# Name: Javaid Ahmed

# Email: javaid.sameer@gmail.com

# Mobile: 0321-3210333


# Batch: PNY Machine Learning & Artifiial Intelligence (6th Batch)

## *Sentimental Analysis using Natural Language Processing w.r.t Text and Speech Recognition based on Support Vector Machine (SVM) & Naive Bayes Classifier (NBC)*

## To Ignore Warnings

In [1]:
```
# To ignore future and updation warnings of libraries

import warnings
warnings.filterwarnings("ignore")
```

## Import Useful Libraries

In [3]:
```
import pandas as pd                                          # For read and write on .csv files
import numpy as np                                           # For dealing with numeric arrays

from sklearn import metrics                                  # For dealing with matrices
from sklearn.model_selection import train_test_split         # For spliting the data
from sklearn.feature_extraction.text import TfidfVectorizer  # For forming feature vectors of text data

from sklearn import svm                                      # For performing SVM algorithm
from sklearn.naive_bayes import MultinomialNB                # For performing NBC algorithm

import matplotlib.pyplot as plt                              # For graphical presentation of data
import seaborn as sns;    sns.set(font_scale = 1.2)          # For ploting scatter plot of data points.

# Built-in Library for Sentiment Analysis
from textblob import TextBlob                                # For calculating the polarity of sentiments

# Libraries for Speech Recognition
import pyttsx3                                               # For accessing window's built-in voice
import speech_recognition as sr                              # For voice recognition

# Library to translate one language into other
from googletrans import Translator                           # For using Google translator
translator = Translator()

# For date and time
import datetime
```

# Import Datasets in csv Files which are based on Tweets

In [4]:
```
bf1 = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Datasets/Sorrow_tweets.csv")
bf2 = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Datasets/Sad_tweets.csv")
bf3 = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Datasets/Pleasure_tweets.csv")
bf4 = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Datasets/Happy_tweets.csv")
```

# Preparing the Datasets for visualization

## Clean any rows which contain a "NaN" in them

In [5]:
```
def ClearNaN(df):
    return df.dropna(axis=0, how = 'any')

bf1 = ClearNaN(bf1)
bf2 = ClearNaN(bf2)
bf3 = ClearNaN(bf3)
bf4 = ClearNaN(bf4)
```

## Any non-understandable text converts into nothing

In [6]:
```
# Fuction due to which this process takes place
def removetext(text):
    return ''.join([j if ord(j) < 128 else '' for j in text])

bf1['Text'] = bf1['Text'].apply(removetext)
bf2['Text'] = bf2['Text'].apply(removetext)
bf3['Text'] = bf3['Text'].apply(removetext)
bf4['Text'] = bf4['Text'].apply(removetext)
```

## Convert all text either in (upper or lower) into lower case

In [7]:
```
bf1['Text'] = bf1['Text'].apply(lambda y: y.lower())
bf2['Text'] = bf2['Text'].apply(lambda y: y.lower())
bf3['Text'] = bf3['Text'].apply(lambda y: y.lower())
bf4['Text'] = bf4['Text'].apply(lambda y: y.lower())
```

## Remove all punctuation and extra lines

In [8]:
```
def Punc_ExtraLines_Removal(pf):
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('.',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('\n',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('?',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('!',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('"',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace(';',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('#',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace('&amp',' '))
    pf['Text'] = pf['Text'].apply(lambda z: z.replace(',',' '))
    return pf

pf1 = Punc_ExtraLines_Removal(bf1)
pf2 = Punc_ExtraLines_Removal(bf2)
pf3 = Punc_ExtraLines_Removal(bf3)
pf4 = Punc_ExtraLines_Removal(bf4)
```

## Get each unique keyword from Dataframe

In [9]:
```
array1 = pf1['Text'].str.split(' ', expand=True).stack().value_counts()
array2 = pf2['Text'].str.split(' ', expand=True).stack().value_counts()
array3 = pf3['Text'].str.split(' ', expand=True).stack().value_counts()
array4 = pf4['Text'].str.split(' ', expand=True).stack().value_counts()
```

