**NLP PROJECT**

*Title*

**Movie Review Based on Real Time Sentiment Analysis of Twitter posts using Ensemble Technique on Naïve Bayes, SVC and Random Forest Classifier Algorithms**

*This is submitted as Assignment for the consideration*

*Of the Internal marks of the course Natural Language Processing*

Submitted by

***Madireddy Kalyan Venkat***

***Regno.122003141***

***Contact no. 6385884778***

***Email:madireddykalyanvenkat@gmail.com***

**JUNE 2021**



**SCHOOL OF COMPUTING**

**THANJAVUR, TAMIL NADU, INDIA – 613 401**

**BASE PAPER**

Here, I’m using three base paper ideas in a fair amount to do this project. I implemented the first [1] paper almost full, in the [2] paper I took the idea of NLP-NLTK Pipeline process of pre-processing and cleaning and in the [3] paper I took the idea of Using Multi ML models and I applied the Ensemble technique to obtain the result.

**Citation-**

[1] V. Prakruthi, D. Sindhu and D. S. Anupama Kumar, "Real Time Sentiment Analysis Of Twitter Posts," 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), 2018, pp. 29-34, doi:10.1109/CSITSS.2018.8768774.URL:https://ieeexplore.ieee.org/document/8768774

[2] M. R. Hasan, M. Maliha and M. Arifuzzaman, "Sentiment Analysis with NLP on Twitter Data," 2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2), 2019, pp. 1-4, doi: 10.1109/IC4ME247184.2019.9036670. URL:https://ieeexplore.ieee.org/document/9036670

[3] A. Pimpalkar and R. J. R. Raj, "Evaluation of Tweets for Content Analysis Using Machine Learning Models," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), 2020, pp. 454-459, doi: 10.1109/CICN49253.2020.9242611. URL:https://ieeexplore.ieee.org/document/9242611

**Abstract**

In this project, I Performed Real time movie analysis using Twitter posts. These posts were collected at the moment by using the Twitter API connection. The connection is established using some secure keys of one’s account with python. The collected data is then converted in to data frame using pandas and then the description of tweets is Pre-processed using NLTK tools like sentence, word tokenizer, lemmatizer, spell checker, stop words remover and others. This pre-processed tweets are then cleaned of null values and symbols and other irrelevant characters. Later this description is converted in to bag of words form using sparse matrix. The entire data is converted in to train and test, and the trained data is passed to 3 algorithms naïve Bayes, SVC and RFC. This trained models are then used to predict test data and then all the 3 results were used in ensemble technique to identify the actual Polarity

**Source code**

**Note-**  All this code is done in Jupyter Note book in python. The code here is pasted after converting the .ipynb file to .py file format, for better valuation ipynb on jupyter note book is preffered, so Apart from pasting the code, I also uploaded the .ipynb and also .py file in this drive link- <https://drive.google.com/drive/folders/1AUcydR6TotuQti9QFhgQETH11F2NWNN-?usp=sharing>

CODE

#!/usr/bin/env python

# coding: utf-8

# In[ ]:

# # Extracting the live Tweets from Twitter API

#

# Importing necessary packages

# In[1]:

import os

import tweepy as tw

import pandas as pd

import nltk

# In[2]:

#twitter API connection

access\_token= '1228242149247860742-LvB1ZifktBaTciJKmLcZIsomsHIaZY'

access\_token\_secret= 'gvKiUsttlEYazbPf7pUs2FzhISvJreOv6KhzLeNS9isP9'

consumer\_key= 'GGP3RRXNZEr4xeQYuEXMsiLH1'

consumer\_secret= '1nBtRP9LMUFEGv0eO0SvVSox1ak2rLchz55rQtE2SPF3hog72P'

# In[3]:

auth = tw.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

api = tw.API(auth)

# Give the MOVIE NAME(you want to check)

# In[10]:

movie\_name=input("Enter the Movie name ")

movie\_name\_with\_hashtag='#'+movie\_name

# In[11]:

search\_words = movie\_name\_with\_hashtag

date\_since = "2019-04-28"

