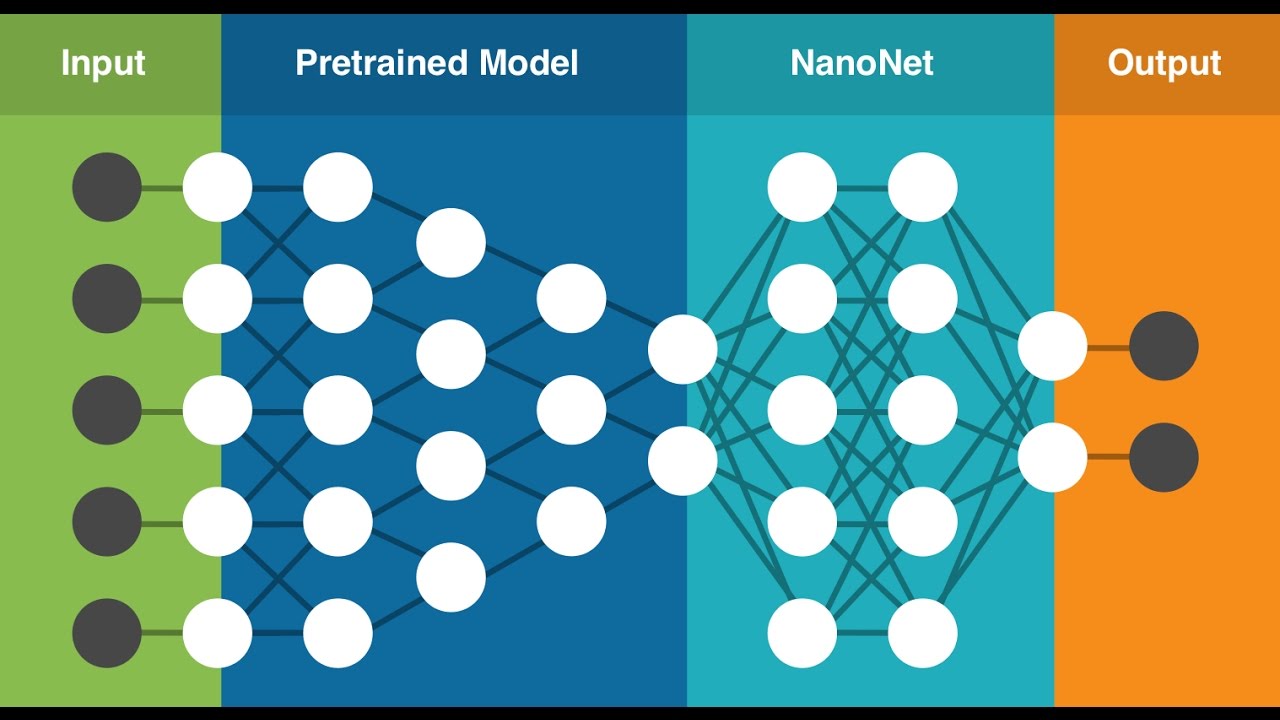
**SENTIMENT ANALYSIS FOR MARKETING**

**PROJECT PHASE 2**

One innovative technique we can explore is using pretrained language models like BERT for feature extraction. These models have demonstrated superior performance in NLP tasks.



# **INTRODUCTIONS**

Sentiment analysis, also known as opinion mining, is a valuable tool in marketing that involves analyzing and understanding the sentiments, emotions, and opinions expressed by customers and the general public about a product, brand, or service. It uses natural language processing and machine learning techniques to classify text data as positive, negative, or neutral.

**OBJECTIVES**

Sentiment analysis in marketing aims to understand customer opinions, improve brand perception, enhance products, measure campaign effectiveness, select influencers, manage crises, handle feedback, gauge satisfaction, tailor content, identify trends, position products, make data-driven decisions, and optimize the customer experience.

**METHODOLOGY**

**Data Preprocessing:** We start by importing the necessary libraries for data cleaning and tokenization, such as NLTK and spaCy. We then load the airline tweets dataset and perform some initial data exploration. We can also remove any irrelevant or noisy data such as tweets that contain links or mentions of other Twitter users. After that, we perform text cleaning by removing stop words, special characters, and other noise from the text data. Finally, we tokenize the text data and convert it into numerical format for feature extraction.

**Feature Extraction:** We can extract features from the tokenized text data using techniques such as Word2Vec or GloVe. These methods convert the text data into fixed-length vectors that can be fed into the RNN model. We can also use techniques such as tf-idf or bag-of-words to represent the text data.

**Model Building:** We can build a Recurrent Neural Network (RNN) model to classify the sentiment of the airline tweets. We can use architectures such as Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRU) to model the sequence of the text data. We can also use techniques such as dropout and batch normalization to regularize the model and prevent overfitting.

**Model Training:** We can train the RNN model on the preprocessed and feature-extracted data. We can use techniques such as cross-validation to evaluate the model’s performance. We can also use techniques such as early stopping to prevent the model from overfitting the training data.