## Make a dataframe of words and the frequency with which the words appear

In [10]:
```
d1 = {'Word': array1.index, 'Frequency':array1}
wf1 = pd.DataFrame(data = d1)

d2 = {'Word': array2.index, 'Frequency':array2}
wf2 = pd.DataFrame(data = d2)

d3 = {'Word': array3.index, 'Frequency':array3}
wf3 = pd.DataFrame(data = d3)

d4 = {'Word': array4.index, 'Frequency':array4}
wf4 = pd.DataFrame(data = d4)
```

## Remove all words which are less than 10 times, NaN & signs in these words e.g. :(

In [11]:
```
def DataFrame_Cleaning_1(rf):
    rf['Frequency'] = rf['Frequency'][rf['Frequency'] > 10]
    rf = rf.dropna(axis=0, how = 'any')
    rf = rf.drop([':(','https://t',':((', ':(((', ':(((((', ':((((((', ':', '(', ''])
    return rf

def DataFrame_Cleaning_2(rf):
    rf['Frequency'] = rf['Frequency'][rf['Frequency'] > 10]
    rf = rf.dropna(axis=0, how = 'any')
    rf = rf.drop([':(','https://t',':((', ':(((', ''])
    return rf

def DataFrame_Cleaning_3(rf):
    rf['Frequency'] = rf['Frequency'][rf['Frequency'] > 10]
    rf = rf.dropna(axis=0, how = 'any')
    rf = rf.drop([':(','https://t', ''])
    return rf

def DataFrame_Cleaning_4(rf):
    rf['Frequency'] = rf['Frequency'][rf['Frequency'] > 10]
    rf = rf.dropna(axis=0, how = 'any')
    rf = rf.drop([':(','https://t', ''])
    return rf

cf1 = DataFrame_Cleaning_1(wf1)
cf2 = DataFrame_Cleaning_2(wf2)
cf3 = DataFrame_Cleaning_3(wf3)
cf4 = DataFrame_Cleaning_4(wf4)
```

## Sorrow Dataset

In [12]:
```
cf1.head(5)
```

Out[12]:

| | Word | Frequency |
|---|---|---|
| **i** | i | 8448.0 |
| **to** | to | 3588.0 |
| **the** | the | 2940.0 |
| **my** | my | 2663.0 |
| **and** | and | 2552.0 |

## Sad Dataset

In [13]:
```
cf2.head(5)
```

Out[13]:

| | Word | Frequency |
|---|---|---|
| **sad** | sad | 12748.0 |
| **i** | i | 6427.0 |
| **the** | the | 5511.0 |
| **to** | to | 5262.0 |
| **and** | and | 4363.0 |

## Pleasure Dataset

In [14]:
```
cf3.head(5)
```

Out[14]:

| | Word | Frequency |
|---|---|---|
| **fun** | fun | 12274.0 |
| **the** | the | 6306.0 |
| **to** | to | 5388.0 |
| **a** | a | 4883.0 |
| **and** | and | 4723.0 |

### Happy Dataset

In [15]:
```
cf4.head(5)
```

Out[15]:

| | Word | Frequency |
|---|---|---|
| **happy** | happy | 13776.0 |
| **i** | i | 5019.0 |
| **to** | to | 4991.0 |
| **you** | you | 4444.0 |
| **birthday** | birthday | 4029.0 |