# In[12]:

#Extracting the 1000 tweets of the entered movie name

tweets = tw.Cursor(api.search,

q=search\_words,

lang="en",

since=date\_since).items(1000)

# In[13]:

users\_locs = [[tweet.user.id, tweet.user.screen\_name, tweet.user.location, tweet.user.url, tweet.user.description, tweet.user.protected, tweet.user.verified, tweet.user.followers\_count, tweet.user.favourites\_count, tweet.user.friends\_count,tweet.user.statuses\_count] for tweet in tweets]

# In[19]:

#conveting the extracted tweeets in to data frame using pandas

df= pd.DataFrame(data=users\_locs,

columns=['id','user', 'location', 'url', 'description', 'protected', 'verified', 'followers\_count', 'favourites\_count', 'friends\_count','total\_tweets\_count'])

df.head()

# In[20]:

#the data set shape

df.shape

# In[21]:

#removing the null descriptive rows if any

df=df[df['description'].isna()==False]

df.shape

# # preprocessing the data

# (1)Removing the unnecessary characters

#

# In[23]:

#importing the necessary packages

from nltk.tokenize import sent\_tokenize,word\_tokenize

from nltk.corpus import stopwords

import re

# In[24]:

clean\_1=[]

for each\_row in range(df.shape[0]):

text=df['description'].values[each\_row]

tempo=str(text)

tempe=re.sub('[^A-Za-z0-9!?]', ' ', tempo)

clean\_1.append(tempe)

df["cleaned\_1"]=clean\_1

# In[25]:

df[['cleaned\_1']]

# (2).Converting the cleaned\_1 description of tweets in to lower case

# In[26]:

clean\_2=[]

for each\_row in range(df.shape[0]):

text=df['cleaned\_1'].values[each\_row]

tempo=str(text)

tempe=tempo.lower()

clean\_2.append(tempe)

df["cleaned\_2"]=clean\_2

# In[27]:

df[['cleaned\_2']]

# (3).Removing the stop words

# In[28]:

#importing necessary packages

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize,sent\_tokenize

stopwords\_list=stopwords.words('english')

# In[29]:

#tokenizing and remocving stop words

mlist=[]

for each\_row in range(df.shape[0]):

text=df['cleaned\_2'].values[each\_row]

tokenized\_words=word\_tokenize(text)

clean\_words=[]

for each\_word in tokenized\_words:

if not each\_word in stopwords\_list:

clean\_words.append(each\_word)

dre=''

for each\_data in clean\_words:

dre=dre+str(each\_data)+' '

mlist.append(dre)

df['cleaned\_3']=mlist

# In[30]:

df[['cleaned\_3']]

# (4).converting all the words in to their root words

# In[32]:

#importing the necessary\_packages

from nltk.stem import WordNetLemmatizer

lm=WordNetLemmatizer()

# In[33]:

#tokenizing and converting to lem words

clean\_4=[]

for each\_row in range(df.shape[0]):

text=df['cleaned\_3'].values[each\_row]

tokenized\_words=word\_tokenize(text)

clean\_words=[]

for each\_word in tokenized\_words:

clean\_words.append(lm.lemmatize(each\_word))

dre=''

for each\_data in clean\_words:

dre=dre+str(each\_data)+' '

clean\_4.append(dre)

df['cleaned\_4']=clean\_4

# In[34]:

df[['cleaned\_4']]

# (5).Removing words that are not there in english dictionary

# In[37]:

#importing the necessary packages

from nltk.corpus import words

word\_list = words.words()

# In[38]:

#tokenizing and removing the words not there in dictionary

clean\_5=[]

for each\_row in range(df.shape[0]):

text=df['cleaned\_4'].values[each\_row]

tokenized\_words=word\_tokenize(text)

clean\_words=[]

for each\_word in tokenized\_words:

if each\_word in word\_list:

clean\_words.append(each\_word)

dre=''

for each\_data in clean\_words:

dre=dre+str(each\_data)+' '

clean\_5.append(dre)

df['cleaned\_5']=clean\_5

# In[39]:

df[['cleaned\_5']]

