5.1.2 HARDWARE REQUIREMENTS

Following are the hardware requirements necessary for faster execution of the code.

1.A minimum of Intel Core I3 processor

2.A minimum of 4 GB Ram

5.2 ANALYSIS OF INPUT AND OUTPUT

Tweepy is an easy-to-use Python library for accessing the Twitter API. You need to have a Twitter developer account. Pandas can be used for data cleaning, data inspection, data visualization.

```
!pip install tweepy
import tweepy
import pandas as pd
```

To access twitter API we need to have developer account.Then we will be given the below keys and can be used for authentication and accessing tweets.

```
consumer_key='fRoPrsmLdUUuGM00IASrij5qZ'
consumer_secret='3ZQGJJoWIy150n4Zqa77nud60eCJTAM9wSTmOanY8HAMXKsHcx'
access_token='1519724749303459840-TXNpLMF8wDAzxHYxZRyetlFaUjDZe8'
access_token_secret='xBhlVUgsrjfZ802QWrF8grLvkgYTCLkgpebsaporG7iHx'
```

By using tweepy and passing the above keys we can authenticate our account and access tweets from twitter API.

```
auth=tweepy.OAuthHandler(consumer_key,consumer_secret)
auth.set_access_token(access_token,access_token_secret)
api=tweepy.API(auth)
```

By using home_timeline() method in twitter api we can access the tweets in homeline of an account.

```
public_tweets=api.home_timeline()
for tweet in public_tweets:
  print(tweet.text)
```

```
One of the most massive black holes is putting a spin on the way scientists think black holes interact with their s… https://t.co/TosfQyXvDb
Suck it up, it's almost Prime Day. July 12-13. https://t.co/uIEQs6PcME
Babe, you OK? You haven't touched your Gravity Assist podcast episode about possible diamond rain on Neptune and Ur… https://t.co/eV8ylgN39p
More on how you can make the most out of your Microsoft Teams meetings: https://t.co/K9yCkon3sx
1. Create an agenda 📝
2. Actively participate 🙋
3. Turn your camera on 🎥
4. Keep the group small 👯
5. Share materials beforehand 📲
Ready to make your virtual meetings more engaging and effective?

It's simple with these 5 tips ⬇️
This week @ NASA: A satellite launches to test a new orbit around the Moon, #Cygnus departs the @Space_Station, and… https://t.co/8oKvYz31GC
Ever search something and wonder: "Is someone... somewhere… searching this too?" ✨
Summer nights call for stargazing!

Look out for some of July's celestial events, including the planets of dawn, th… https://t.co/EpjVIwvy4x
```

By using screen_name attribute we can specify a username and access the tweets sent by a particular user.We can also set the number of tweets we should extract. Store the data in a dataframe for better visualization.

```
user='veritasium'
limit=300
tweets=tweepy.Cursor(api.user_timeline,screen_name=user,count=200,tweet_mode="extended").items(limit)
#tweets=api.user_timeline(screen_name=user,count=limit,tweet_mode="extended")
columns=['User','Tweet']
data=[]
for tweet in tweets:
  data.append([tweet.user.screen_name,tweet.full_text])
df=pd.DataFrame(data,columns=columns)
print(df)
```

```
           User                                              Tweet
0    veritasium  @captainspinifex I am curious, which ones do y...
1    veritasium                    Which thumbnail do you prefer?
2    veritasium  Which thumbnail do you prefer?\n(poll below) h...
3    veritasium   @Robin_B Was great to meet you and see your art!
4    veritasium  @christos_markou I can upload an .srt file if ...
..          ...                                               ...
295  veritasium  @SisyphusRedemed @SciencePundit @BillNye A phy...
```

The accessed tweets should be preprocessed for removing unnecessary parts. Data cleaning is done.

```
def cleanTxt(text):
 text = re.sub('@[A-Za-z0-9]+', '', text) #Removing @mentions
 text = re.sub('#', '', text) # Removing '#' hash tag
 text = re.sub('RT[\s]+', '', text) # Removing RT
 text = re.sub('https?:\/\/\S+', '', text) # Removing hyperlink
 text=re.sub(':+','',text) #Removing colon

 return text


# Clean the tweets
df['Tweets'] = df['Tweets'].apply(cleanTxt)

# Show the cleaned tweets
df
```

We can access the tweets that are related to a particular keyword. If we pass a keyword and languages we can access the tweets related to keyword of specified language.