# To split Training data from csv files

In [16]:
```
path1 = "E:/WORK/FINAL YEAR PROJECT/Project Codes/Datasets/Sorrow_tweets.csv"
path2 = "E:/WORK/FINAL YEAR PROJECT/Project Codes/Datasets/Sad_tweets.csv"
path3 = "E:/WORK/FINAL YEAR PROJECT/Project Codes/Datasets/Pleasure_tweets.csv"
path4 = "E:/WORK/FINAL YEAR PROJECT/Project Codes/Datasets/Happy_tweets.csv"

def split_dataset(path):
    sf = pd.read_csv(path)

    # Clean any rows with which contain a "NaN" in them
    sf = ClearNaN(sf)

    # Any non-understandable text converts into nothing
    sf['Text'] = sf['Text'].apply(removetext)

    # Make all my texts lower case
    sf['Text'] = sf['Text'].apply(lambda i: i.lower())

    # Remove all puctuation marks
    sf = Punc_ExtraLines_Removal(sf)

    # Separate all the words in the text
    sf['Text']= sf['Text'].str.split()

    return sf

sf1 = split_dataset(path1)
sf2 = split_dataset(path2)
sf3 = split_dataset(path3)
sf4 = split_dataset(path4)

sf1.to_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Sorrow_split.csv")
sf2.to_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Sad_split.csv")
sf3.to_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Pleasure_split.csv")
sf4.to_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Happy_split.csv")
```

## Sorrow Split Dataset

In [17]:
```
sf1.head(5)
```

Out[17]:

| | ID | Text |
|---|---|---|
| **0** | 1 | [why, does, my, stupid, stomach, keep, playing... |
| **1** | 1 | [if, anyone, boos, my, boo, ever, again, while... |
| **2** | 1 | [my, wifes, boyfriend, took, my, nintendo, swi... |
| **3** | 1 | [i've, been, trying, to, set, up, a, tip, anim... |
| **4** | 1 | [isnt, the, first, kiss, with, someone, magica... |

## Sad Split Dataset

In [18]:
```
sf2.head(5)
```

Out[18]:

| | ID | Text |
|---|---|---|
| **0** | 1 | [this, made, me, sad, https://t, co/o1y4mfz8au] |
| **1** | 1 | [@donv86, @hustlethecode, @bigkannon, @chillaj... |
| **2** | 1 | [lacethefuckupkells:, i, swear, this, love, is... |
| **3** | 1 | [i, cant, believe, drake, going, out, sad, lik... |
| **4** | 1 | [@kerrrieb, @mariamentado, a, part, of, me, is... |

## Pleasure Split Dataset

In [19]:
```
sf3.head(5)
```

Out[19]:

| | ID | Text |
|---|---|---|
| **0** | 1 | [kids, movie, night, -, popcorn, soda, fun, th... |
| **1** | 1 | [sounds, like, a, fun, weekend, @weareroli, ht... |
| **2** | 1 | [@nepunepugear, may, i, use, my, meaty, shryng... |
| **3** | 1 | [i'm, having, fun, laughing, at, nbc, but, i, ... |
| **4** | 1 | [@smilecorbyn, @whydontwemusic, have, fun] |

### Happy Split Dataset

In [20]:
```
sf4.head(5)
```

Out[20]:

| | ID | Text |
|---|---|---|
| **0** | 1 | [happy, birthday, big, bro] |
| **1** | 1 | [happy, birthday, shimika, brown] |
| **2** | 1 | [happy, birthday, my, son, mamabendecida, http... |
| **3** | 1 | [oooooo, ya, girl, is, so, happy, rn, https://... |
| **4** | 1 | [@williegeist, sweet, castle, br0, happy, bday] |

## Classify the Tweets into positive & negative

In [21]:
```
sorrow = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Sorrow_split.csv")
sad = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Sad_split.csv")
pleasure = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Pleasure_split.csv")
happy = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Split_Datasets/Happy_split.csv")

# The dataframe pleasure and happy as 1 (positive)
# The dataframe sorrow and sad as  0 (negative)

sorrow['Type'] = 0
sad['Type'] = 0
pleasure['Type'] = 1
happy['Type'] = 1
```

