AI_PHASE 2 PROJECT

BY

MEENAKSHI B(513421106028)

BE(ECE)

UNIVERSITY COLLEGE OF ENGINEERING OF KANCHIPURAM

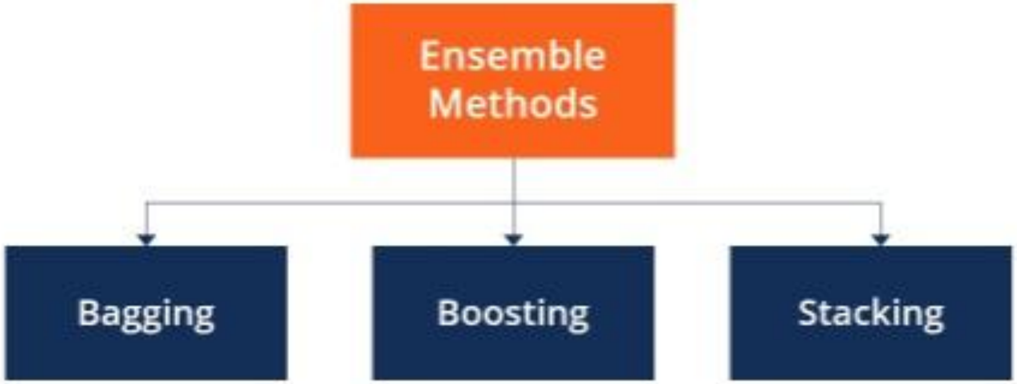
ENSEMBLE METHODS

- Ensemble methods are machine learning techniques that combine multiple models or model instances to improve overall prediction accuracy and robustness.

- Instead of relying on a single model, ensemble methods leverage the outputs of multiple models to make more accurate predictions.

- Ensemble methods aim at improving predictability in models by combining several models to make one very reliable model. The most popular ensemble methods are boosting, bagging, and stacking.

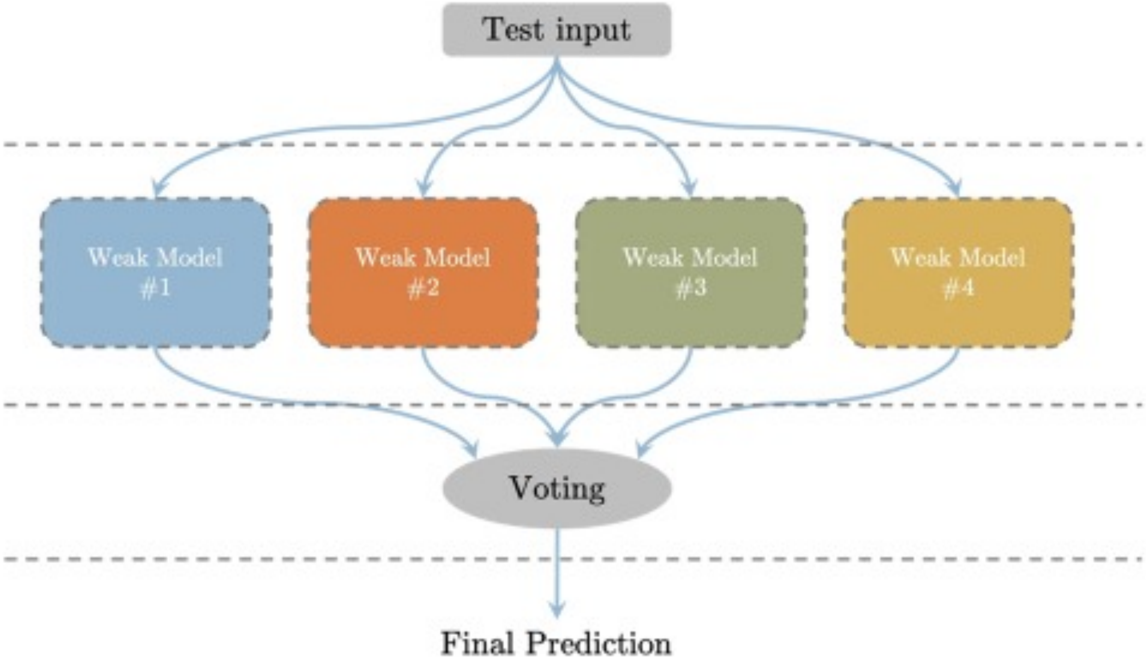
OVERVIEW ENSEMBLES METHODS



Ensemble methods in deep learning are used to improve the performance of neural networks and can take many forms including:

