big Suitcase," in *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 74-80, November/December 2017, doi: 10.1109/MIS.2017.4531228.

[2] D. Cenni, P. Nesi, G. Pantaleo and I. Zaza, "Twitter vigilance: A multi-user platform for cross-domain Twitter data analytics, NLP and sentiment analysis," 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 2017, pp. 1-8, doi: 10.1109/UIC-ATC.2017.8397589.

[3] maran, S.Muthuku & esh, P.Sur. (2017). Text Analysis for Product Reviews for Sentiment Analysis using NLP Methods. International Journal of Engineering Trends and Technology. 47. 474-480. 10.14445/22315381/IJETT-V47P278.

[4] Detecting Emotion in Human Speech (2011), Alex Mordkovich , Kelly Veit , Daniel Zilber ,stanford university

[4] SENTIMENT' BASED MODEL FOR REPUTATION SYSTEMS N AMAZON (2011), Milad Sharif ,Soheil Norouzi, stanford university

[6] Victoria Ikoro, Maria Sharmina, Khaleel Malik, and Riza Batista-Navarro : Analyzing Sentiments Expressed on Twitter by UK Energy Company Consumers. 2018

[7] A Machine Learning based Framework for Sentiment Classification: Indian Railways Case Study (IJITEE ISSN: 2278- 3075, Volume-8 Issue-4, February 2019)

[8] A Survey: Sentiment Analysis Using Machine Learning Techniques for Social Media Analytics (IJPAM InternationalJournal of Pure and Applied Mathematics)