# # Cleaning the Data Set

# Removing the null value cleaned description rows

# In[47]:

df=df[df['cleaned\_5']!='']

# In[50]:

#checking shape after removing

df.shape

# using VADER for pre-analysis

# In[53]:

#import necessaty packages

from nltk.sentiment import SentimentIntensityAnalyzer

sia = SentimentIntensityAnalyzer()

# In[54]:

tra\_list=[]

for each\_row in range(df.shape[0]):

text=df['cleaned\_5'].values[each\_row]

Ana\_dict=sia.polarity\_scores(text)

neg\_amount=Ana\_dict['neg']

neu\_amount=Ana\_dict['neu']

pos\_amount=Ana\_dict['pos']

if (max(neg\_amount,neu\_amount,pos\_amount)==neg\_amount):

val='negative'

elif (max(neg\_amount,neu\_amount,pos\_amount)==pos\_amount):

val='positive'

else:

val='neutral'

tra\_list.append(val)

df['sentiment']=tra\_list

# In[58]:

df['sentiment']

# In[62]:

df[df['sentiment']=='negative'].shape

# Converting the data set in to almost equal distribution Set for making good ML model prediction

# In[63]:

kn=df[df['sentiment']=='neutral'][:160]

# In[64]:

df=df[df['sentiment']!='neutral']

# In[65]:

df=df.append(kn)

# Removing the Unnecessary rows

# In[69]:

df.columns

# In[70]:

df=df[['user','cleaned\_5','sentiment']]

# In[71]:

df['cleaned\_description']=df['cleaned\_5']

# In[72]:

df.columns

# In[73]:

df=df[['user', 'sentiment', 'cleaned\_description']]

# In[74]:

df.head()

# In[75]:

#shuffling the data frame for more accurate training

df = df.sample(frac = 1)

# In[76]:

df.head()

# # Converting the data in to data for training and data for Sentiment Prediction

# Converting cleaned\_description into Sparse\_matrix(BAG OF WORDS FORM)

# In[78]:

#importing the necessary packages

from sklearn.feature\_extraction.text import CountVectorizer

cv=CountVectorizer()

# In[86]:

sparse\_matrix=cv.fit\_transform(df['cleaned\_description'])

print(sparse\_matrix.shape)

# Encoding the Sentiment in to int format

# In[87]:

#importing the necessary packages

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

# In[90]:

df['sentiment']=le.fit\_transform(df['sentiment'])

df['sentiment'].head()

# Splitting the data for modelling

# In[91]:

#importing the necessary packages

from sklearn.model\_selection import train\_test\_split

# In[92]:

x\_train,x\_test,y\_train,y\_test=train\_test\_split(sparse\_matrix,df['sentiment'])

# In[93]:

x\_train.shape, y\_train.shape, x\_test.shape, y\_test.shape

# # Training Data using Naive Bayes

# In[94]:

#importing the necessary packages

from sklearn.naive\_bayes import MultinomialNB

nb=MultinomialNB()

# In[95]:

nb.fit(x\_train,y\_train)

# # Training Data using Support Vector Clustering

# In[99]:

#importing the necessary packages

from sklearn.svm import SVC

svc=SVC()

# In[100]:

svc.fit(x\_train,y\_train)

# # Training Data using Random Forest Classifier

# In[102]:

#importing the necessary packages

from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier()

# In[104]:

rfc.fit(x\_train,y\_train)

# # Predicting the Accuracy of each Algorithm

# In[106]:

#importing the necessary packages

from sklearn.metrics import accuracy\_score

# Accuracy of Naive Bayes

# In[108]:

y\_pred\_1=nb.predict(x\_test)

# In[109]:

accuracy\_score(y\_pred\_1,y\_test)

# Accuracy of Support Vector Clustering

# In[110]:

y\_pred\_2=svc.predict(x\_test)