- **Stacking**: Training multiple deep learning models and utilizing the outputs of each model to train a "meta-model", a machine learning model that takes other models' outputs as inputs. The meta-model takes the base model predictions as inputs and learns how to best combine them to make the final prediction. This approach can enhance the model's predictive power and capture complex relationships in the data.
- **Bagging**: Training multiple instances of the same model on different subsets of data and combining the model outputs through averaging or voting. This approach can improve the model's generalizability.
- **Model Averaging:** Independently training multiple instances of the same deep learning model with different initializations (the initial values of the parameters or weights of a model before training), and averaging the model outputs to obtain a final prediction. This approach can reduce the impact of varying initializations among models and provide more stable predictions.
- **Boosting,** a very common ensemble method in classical machine learning is not prevalent in deep learning. Boosting entails combining weaker machine learning models, such as decision trees in classical machine learning, to create a single strong model. While there are some recent examples of boosting in deep learning, deep learning models are often capable of achieving high accuracy without the need for boosting.

FLOW CHART



Test input
Weak Model #1
Weak Model #2
Weak Model #3
Weak Model #4
Voting
Final Prediction

DATASET:

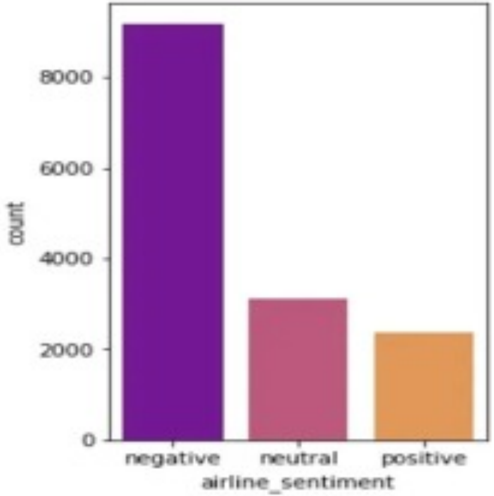
Dataset Link: https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

tweet_id	airline_sen	airline_sen	negativere	negativere	airline	airline_sen	name	negativere	retwee
5.7E+17	neutral	1			Virgin America		cairdin		
5.7E+17	positive	0.3486		0	Virgin America		jnardino		
5.7E+17	neutral	0.6837			Virgin America		yvonnalynn		
5.7E+17	negative	1	Bad Flight	0.7033	Virgin America		jnardino		
5.7E+17	negative	1	Can't Tell	1	Virgin America		jnardino		
5.7E+17	negative	1	Can't Tell	0.6842	Virgin America		jnardin		
5.7E+17	positive	0.6745		0	Virgin America		cjmcgi		
5.7E+17	neutral	0.634			Virgin America		pilot		
5.7E+17	positive	0.6559			Virgin America		dhepburn		
5.7E+17	positive	1			Virgin America		YupitsTate		
5.7E+17	neutral	0.6769		0	Virgin America		idk_but_youtube		
5.7E+17	positive	1			Virgin America		HyperCamiLax		
5.7E+17	positive	1			Virgin America		HyperCamiLax		
5.7E+17	positive	0.6451			Virgin America		mollanderson		
5.7E+17	positive	1			Virgin America		sjespers		
5.7E+17	negative	0.6842	Late Flight	0.3684	Virgin America		smartwatermelon		
5.7E+17	positive	1			Virgin America		ItzBrianHunty		
5.7E+17	negative	1	Bad Flight	1	Virgin America		heatherovieda		
5.7E+17	positive	1			Virgin America		thebrandiray		
5.7E+17	positive	1			Virgin America		JNLpierce		
5.7E+17	negative	0.6705	Can't Tell	0.3614	Virgin America		MISSGJ		
5.7E+17	positive	1			Virgin America		DT_Les		
5.7E+17	positive	1			Virgin America		ElvinaBeck		
5.7E+17	neutral	1			Virgin America		rjlynch21086		
5.7E+17	negative	1	Customer	0.3557	Virgin America		ayeevickiee		
5.7E+17	negative	1	Customer	1	Virgin America		Leora13		
5.7E+17	negative	1	Can't Tell	0.6614	Virgin America		meredithjlynn		
5.7E+17	neutral	0.6854			Virgin America		AdamSinger		
5.7E+17	negative	1	Bad Flight	1	Virgin America		blackjackpro911		
5.7E+17	neutral	0.615		0	Virgin America		TenantsUpstairs		
5.7E+17	negative	1	Flight Boo	1	Virgin America		jordanpichler		
5.7E+17	neutral	1			Virgin America		JCervantezzz		
5.7E+17	negative	1	Customer	1	Virgin America		Cuschoolie1		
5.7E+17	negative	1	Customer	1	Virgin America		amandabmoriarty		
5.7E+17	positive	1			Virgin America		NorthTahomeTeam		
5.7E+17	neutral	0.6207			Virgin America		miaerolinea		