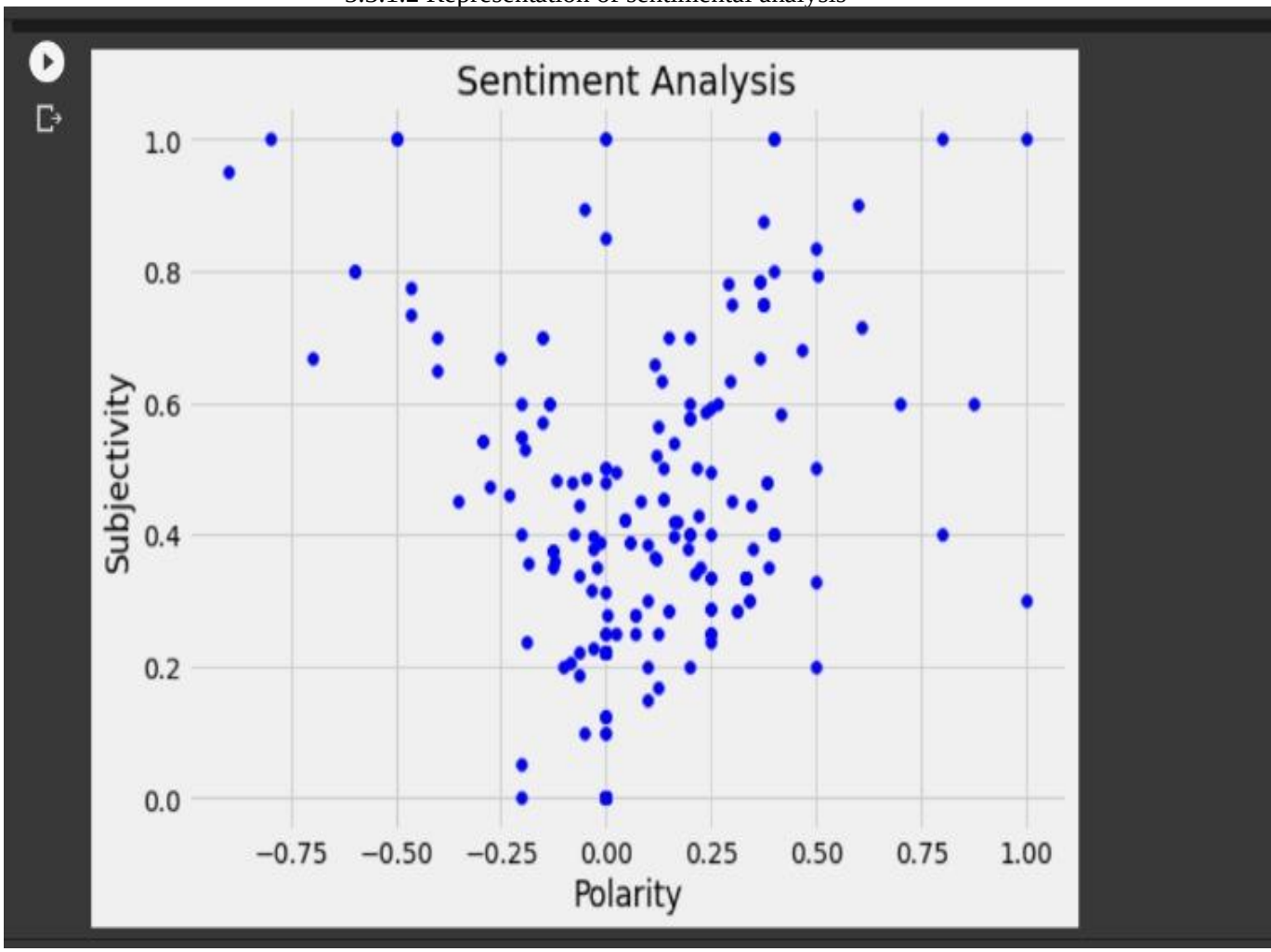
```
import re
keywords="corona"
limit=300
tweets=tweepy.Cursor(api.search,q=keywords,count=100,lang="en",tweet_mode="extended").items(limit)
columns=['Tweets']
data=[]
# def clean(text):
#   return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", text).split()
# for tweet in tweets:
#   a=api.clean(tweet)
#   data.append([a])
for tweet in tweets:
  data.append([tweet.full_text])
df=pd.DataFrame(data,columns=columns)
print(df)
```

```
                                              Tweets
0    RT @The_MaquinaEN: 𝑵𝑬𝑾 𝑬𝑹𝑨 \nIt truly feels th...
1    RT @MrPatNguyen: Dusk till dawn.\n\n#workfromh...
```