### Length of the dataframes

In [22]:
```
# Positive dataframes
p = len(pleasure) + len(happy)
print("Length of Positive dataframes: ", p)

# Negative dataframes
n = len(sorrow) + len(sad)
print("Length of Negative dataframes: ", n)
```

```
Length of Positive dataframes:  30000
Length of Negative dataframes:  30000
```

# Join all of the dataframes into a big one for easier processing

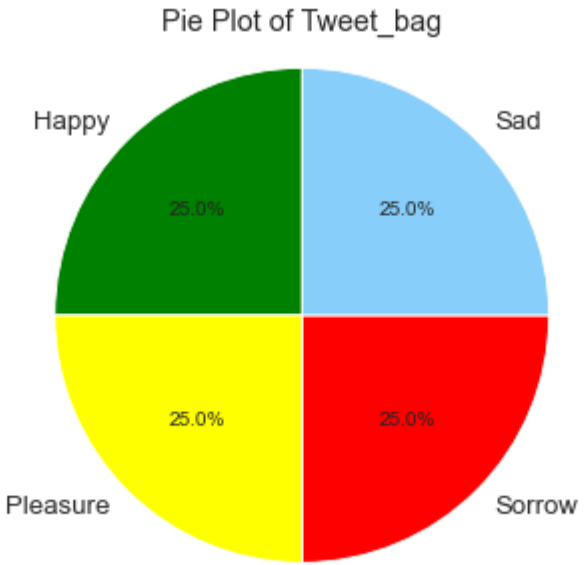
In [23]:
```
tbf = pd.concat([sorrow,sad,pleasure,happy]).reset_index(drop=True)

tbf.to_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Tweet_bag.csv")
tbf.head(5)
```

Out[23]:

| | Unnamed: 0 | ID | Text | Type |
|---|---|---|---|---|
| **0** | 0 | 1 | ['why', 'does', 'my', 'stupid', 'stomach', 'ke... | 0 |
| **1** | 1 | 1 | ['if', 'anyone', 'boos', 'my', 'boo', 'ever', ... | 0 |
| **2** | 2 | 1 | ['my', 'wifes', 'boyfriend', 'took', 'my', 'ni... | 0 |
| **3** | 3 | 1 | ["i've", 'been', 'trying', 'to', 'set', 'up', ... | 0 |
| **4** | 4 | 1 | ['isnt', 'the', 'first', 'kiss', 'with', 'some... | 0 |

# Pie Plot of Tweet bag

In [24]:
```
data = np.array([len(sorrow), len(sad), len(happy), len(pleasure)])
labels = ['Sorrow', 'Sad', 'Happy', 'Pleasure']
colrs=['red', 'lightskyblue', 'green', 'yellow']

fig, ax = plt.subplots(figsize=(10, 5))
explode = (0, 0, 0, 0)
ax.pie(data, explode=explode, labels=labels, autopct='%1.1f%%', startangle=270, colors=colrs)
ax.axis('equal')             # keep it a circle

plt.title("Pie Plot of Tweet_bag")
fig.savefig('C:/Users/HP/Desktop/PNY Training Project/Pie Plot of Tweet_bag.png', dpi=300)
plt.show()
```




# Create Training dataset and Test dataset

In [25]:
```
# Evaluate the model by split it into train and test sets
training, testing = train_test_split(tbf, test_size=0.2, random_state=1)

X_train = training['Text'].values
X_test = testing['Text'].values

y_train = training['Type']
y_test = testing['Type']
```

# Create Feature vectors for SVM

In [26]:
```
v1 = TfidfVectorizer(min_df = 5, max_df = 0.8, sublinear_tf = True, use_idf = True)

tr_vectors1 = v1.fit_transform(X_train)
ts_vectors1 = v1.transform(X_test)
```

# Create Feature vectors for NBC

In [27]:
```
v2 = TfidfVectorizer(max_df=0.8, sublinear_tf=True, smooth_idf = True, ngram_range=(1, 2), stop_words='englis
h')

tr_vectors2 = v2.fit_transform(X_train)
ts_vectors2 = v2.transform(X_test)
```