# In[111]:

accuracy\_score(y\_pred\_2,y\_test)

# Accuracy of Random Forest Classifier

# In[112]:

y\_pred\_3=rfc.predict(x\_test)

# In[113]:

accuracy\_score(y\_pred\_3,y\_test)

# # Applying Ensembling Technique

# In[122]:

list1=list(y\_pred\_1)

list2=list(y\_pred\_2)

list3=list(y\_pred\_3)

list4=[]

# In[119]:

#importing necessary packages

import matplotlib.pyplot as plt

import numpy as np

# LIST1

# In[123]:

count\_0=0

count\_1=0

count\_2=0

for each in list1:

if each==0:

count\_0=count\_0+1

elif each==1:

count\_1=count\_1+1

elif each==2:

count\_2=count\_2+1

else:

pass

if max(count\_1,count\_2,count\_0)==count\_1:

list4.append(1)

elif max(count\_1,count\_2,count\_0)==count\_2:

list4.append(2)

if max(count\_1,count\_2,count\_0)==count\_0:

list4.append(0)

y\_y = np.array([count\_0,count\_1,count\_2])

mylabels = ["negative", "neutral", "positive"]

plt.pie(y\_y, labels = mylabels)

# LIST2

# In[124]:

count\_0=0

count\_1=0

count\_2=0

for each in list2:

if each==0:

count\_0=count\_0+1

elif each==1:

count\_1=count\_1+1

elif each==2:

count\_2=count\_2+1

else:

pass

if max(count\_1,count\_2,count\_0)==count\_1:

list4.append(1)

elif max(count\_1,count\_2,count\_0)==count\_2:

list4.append(2)

if max(count\_1,count\_2,count\_0)==count\_0:

list4.append(0)

y\_y = np.array([count\_0,count\_1,count\_2])

mylabels = ["negative", "neutral", "positive"]

plt.pie(y\_y, labels = mylabels)

# LIST3

# In[125]:

count\_0=0

count\_1=0

count\_2=0

for each in list3:

if each==0:

count\_0=count\_0+1

elif each==1:

count\_1=count\_1+1

elif each==2:

count\_2=count\_2+1

else:

pass

if max(count\_1,count\_2,count\_0)==count\_1:

list4.append(1)

elif max(count\_1,count\_2,count\_0)==count\_2:

list4.append(2)

if max(count\_1,count\_2,count\_0)==count\_0:

list4.append(0)

y\_y = np.array([count\_0,count\_1,count\_2])

mylabels = ["negative", "neutral", "positive"]

plt.pie(y\_y, labels = mylabels)

# Deciding

# In[128]:

count\_0=0

count\_1=0

count\_2=0

for each in list4:

if each==0:

count\_0=count\_0+1

elif each==1:

count\_1=count\_1+1

elif each==2:

count\_2=count\_2+1

else:

pass

y\_y = np.array([count\_0,count\_1,count\_2])

mylabels = ["negative", "neutral", "positive"]

plt.pie(y\_y, labels = mylabels)

if max(count\_1,count\_2,count\_0)==count\_1:

print('neutral')

elif max(count\_1,count\_2,count\_0)==count\_2:

print('positive')

