Experiment 6: Use a pre-trained Hugging Face model to analyze sentiment in text.

- Assume a Real-World Application,
- Load the **Sentiment Analysis** Pipeline.
- **Analyze** the sentiment by giving sentences to input.

AIM: To use a pre-trained Hugging Face model to analyze sentiment in text by loading the sentiment analysis pipeline and providing real-world input sentences.

Introduction:

- Sentiment analysis is one of the most common applications of **Natural Language Processing (NLP),** which helps machines understand and interpret human emotions from written text. It is widely used in real-world applications such as analyzing customer reviews, monitoring opinions on social media, and gaining insights into user satisfaction. For example, companies use sentiment analysis to understand how customers feel about their products or services.

- In this lab, we use the Hugging Face Transformers library, an open-source toolkit that provides access to thousands of pre-trained language models, including BERT, GPT, and DistilBERT. These models have already been trained on massive amounts of text data, so we can use them without training from scratch.

- To perform sentiment analysis, we need to convert raw text into a format that the model can understand. This process involves:

Tokenization: Breaking a sentence into smaller pieces called tokens (words or subwords).

Tokenizer: A tool that converts tokens into numerical IDs that the model can process.

Model: A pre-trained deep learning model that understands the structure and meaning of language and classifies the sentiment.

Pipeline: A simplified interface from Hugging Face that bundles the tokenizer and model into one easy-to-use function for a specific task like sentiment analysis. **Pipeline** bundles together preprocessing, model inference, and output formatting.

In this experiment, we use the ***distilbert-base-uncased-finetuned-sst-2-english*** model, which is a lighter and faster version of BERT. It has been fine-tuned specifically for sentiment classification using the Stanford Sentiment Treebank (SST-2) dataset. We also use **pandas**, a powerful Python library for displaying data in a structured table format.This lab introduces key AI/ML concepts while showing how easy it is to apply state-of-the-art NLP with just a few lines of code using Hugging Face.

Sentiment analysis is a Natural Language Processing (NLP) technique used to determine whether a piece of text is positive, negative, or neutral. It is widely used in real-world applications such as product reviews, social media monitoring, and customer feedback analysis.

In this experiment, we use the Hugging Face transformers library, which provides easy access to powerful pre-trained models for various NLP tasks. One of the simplest ways to perform sentiment analysis is by using the pipeline interface, which abstracts away the complexities of tokenization and model handling.

Key Concepts Introduced:

Hugging Face Transformers: An open-source library offering thousands of pre-trained models for NLP tasks.

Pre-trained Model: A machine learning model that has already been trained on a large dataset and can be reused.

DistilBERT: A smaller, faster version of BERT that retains most of its performance. We use the `distilbert-base-uncased-finetuned-sst-2-english` model, fine-tuned on sentiment classification.

By using the Hugging Face pipeline, we simplify the steps of loading the model, preparing the input, and interpreting the result—all done in a few lines of code.

Confidence Score in Sentiment Analysis.

In sentiment analysis using machine learning models, the confidence score indicates how sure the model is about its prediction.For example, if a review is classified as POSITIVE with a confidence score of 0.98, it means:The model is 98% confident that this sentence expresses positive sentiment.

Objectives of the Program

1. To understand and implement a pre-trained sentiment analysis model using Hugging Face.
2. To process and classify text into positive or negative sentiment.
3. To create a real-world dataset of customer reviews.
4. To display sentiment results in a structured table using pandas.
5. To perform overall sentiment aggregation and derive recommendation insights.

Step 1: Mount Google Drive to store the downloaded model

This step connects your Google Drive to Colab so you can store the model permanently and reuse it later.

```
from google.colab import drive
drive.mount('/content/drive')
```

Step 2: Create a directory in Google Drive to cache Hugging Face models

This sets up a folder in your Google Drive to store the model files.

```
import os

cache_dir = "/content/drive/MyDrive/transformers_cache"

os.makedirs(cache_dir, exist_ok=True)

os.environ['TRANSFORMERS_CACHE'] = cache_dir
```

Step 3: Install Hugging Face Transformers and Pandas

This installs the necessary libraries: transformers (for the model) and pandas (for table display).

```
!pip install transformers pandas --quiet
```

Step 4: Import Python libraries for NLP and data handling

This imports the Python libraries needed for model handling and data display.

```
import pandas as pd

from transformers import pipeline, AutoTokenizer, AutoModelForSequenceClassification
```