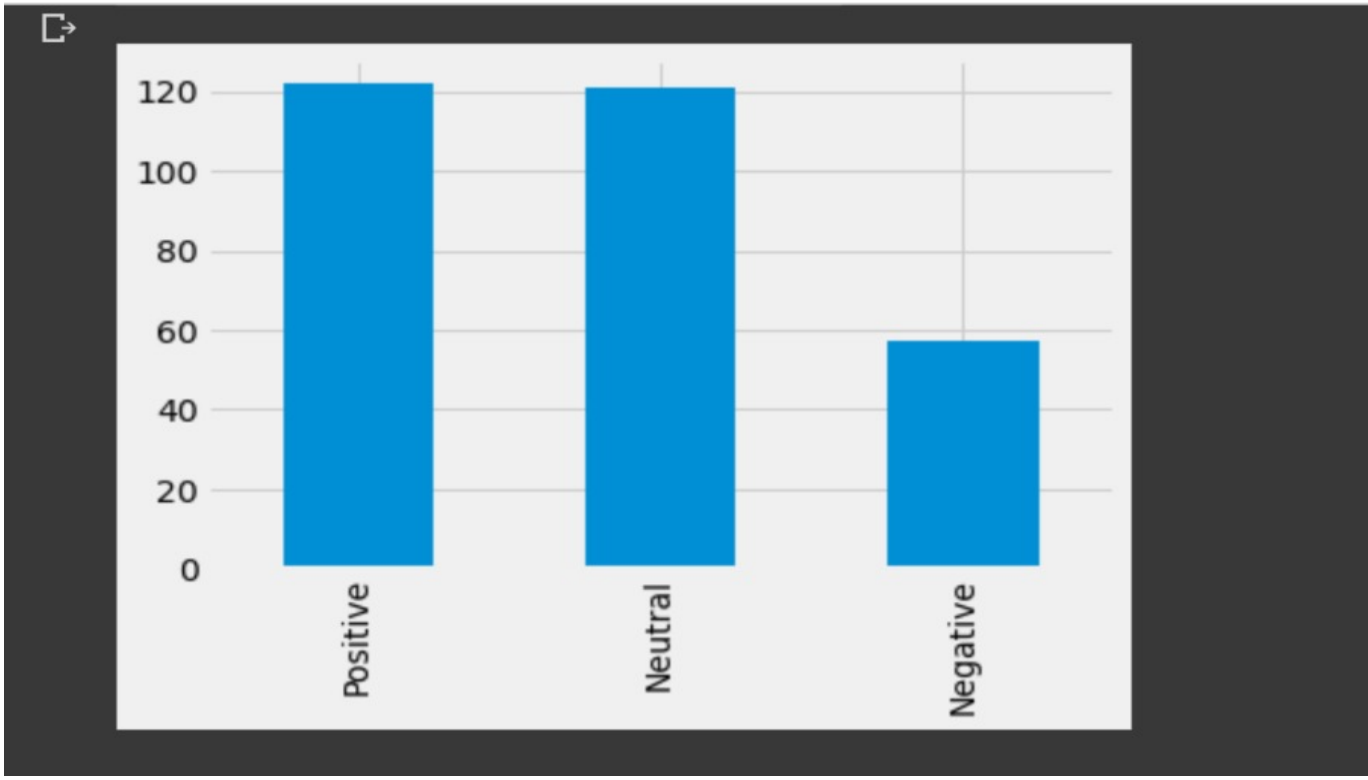
5.3.1 OUTPUT

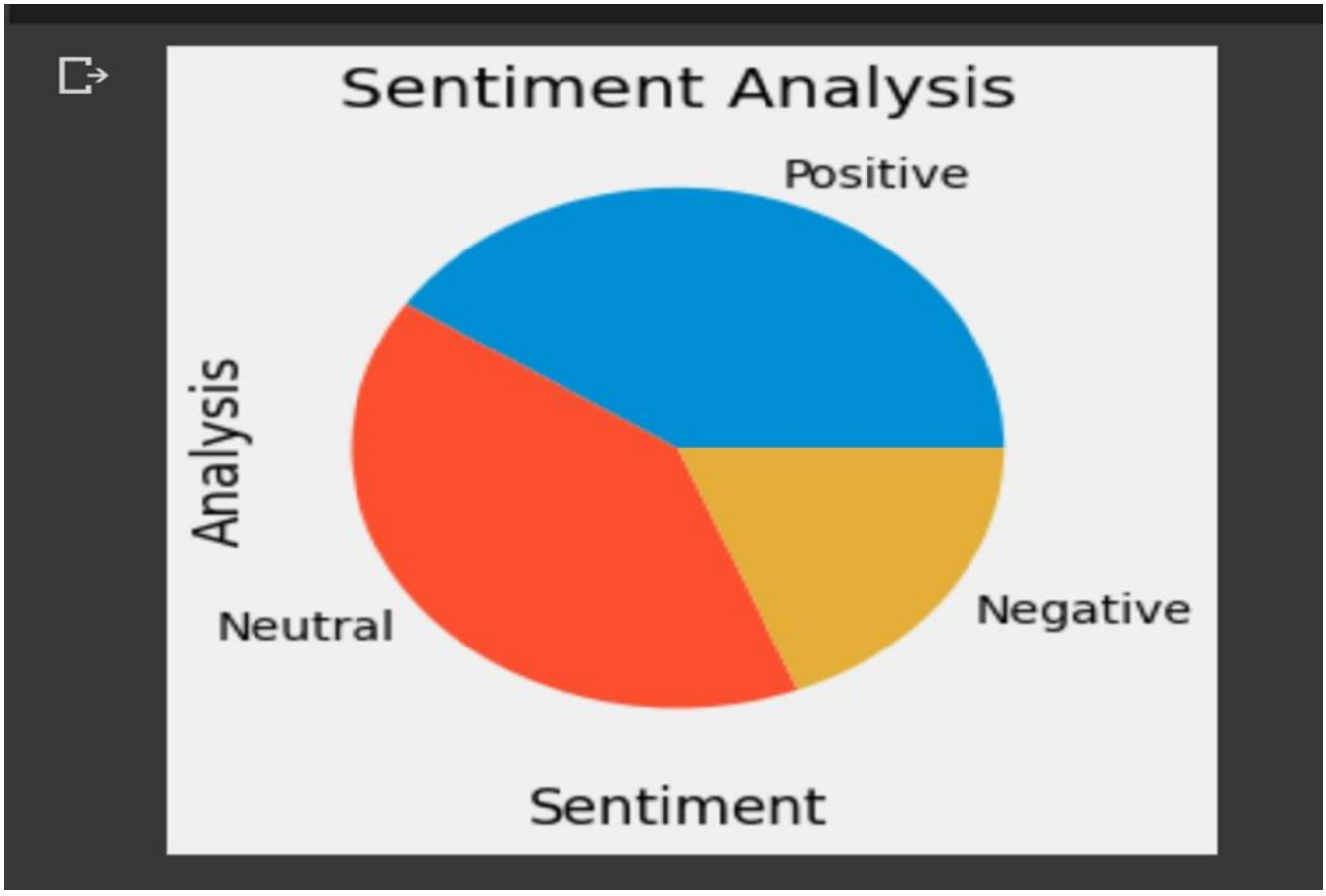
5.3.1.1 Word cloud based on frequency of words

5.3.1.2 Representation of sentimental analysis



5.3.1.3 Visualization in bar graph and Pie chart





5.4 EXPERIMENT RESULTS AND ANALYSIS

ACCURACY COMPARISON:

Dataset is passed to the support vector machine model and tokenization, vectorization are performed to break the sentence and understand the sentiment within the sentence. Precision, accuracy, recall, f1-score, confusion matrix are indicators of machine learning model's performance.

```
import pandas as pd
# train Data
trainData = pd.read_csv("/content/train.csv")
# test Data
testData = pd.read_csv("/content/test.csv")
from sklearn.feature_extraction.text import TfidfVectorizer
# Create feature vectors
vectorizer = TfidfVectorizer(min_df = 5,
                             max_df = 0.8,
                             sublinear_tf = True,
                             use_idf = True)
train_vectors = vectorizer.fit_transform(trainData['Content'])
test_vectors = vectorizer.transform(testData['Content'])
import time
from sklearn import svm
from sklearn import metrics
```

Fig 5.4.1 Passing dataset to model

```
Out[71]: Text(0.5, 0, 'test-size percent')
```