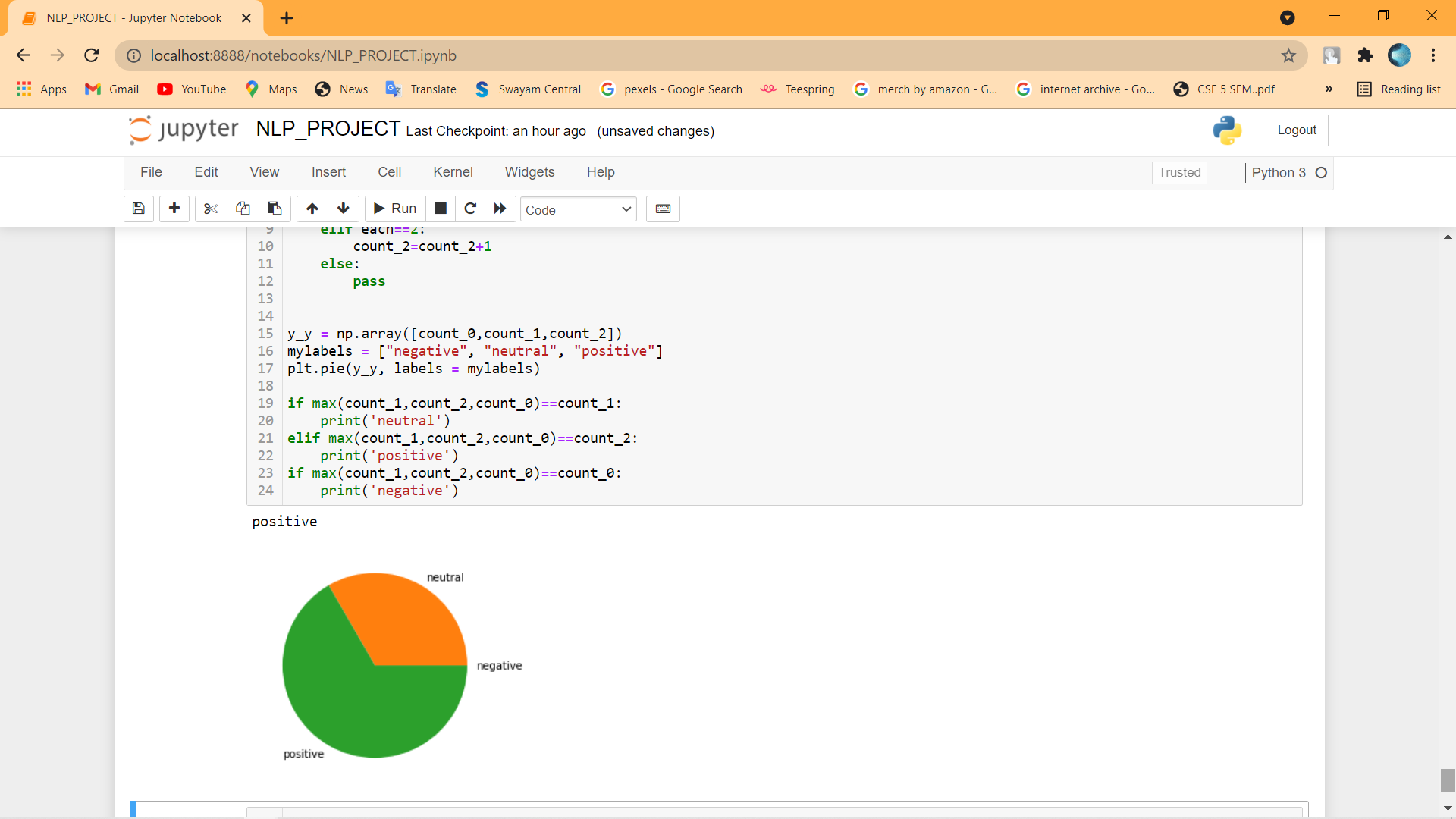
if max(count\_1,count\_2,count\_0)==count\_0:

print('negative')