**Model Testing:** We can test the trained model on new, unseen data to evaluate its performance. We can use metrics such as accuracy, precision, recall, and F1 score to evaluate the model’s performance on the test data. We can also use techniques such as confusion matrix and ROC curve to analyze the model’s performance in more detail.

**SOURCE CODE:**

# **Import the Neccesary Libraries**

importre

importwarnings

importitertools

importnumpyasnp

importpandasasPd

importseabornassns

importtensorflowastf

importmatplotlib.pyplotasplt

fromkeras.preprocessing.textimportTokenizer

fromkeras.preprocessing.sequenceimportpad\_sequences

fromsklearn.model\_selectionimporttrain\_test\_split

fromsklearn.metricsimportconfusion\_matrix

warnings.filterwarnings("ignore",category=**FutureWarning**)

# **Reading the Dataset**

data=pd.read\_csv("../input/Tweets.csv")

df=data[["text","airline\_sentiment"]]

df['text']=df['text'].map(lambdax:x.lstrip('@VirginAmerica@UnitedAir@Southwestairline@DeltaAir@USAirways@American').rstrip('@'))

df

sns.countplot(data.airline);

df=df[df.airline\_sentiment!="neutral"]*# To remove neutral responses*

df['text']=df['text'].apply(lambdax:x.lower())*# To lower*

df['text']=df['text'].apply((lambdax:re.sub('[^a-zA-z0-9\s]','',x)))*# To keep numbers and strings only*

df.head(5)*#Quick Look*

sns.countplot(df.airline\_sentiment);*#Mostly Negative Reviews(Class Imbalance found)*

df=df.drop(df[df.airline\_sentiment=="negative"].iloc[:5000].index)

sns.countplot(df.airline\_sentiment);

max\_fatures=4000

tokenizer=Tokenizer(num\_words=max\_fatures,split=' ')

tokenizer.fit\_on\_texts(df['text'].values)

X=tokenizer.texts\_to\_sequences(df['text'].values)

X=pad\_sequences(X)

Y=df['airline\_sentiment']

L=Y.values

X

L

k=[]

fori**in**range(6541):

ifL[i]=="negative":

k.append(0)

elifL[i]=="positive":

k.append(1)

X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,k,

test\_size=0.3,

shuffle=True,

stratify=k,

random\_state=1)

**DEFINING THE CODE:**

embed\_dim=128

lstm\_out=196

model=tf.keras.models.Sequential()

model.add(tf.keras.layers.Embedding(max\_fatures,128,input\_length=X\_train.shape[1]))

model.add(tf.keras.layers.SpatialDropout1D(0.5))

model.add(tf.keras.layers.LSTM(196,dropout=0.3,recurrent\_dropout=0.3))

model.add(tf.keras.layers.Dropout(0.2))

model.add(tf.keras.layers.Dense(100,activation=tf.nn.relu))

model.add(tf.keras.layers.Dropout(0.4))

model.add(tf.keras.layers.Dense(2,activation=tf.nn.softmax))

model.compile(optimizer="adam",loss="sparse\_categorical\_crossentropy",metrics=["accuracy"])

TRAINING THE MODEL

Model=model.fit(X\_train,

Y\_train,

epochs=20,

batch\_size=32,

validation\_split=0.2,

verbose=2)

score=model.evaluate(X\_test,Y\_test,verbose=False)

print("loss = ",score[0])

print("accuracy = ",score[1])

**Plottiing the confusion matrix**

defplot\_confusion\_matrix(cm,classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.imshow(cm,interpolation='nearest',cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks=np.arange(len(classes))

plt.xticks(tick\_marks,classes,rotation=45)

plt.yticks(tick\_marks,classes)

ifnormalize:

cm=cm.astype('float')/cm.sum(axis=1)[:,np.newaxis]

thresh=cm.max()/2.

fori,j**in**itertools.product(range(cm.shape[0]),range(cm.shape[1])):

plt.text(j,i,cm[i,j],

horizontalalignment="center",

color="white"ifcm[i,j]>threshelse"black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

y\_pred=model.predict(X\_test)

y\_pred\_classes=np.argmax(y\_pred,axis=1)

confusion\_mtx=confusion\_matrix(Y\_test,y\_pred\_classes)

plot\_confusion\_matrix(confusion\_mtx,classes=range(2))

MODEL TEST:

sample=['Meetings: Air crew is so dumb.']

sample=tokenizer.texts\_to\_sequences(sample)

sample=pad\_sequences(sample,maxlen=31,dtype='int32',value=0)

print(sample)

sentiment=model.predict(sample,batch\_size=1,verbose=2)[0]

if(np.argmax(sentiment)==0):

print("negative")

elif(np.argmax(sentiment)==1):

print("positive")

**DATA SET:**

[**Tweets (1).xlsx**](Tweets%20(1).xlsx)

**CONCLUSION :**

Twitter sentiment analysis for airline marketing provides real-time insights, aids in service improvement, guides marketing strategies, and enables competitive benchmarking, but model accuracy, data privacy, and ethics are key considerations.