# Support Vector Machine (SVC)

In [28]:
```
classifier_linear = svm.SVC(kernel='linear')
classifier_linear.fit(tr_vectors1, y_train)

y_train_pred1 = classifier_linear.predict(tr_vectors1)
y_test_pred1  = classifier_linear.predict(ts_vectors1)

print("Accuracy of trained model :", metrics.accuracy_score(y_train, y_train_pred1))
print("Accuracy of test model :", metrics.accuracy_score(y_test, y_test_pred1))
```

```
Accuracy of trained model : 0.9579583333333334
Accuracy of test model : 0.9395
```

# Naive Bayes Classifier (NBC)

In [29]:
```
classifier = MultinomialNB()
classifier = classifier.fit(tr_vectors2, y_train)

y_train_pred2 = classifier.predict(tr_vectors2)
y_test_pred2 = classifier.predict(ts_vectors2)

print('\nAccuracy of trained model : ', metrics.accuracy_score(y_train, y_train_pred2))
print('Accuracy of trained model : ', metrics.accuracy_score(y_test, y_test_pred2))
```

```
Accuracy of trained model :  0.9899166666666667
Accuracy of trained model :  0.8950833333333333
```

### Cleaning of text and assigning polarity nature to each seantence in the Tweet Bag

In [30]:
```
def clean_dataset(path):
    ef = pd.read_csv(path)

    # Clean any rows with which contain a "NaN" in them
    ef = ClearNaN(ef)

    # Any non-understandable text converts into nothing
    ef['Text'] = ef['Text'].apply(removetext)

    # Make all my texts lower case
    ef['Text'] = ef['Text'].apply(lambda c: c.lower())

    # Remove all puctuation markss
    ef = Punc_ExtraLines_Removal(ef)

    return ef

sf1 = clean_dataset(path1)
sf2 = clean_dataset(path2)
sf3 = clean_dataset(path3)
sf4 = clean_dataset(path4)

wordbag2 = pd.concat([sf1,sf2,sf3,sf4])
wordbag2 = wordbag2.drop_duplicates()
wordbag2 = wordbag2.dropna(axis=0, how = 'any')


wordbag2.to_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Tweet_bag_clean.csv")

wordbag3 = pd.read_csv("C:/Users/HP/Desktop/PNY Training Project/Datasets/Tweet_bag_clean.csv")

wordbag3['positive'] = 0
wordbag3['negative'] = 0

k = wordbag3['Text']
g = wordbag3['positive']
h = wordbag3['negative']

for i in range(len(k)):
    c = TextBlob(str(k[i])).sentiment.polarity
    if c > 0:
        g[i] = round(c*100)
```

```
        h[i] = round((1-c)*100)
    elif c == 0:
        g[i] = 50
        h[i] = 50
    else:
        g[i] = (round(c*100))+100
        h[i] = 100 - g[i]

wordbag3.to_csv('C:/Users/HP/Desktop/PNY Training Project/Datasets/Tweet_bag_polarity.csv')
wordbag3.head(5)
```

Out[30]:

| | Unnamed: 0 | ID | Text | positive | negative |
|---|---|---|---|---|---|
| **0** | 0 | 1 | why does my stupid stomach keep playing up on ... | 23 | 77 |
| **1** | 1 | 1 | if anyone boos my boo ever again while hes bat... | 25 | 75 |
| **2** | 2 | 1 | my wifes boyfriend took my nintendo switch awa... | 25 | 75 |
| **3** | 3 | 1 | i've been trying to set up a tip animation for... | 25 | 75 |
| **4** | 4 | 1 | isnt the first kiss with someone magical i do... | 50 | 50 |