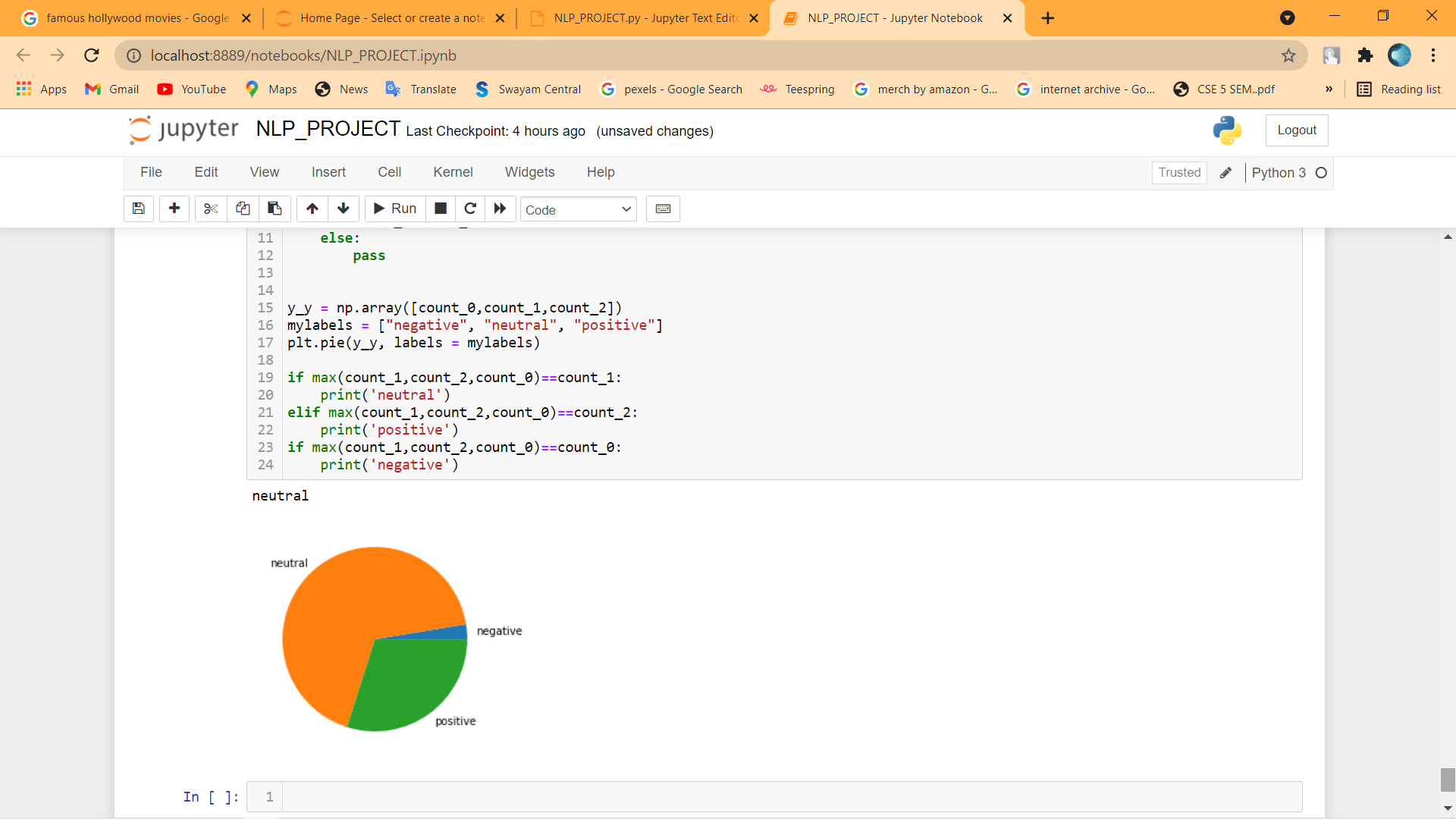
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*code end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