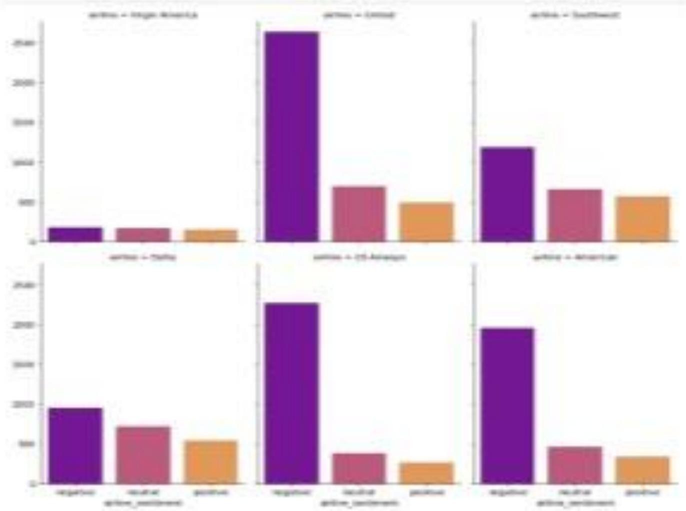
- tweets = pd.read_csv('Tweets.csv')
- Let’s look at features included in dataset:
- tweets.head()
- tweets.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639
Data columns (total 15 columns):
tweet_id                        14640 non-null int64
airline_sentiment               14640 non-null object
airline_sentiment_confidence    14640 non-null float64
negativereason                  9178 non-null object
negativereason_confidence       10522 non-null float64
airline                         14640 non-null object
airline_sentiment_gold          40 non-null object
name                            14640 non-null object
negativereason_gold             32 non-null object
retweet_count                   14640 non-null int64
text                            14640 non-null object
tweet_coord                     1019 non-null object
tweet_created                   14640 non-null object
tweet_location                  9907 non-null object
user_timezone                   9820 non-null object
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

- plt.figure(figsize=(3,5))
  sns.countplot(tweets['airline_sentiment’],
- order=tweets.airline_sentiment.value_counts().index,palette='plasma')
  plt.show()



- g = sns.FacetGrid(tweets, col=”airline”, col_wrap=3, height=5, aspect =0.7) g = g.map(sns.countplot, “airline_sentiment”,order =tweets.airline_sentiment.value_counts().index, palette=’plasma’) plt.show()



- To do sentiment analysis, we need to import a few libraries. Since this is a classification problem, I use LGBMClassifier.

- from lightgbm import LGBMClassifier
- We need to convert these tweets (texts) to a matrix of token counts.

- from sklearn.feature_extraction.text import CountVectorizer
- The next step is to normalize the count matrix using tf-idf representation.

- from sklearn.feature_extraction.text import TfidfTransformer
- I used the pipeline function to do all steps together.