# Testing using Text

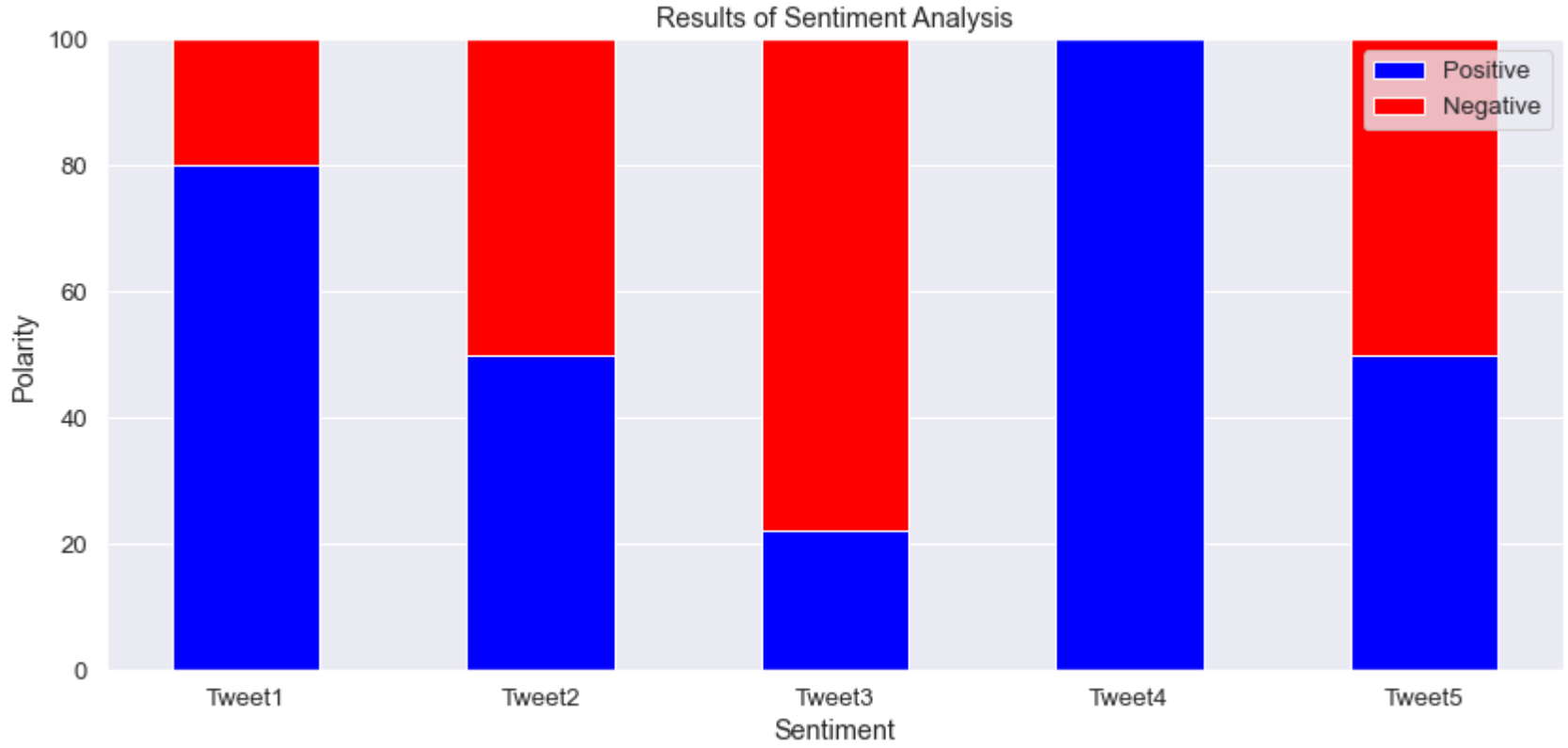
In [31]:
```
# Function for Testing the program code

def Testing(tweet):
    q1 = v1.transform([tweet])        # Using SVM
    r1 = classifier_linear.predict(q1)

    q2 = v2.transform([tweet])        # Using NBC
    r2 = classifier.predict(q2)

    # Built-in function to check the polarity of the sentiment
    polarity = TextBlob(str(tweet)).sentiment.polarity

    if polarity > 0:
        m = round(polarity*100)
        n = round((1-polarity)*100)
    elif polarity == 0:
        m = 50
        n = 50
    else:
        m = (round(polarity*100))+100
        n = 100 - m

    if r1 == 1:
        print('        Sentiment by using SVM: Positive')
    else:
        print('        Sentiment by using SVM: Negative')

    if r2 == 1:
        print('        Sentiment by using NBC: Positive')
    else:
        print('        Sentiment by using NBC: Negative')

    print('         Positive Polarity: ', m, "%")
    print('         Negative Polarity: ', n, "%")
    print('')
```