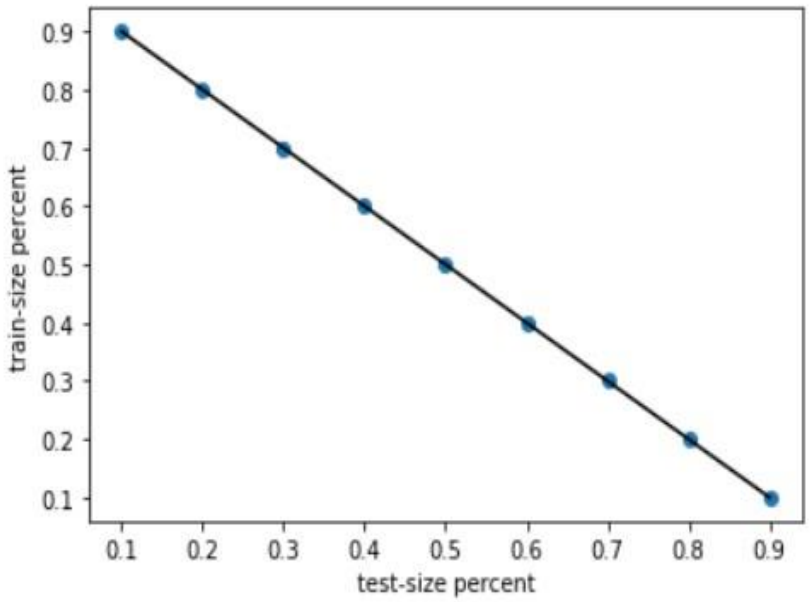


Fig 5.4.2 Train data and Test data(Their values affect final scores Fig 5.4.3)

```
Out[70]: Text(0.5, 0, 'test_split')
```

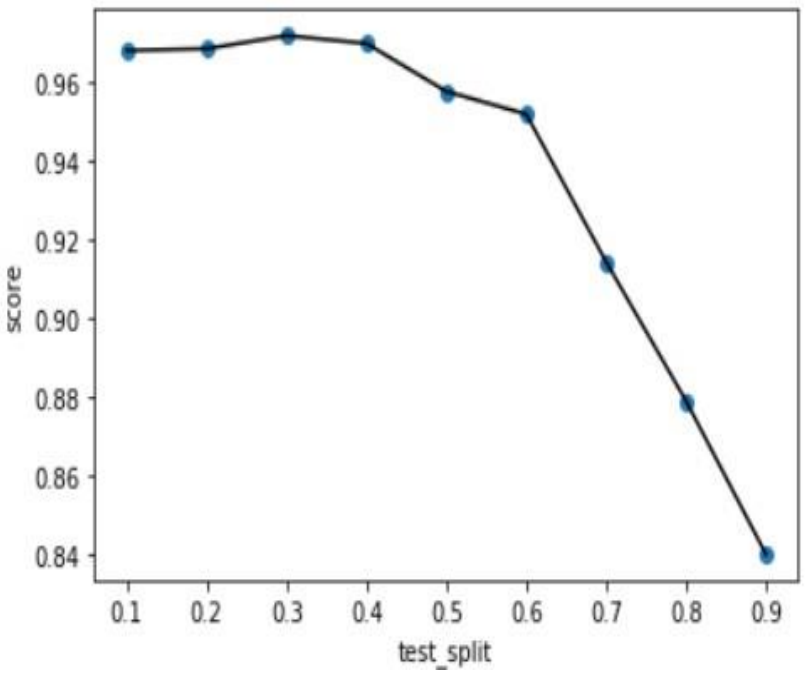


Fig 5.4.3 Scores based on test_split

Thus it can be inferred from these figures that test-split (which means selecting some percent as test data and the rest as train data) affects the final score of our classifier. So choosing an optimal value is a necessity in our case test-split=0.3 ended up with the maximum score.

5.4.4 Accuracy of SVM model

```
t2 = time.time()
time_linear_train = t1-t0
time_linear_predict = t2-t1
# results
print("Training time: %fs; Prediction time: %fs" % (time_linear_train, time_li
report = classification_report(testData['Label'], prediction_linear, output_di
print('positive: ', report['pos'])
print('negative: ', report['neg'])
#print(metrics.confusion_matrix(train_vectors, test_vectors))
print("Confusion matrix")
cf=metrics.confusion_matrix(testData['Label'],prediction_linear)
print(cf)
review="Sree vidyanikethan college in tirupati is good"
review_vector = vectorizer.transform([review]) # vectorizing
print(classifier_linear.predict(review_vector))
```

```
Training time: 9.288609s; Prediction time: 0.900316s
positive:  {'precision': 0.9191919191919192, 'recall': 0.91, 'f1-score': 0.914
negative:  {'precision': 0.9108910891089109, 'recall': 0.92, 'f1-score': 0.915
Confusion matrix
[[92  8]
 [ 9 91]]
['pos']
```

5.4.5 Accuracy of Naïve Bayes model

```
[ ]
model = MultinomialNB()
model.fit(x, y)
#print(model.score(x_test, y_test))
y_pred=model.predict(x_test)
a=metrics.accuracy_score(y_test,y_pred)
print("Accuracy:")
print(a*100,end="")
print("%")
print("Confusion matrix:")
cf=metrics.confusion_matrix(y_test,y_pred)
print(cf)
```

```
Accuracy:
81.55555555555556%
Confusion matrix:
[[187  38]
 [ 45 180]]
```