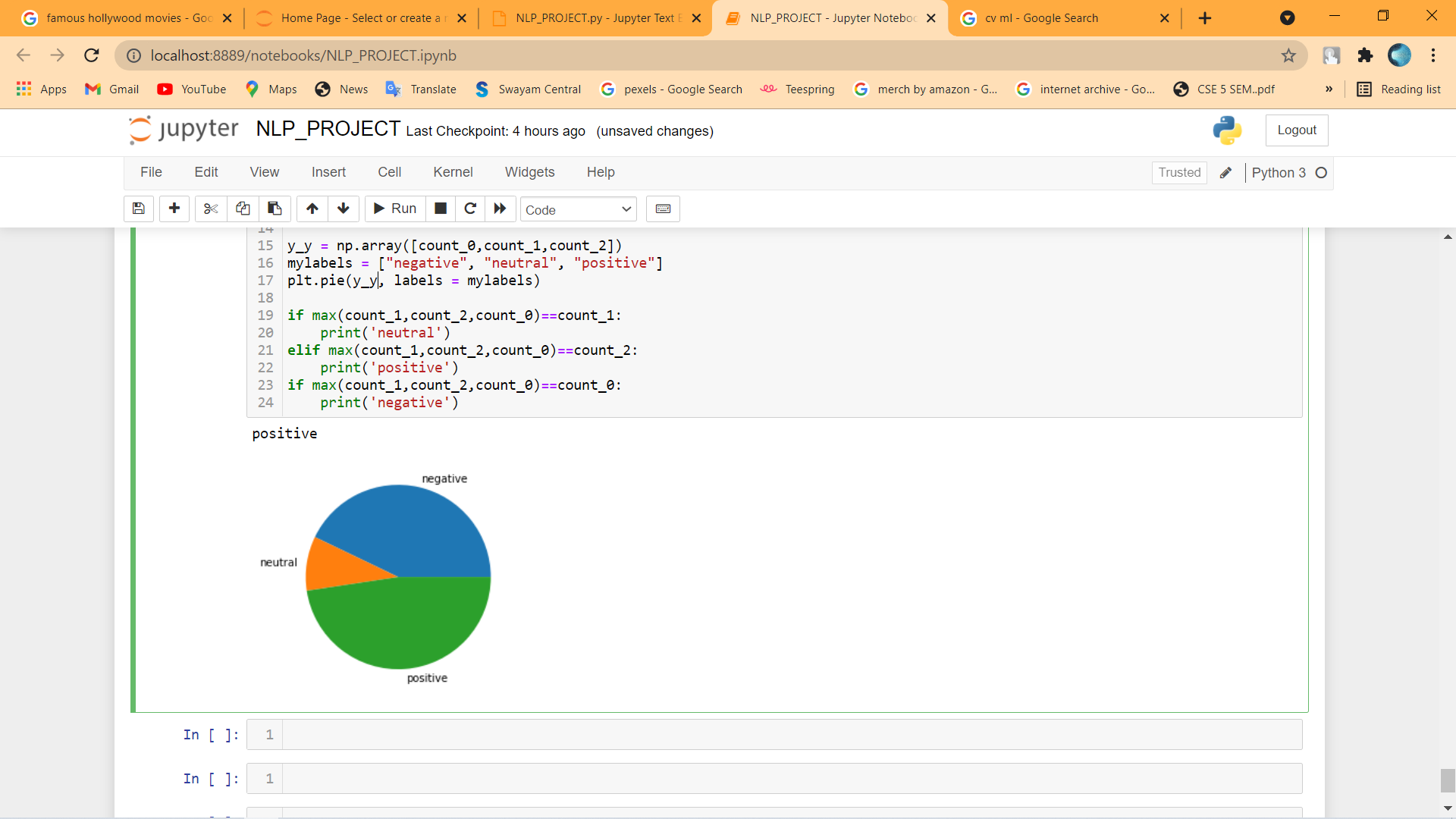
**O/P screen Shots**

1. Screen shots for movie name-**Justice League-**Positive Review ourcome



2) Screen shot for the movie name-**Interstellar-**neutral review



3) Screen shot for the movie **(Fast and furious 9)f9**-positive Review for the movie f9

**CONCLUSION**

The usage of ensemble technique produces more accurate results compared to the usage of single algorithm of machine learning and also the review of the movie may vary most of times and is constant because the outcome depends on the recent 1000 tweets extracted and as we know per minute millions of tweets gets posted, the answer differs. But Twitter API application of usage and it compatibility to handle are in line fantastic. The Bag of words is one of the most common technique used in the sentiment analysis and its implementation using count vectorizer is smooth. And more over this project is a good learning Experience and since NLP is a hot field of tech in present modern day science, I would like to improve my skills and concepts in this area in the future.

Thank you