In [32]:
```
tweet1 = "Happy Birthday to you"
print('Tweet 1: ')
Testing(tweet1)

tweet2 = "I dont know that I can do it anymore."
print('Tweet2: ')
Testing(tweet2)

tweet3 = "This place is really very dirty to stay."
print('Tweet3: ')
Testing(tweet3)

tweet4 = "You are looking very beautiful and awesome today"
print('Tweet4: ')
Testing(tweet4)


tweet5 = "Happy! Final Year Project is done."
print('Tweet5: ')
Testing(tweet5)
```

```
Tweet 1:
        Sentiment by using SVM: Positive
        Sentiment by using NBC: Positive
         Positive Polarity:  80 %
         Negative Polarity:  20 %

Tweet2:
        Sentiment by using SVM: Negative
        Sentiment by using NBC: Negative
         Positive Polarity:  50 %
         Negative Polarity:  50 %

Tweet3:
        Sentiment by using SVM: Negative
        Sentiment by using NBC: Negative
         Positive Polarity:  22 %
         Negative Polarity:  78 %

Tweet4:
        Sentiment by using SVM: Positive
        Sentiment by using NBC: Positive
         Positive Polarity:  100 %
         Negative Polarity:  0 %

Tweet5:
        Sentiment by using SVM: Positive
        Sentiment by using NBC: Positive
         Positive Polarity:  50 %
         Negative Polarity:  50 %
```

# Stacked Bar Chart of the result

In [33]:
```
pos = np.array([80, 50, 22, 100, 50])
neg = np.array([20, 50, 78, 0, 50])

labels = ['Tweet1', 'Tweet2', 'Tweet3', 'Tweet4', 'Tweet5']

wdth = 0.5

figure, axis = plt.subplots(figsize=(12, 6))
axis.bar(labels, pos, wdth, color='blue', label='Positive')
axis.bar(labels, neg, wdth, color='red', label='Negative', bottom=pos)

axis.yaxis.grid(True)
axis.legend(loc='best')

axis.set_xlabel('Sentiment')
axis.set_ylabel('Polarity')

plt.title('Results of Sentiment Analysis')
figure.tight_layout(pad=2)
figure.savefig('C:/Users/HP/Desktop/PNY Training Project/Testing Results.png', dpi=300)

plt.show()
```




## Speech Recognition

In [34]:
```
engine = pyttsx3.init('sapi5')
engine.setProperty('voice', "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\TTS_MS_EN-US_ZIRA_11.
0")

query = ''

def Speak(audio):
    engine.setProperty('rate', 150)
    engine.say(audio)
    engine.runAndWait()

def TakeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 0.5
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language= 'en-pk')
        print(f"User said: {query} \n")
    except Exception:
        print("Please! Say that again...\n")
        Speak("Please! Say that again...")
        return "None"
    return query
```

## Greeting

In [35]:
```
def Greeting():
    hour = int(datetime.datetime.now().hour)
    if hour>=0 and hour<12:
        print("Good Morning!")
        Speak("Good Morning!")
    elif hour>=12 and hour<18:
        print("Good Afternoon!")
        Speak("Good Afternoon!")
    else:
        print("Good Evening!")
        Speak("Good Evening!")
    print("Please give me your sentiment Sir! \n")
    Speak("Please give me your sentiment Sir!")
```

