

# SENTIMENT ANALYSIS FOR MARKETING

When introducing a new product, sentiment analysis can be utilized to gauge initial public reactions and perceptions.

Analyzing social media, customer feedback, and reviews during the introduction phase helps to identify the sentiment

(positive, negative, neutral) surrounding the product launch. This allows marketing teams to swiftly adjust strategies,

address concerns, emphasize positive aspects, and engage with the audience effectively based on the sentiment

trends observed

## SAMPLE CODE:

```
SENTIMENT ANALYSIS FOR MARKETING
import pandas as pd
import TextBlob
import matplotlib.pyplot as plt
def analyze_sentiment(text):
    analysis = TextBlob(text)
    # Classify the polarity of the text
    if analysis.sentiment.polarity > 0: return 'Positive'
    elif analysis.sentiment.polarity == 0: return 'Neutral' else:
    return 'Negative'
# Sample marketing data (New product launches)
data= {
    'feedback': [
        "This product is amazing!",
        "Not impressed with the customer service.",
        "Neutral comment on the product.",
        # Add more feedback here...
    ]
}
# Create a DataFrame from the data
df = pd.DataFrame(data)
# Analyze sentiment for each feedback
```

```
df['sentiment'] = df[feedback].apply(analyze_sentiment)
# Visualize sentiment distribution
sentiment_counts = df['sentiment'].value_counts
plt.bar (sentiment_counts.index, sentiment _counts.values) plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Analysis for Marketing Feedback') plt.show()
# Display the analyzed data
print(df)
```

## STEP FOR SENTIMENT ANALYSIS

Step 1: normality testing of dataset. In this research data analysis, a normality testing of the dataset for each tenant was conducted to determine the dataset distribution

Step 2: data pre-processing

Step 3: model development (training)

Step 4: model evaluation (test)

Dataset link: <https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment>

# SENTIMENT ANAL PROGRAM

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

nltk.download('vader_lexicon')
nltk.download('stopwords')
nltk.download('punkt')

data = pd.read_csv('D:\set\Tweets.csv')
stop_words = set(stopwords.words('english'))

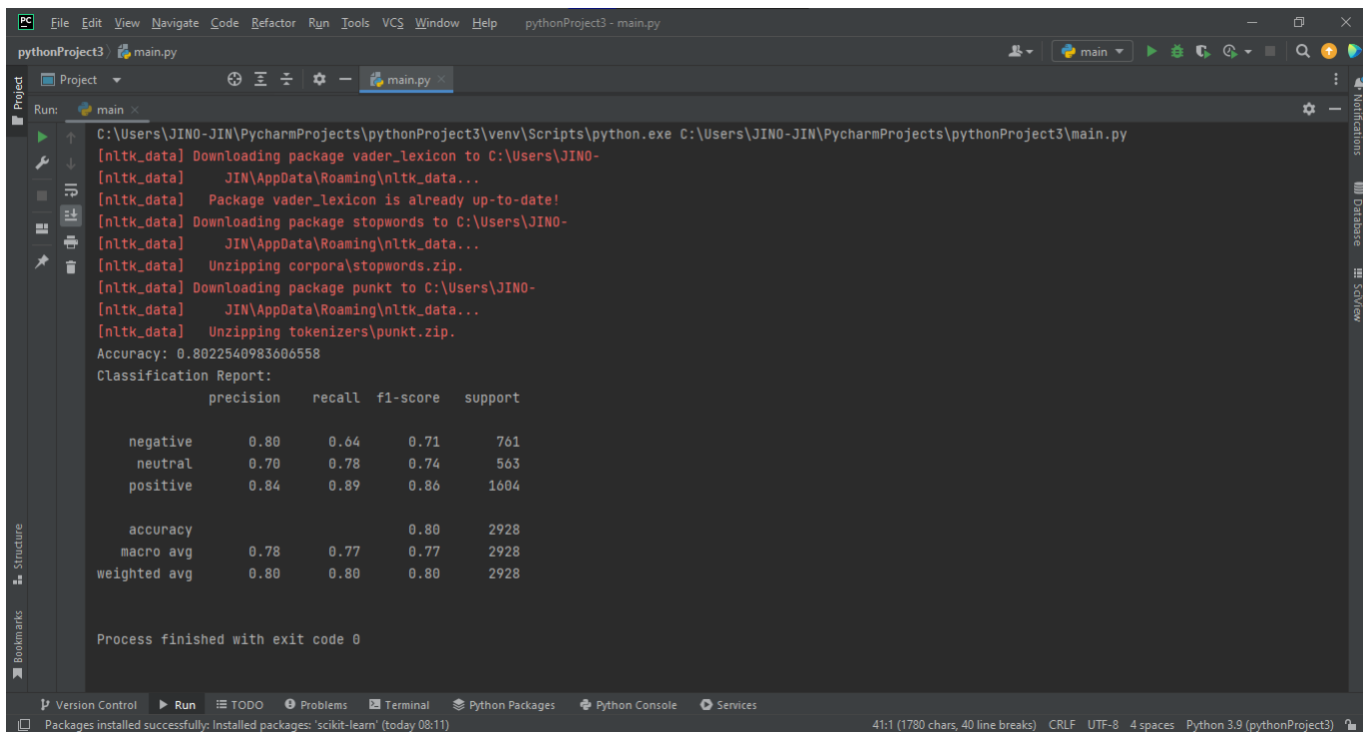
def preprocess_text(text):
    words = word_tokenize(text)
    filtered_words = [word.lower() for word in words if word.isalnum() and
word.lower() not in stop_words]
    return ' '.join(filtered_words)