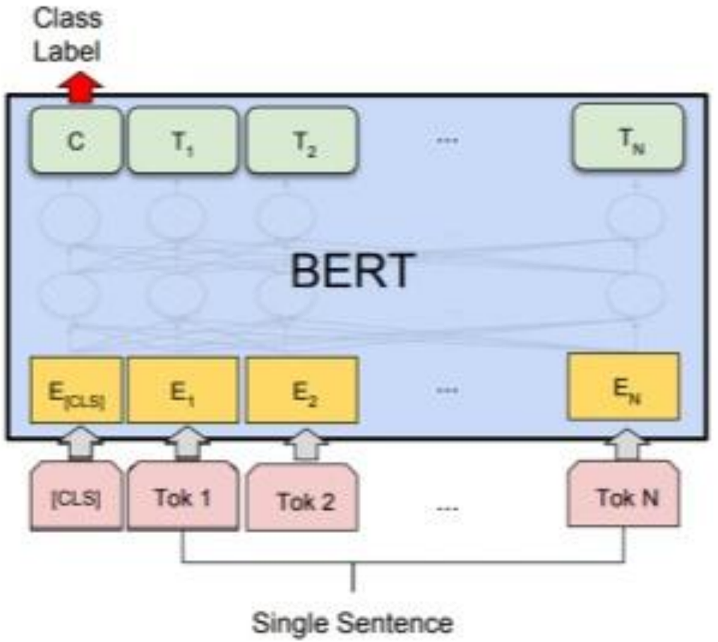
- twitter_sentiment = Pipeline([('CVec', CountVectorizer(CountVectorizer(stop_words='english'))),
- ('Tfidf', TfidfTransformer()),
- ('norm', Normalizer()),
- ('tSVD', TruncatedSVD(n_components=100)),
- ('lgb', LGBMClassifier(n_jobs=-1))])
- In the end, CROSS_VALIDATE is used with ROC_AUC metrics.

- %%time
- cv_pred = cross_validate(twitter_sentiment,
- tweets['text'],
- tweets['airline_sentiment'],
- cv=5,
- scoring=('roc_auc_ovr'))
- The results we have measured using ROC_AUS are as follows.

Bidirectional Representation for Transformers (BERT)

- BERT is a powerful technique for natural language processing that can improve how well computers comprehend human language. The foundation of BERT is the idea of exploiting bidirectional context to acquire complex and insightful word and phrase representations. By simultaneously examining both sides of a word's context, BERT can capture a word's whole meaning in its context, in contrast to earlier models that only considered the left or right context of a word.
- This enables BERT to deal with ambiguous and complex linguistic phenomena including polysemy, co-reference, and long-distance relationships.
- For that, the paper also proposed the architecture of different tasks. In this post, we will be using BERT architecture for Sentiment classification tasks specifically the architecture used for the CoLA (Corpus of Linguistic Acceptability) binary classification task.

SINGLE SENTENCE CLASSIFICATION TASK



Step 1: Import the necessary libraries

- PROGRAM:
- import os
- import shutil
- import tarfile
- import tensorflow as tf
- from transformers import BertTokenizer, TFBertForSequenceClassification
- import pandas as pd
- from bs4 import BeautifulSoup
- import re
- import matplotlib.pyplot as plt
- import plotly.express as px
- import plotly.offline as pyo
- import plotly.graph_objects as go
- from wordcloud import WordCloud, STOPWORDS
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import classification_report

Step 2: Load the dataset

- # Get the current working directory
- current_folder = os.getcwd()

- dataset = tf.keras.utils.get_file(
-         fname ="aclImdb.tar.gz",
-         origin ="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz",
-         cache_dir= current_folder,
-         extract = True)

