Sentiment Analysis For Marketing

Ensemble Methods:

Bagging**:** Techniques like Random Forest or Bagged Decision Trees can be employed to create an ensemble of sentiment classifiers. Each classifier is trained on a subset of the data, and their predictions are combined to produce a final sentiment score.

Boosting: Algorithms like AdaBoost or Gradient Boosting can improve sentiment analysis by giving more weight to misclassified data points in each iteration, leading to better overall accuracy.

Deep Learning Architectures:

Convolutional Neural Networks (CNNs): CNNs can be used for sentiment analysis by treating text as an image, converting words into vectors, and using convolutional layers to detect important features.

Recurrent Neural Networks (RNNs): RNNs, particularly LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) variants, are effective for sequence modeling, making them suitable for sentiment analysis tasks where the order of words matters.

Transformer Models: State-of-the-art models like BERT, GPT, and RoBERTa have revolutionized natural language understanding tasks, including sentiment analysis. They can capture context and nuances in text effectively.

Bagging:

I/N:

Accuracy = accuracy_score(y_test, y_pred)

Report = classification_report(y_test, y_pred)

Print(f”Accuracy: {accuracy}”)

Print(“Classification Report:\n”, report)

Boosting:

I/N:

Y_pred = ada_boost_classifier.predict(X_test_tfidf)

Accuracy = accuracy_score(y_test, y_pred)

Print(f’Accuracy: {accuracy:.2f}’)

Recurrent neural network:

I/N:

Model.compile(optimizer=’adam’, loss=’binary_crossentropy’, metrics=[‘accuracy’])

Model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))

Loss, accuracy = model.evaluate(X_test, y_test)

Print(f’Loss: {loss}, Accuracy: {accuracy}’)

Print(f”Text: {text}\nSentiment: {‘Positive’ if sentiment > 0.5 else ‘Negative’}”)

Convolution neural network:

I/N

Texts = [“This is a positive review.”, “Negative sentiment in this one.”, …]

Model = keras.Sequential

Test_loss, test_acc = model.evaluate(x_test, y_test)

Print(“Test accuracy:”, test_acc)

BERT:

I/N:

Model_name = “bert-base-uncased” # You can choose different BERT variants

Tokenizer = BertTokenizer.from_pretrained(Tweet)

Model = BertForSequenceClassification.from_pretrained(Tweet)

With torch.no_grad():

Outputs = model(**inputs)

Sentiment_labels = {0: “Negative”, 1: “Neutral”, 2: “Positive”}

Sentiment = sentiment_labels[predicted_label]

Text_to_analyze = “Positive”

Result = analyze_sentiment(text_to_analyze)

Print(f”Sentiment: {result}”)

RoBERTa:

I/N:

Tokenizer = RobertaTokenizer.from_pretrained(Tweet)

Model = RobertaForSequenceClassification.from_pretrained(Tweet)

Logits = outputs.logits

Sentiment_labels = [“Negative”, “Neutral”, “Positive”]

Sentiment = sentiment_labels[predicted_class]

Return sentiment, logits.tolist()

Text = “Positive”

Sentiment, scores = analyze_sentiment(Positive)

Print(f”Sentiment: {sentiment}”)

Print(f”Sentiment Scores: {scores}”)

GPT-2:

I/N:

Model_name = “gpt2” # You can also specify other variants like “gpt2-medium”, “gpt2-large”, etc.

Tokenizer = GPT2Tokenizer.from_pretrained(Tweet)

Model = GPT2LMHeadModel.from_pretrained(Tweet)

Input_ids = tokenizer.encode(prompt, return_tensors=”pt”)

```
Generated_text = tokenizer.decode(output[0],
skip_special_tokens=True)

Print(generated_text)


From mpl_toolkits.mplot3d import Axes3D

From sklearn.preprocessing import StandardScaler

Import matplotlib.pyplot as plt

Import numpy as np

Import os

Import pandas as pd

Print(os.listdir('../input'))

nRowsRead =

Df1 = pd.read_csv('../input/Tweets.csv', delimiter=',', nrows =
nRowsRead)

Df1.dataframeName = 'Tweets.csv'

nRow, nCol = df1.shape

print(f'There are {nRow} rows and {nCol} columns')
```

O/p

	Tweet_id	Airline_sentiment	AS_confidence	Negative_reason	airline
1	5787676523440432	neutral	1.0000	NaN	VirginAme
2	5766756565436587	positive	0.7843	Bad Flight	VirginAme
3 . .	5776532457688987	negative	0.9879	Can't tell	VirginAme