data['clean_text'] = data['text'].apply(preprocess_text)
sid = SentimentIntensityAnalyzer()

def analyze_sentiment(text):
    sentiment_scores = sid.polarity_scores(text)
    if sentiment_scores['compound'] >= 0.05:
        return 'positive'
    elif sentiment_scores['compound'] <= -0.05:
        return 'negative'
    else:
        return 'neutral'

data['sentiment'] = data['clean_text'].apply(analyze_sentiment)
X_train, X_test, y_train, y_test = train_test_split(data['clean_text'],
data['sentiment'], test_size=0.2, random_state=42)
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_tfidf, y_train)
y_pred = clf.predict(X_test_tfidf)
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n(classification_rep)')
```

# OUTPUT



```
C:\Users\JINO-JIN\PycharmProjects\pythonProject3\venv\Scripts\python.exe C:\Users\JINO-JIN\PycharmProjects\pythonProject3\main.py
[nltk_data] Downloading package vader_lexicon to C:\Users\JINO-
[nltk_data]   JIN\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\JINO-
[nltk_data]   JIN\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package punkt to C:\Users\JINO-
[nltk_data]   JIN\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
Accuracy: 0.8022540983606558
Classification Report:
      precision    recall  f1-score   support

 negative       0.80      0.64      0.71       761
  neutral       0.70      0.78      0.74       563
  positive       0.84      0.89      0.86      1604

 accuracy              0.80      2928
 macro avg       0.78      0.77      0.77      2928
 weighted avg     0.80      0.80      0.80      2928

Process finished with exit code 0
```

Version Control | Run | TODO | Problems | Terminal | Python Packages | Python Console | Services

Packages installed successfully: installed packages: 'scikit-learn' (today 08:11) 41:1 (1780 chars, 40 line breaks) CRLF UTF-8 4 spaces Python 3.9 (pythonProject3)

# CONCLUSION:

Remember to stay updated with the latest advancements in natural language processing and machine learning to enhance the accuracy and effectiveness of the prediction tool for diabetics

# FLOWCHART:

## BLOCK DIAGRAM FOR NEW LAUNCHES