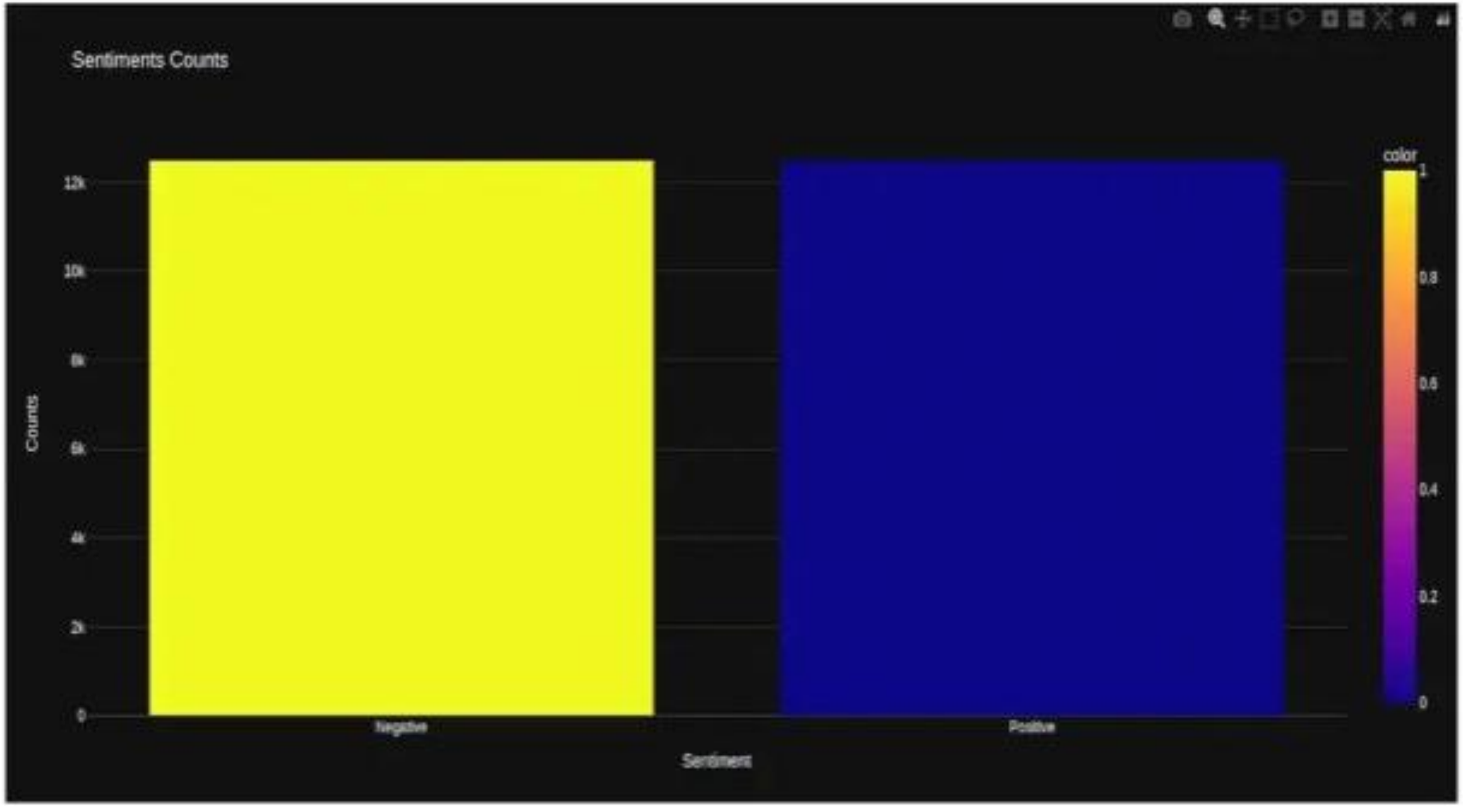
- Output:

- ['aclImdb.tar.gz', 'aclImdb']

Step 3: Preprocessing

```
sentiment_counts = train_df['sentiment'].value_counts()

fig =px.bar(x= {0:'Negative',1:'Positive'},
                              y= sentiment_counts.values,
                              color=sentiment_counts.index,
                              color_discrete_sequence =
px.colors.qualitative.Dark24,
                              title='<b>Sentiments Counts')
```


Sentiments Counts
12k
10k
8k
6k
4k
2k
0
Counts
Negative
Positive
Sentiment
color
1
0.8
0.6
0.4
0.2
0

CONCLUSION:

Sentiment analysis deals with the classification of texts based on the sentiments they contain. Thisarticle focuses on a typical sentiment analysis model consisting of three core steps, namely datapreparation, review analysis and sentiment classification, and describes representative techniquesinvolved in those steps.