## Command Function for Sentiment analysis using voice recognition in Urdu

In [36]:
```
def TakeCommand():
    Greeting()
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 0.5
        audio = r.listen(source)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio)
        print(f"\nUser said: {query} \n")
    except Exception:
        print("Please! Say that again...\n")
        Speak("Please! Say that again...")
        return TakeCommand()
    return query

def SentimentAnalysisUsingVoiceRecognition():
    a = TakeCommand()

    tr_sen = translator.translate(a, src='ur', dest='en')
    s = tr_sen
    print("In English: "+s.text+"\n")
    print("Results:")
    Testing(str(s))

    # Built-in function to check the polarity of the sentiment
    polarity = TextBlob(str(s)).sentiment.polarity

    if polarity > 0:
        o = round(polarity*100)
        p = round((1-polarity)*100)
    elif polarity == 0:
        o = 50
        p = 50
    else:
        o = (round(polarity*100))+100
        p = 100 - o

    x = ["pos", "neg"]
    y = (o, p)
```

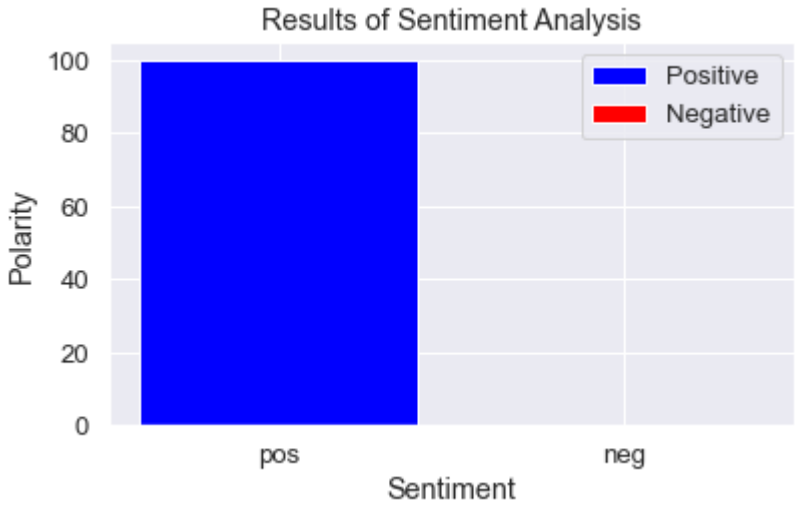
```
    plt.bar(x[0], y[0], color='blue', label='Positive')
    plt.bar(x[1], y[1], color='red', label='Negative')

    plt.legend('Positive')
    plt.legend

    plt.grid(True)
    plt.legend(loc='best')

    plt.xlabel('Sentiment')
    plt.ylabel('Polarity')

    plt.title('Results of Sentiment Analysis')
    plt.tight_layout(pad=2)

    plt.show()
```

# Testing using Voice Recognition

## Happy Sentiment

In [39]:
```
SentimentAnalysisUsingVoiceRecognition()
```

```
Good Morning!
Please give me your sentiment Sir!

Listening...
Recognizing...

User said: main bahut khush hun

In English: I'm very happy

Results:
        Sentiment by using SVM: Positive
        Sentiment by using NBC: Positive
         Positive Polarity:  100 %
         Negative Polarity:  0 %
```




## Sad Sentiment

In [38]:
```
SentimentAnalysisUsingVoiceRecognition()
```

```
Good Morning!
Please give me your sentiment Sir!

Listening...
Recognizing...

User said: main bahut Dukhi hun

In English: I am very sad

Results:
        Sentiment by using SVM: Negative
        Sentiment by using NBC: Negative
         Positive Polarity:  35 %
         Negative Polarity:  65 %
```