Step 5: Load pre-trained model and tokenizer from Hugging Face

```
model_name = "distilbert-base-uncased-finetuned-sst-2-english"

tokenizer = AutoTokenizer.from_pretrained(model_name, cache_dir=cache_dir)

model = AutoModelForSequenceClassification.from_pretrained(model_name, cache_dir=cache_dir)

# Create the sentiment analysis pipeline

sentiment_pipeline = pipeline(

    "sentiment-analysis",

    model=model,

    tokenizer=tokenizer

)
```

Step 6: Create a list of example product reviews (real-world application)

```
sample_reviews = [

    "I absolutely loved this product, it exceeded my expectations!",

    "Great experience, the product quality and delivery were excellent.",

    "Highly recommended! I’m very happy with the purchase.",

    "The design is sleek and the features work perfectly.",

    "Terrible experience. The product stopped working in two days.",

    "Not worth the money — very disappointed with the quality."

]
```

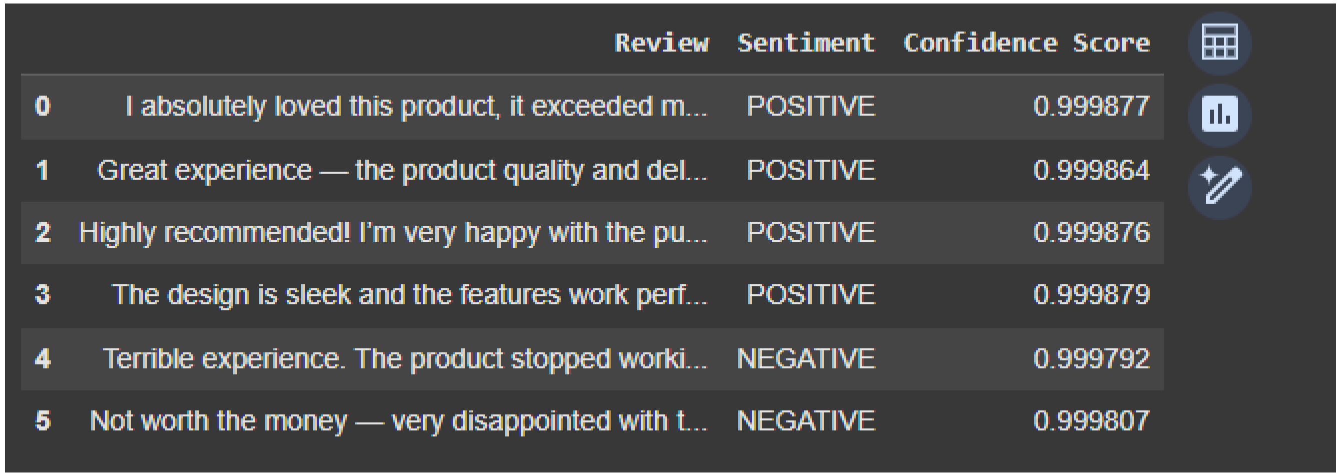
Step 7: Analyze sentiment of each review using the pipeline

```
sentiment_results = sentiment_pipeline(sample_reviews)
```

Step 8: Show results in a table with sentiment and confidence score

```
df_results = pd.DataFrame({
    "Review": sample_reviews,
    "Sentiment": [result["label"] for result in sentiment_results],
    "Confidence Score": [result["score"] for result in sentiment_results]
})
df_results
```

Output:



	Review	Sentiment	Confidence Score
0	I absolutely loved this product, it exceeded m...	POSITIVE	0.999877
1	Great experience — the product quality and del...	POSITIVE	0.999864
2	Highly recommended! I'm very happy with the pu...	POSITIVE	0.999876
3	The design is sleek and the features work perf...	POSITIVE	0.999879
4	Terrible experience. The product stopped worki...	NEGATIVE	0.999792
5	Not worth the money — very disappointed with t...	NEGATIVE	0.999807

Step 9: Count how many reviews are positive and negative

```
num_positive = sum(1 for res in sentiment_results if res["label"] == "POSITIVE")
num_negative = sum(1 for res in sentiment_results if res["label"] == "NEGATIVE")
total_reviews = len(sentiment_results)

positive_percentage = (num_positive / total_reviews) * 100
negative_percentage = (num_negative / total_reviews) * 100
```

```
# Determine overall sentiment
if num_positive > num_negative:
    overall_sentiment = "Positive"
    recommendation = "We recommend this product based on the positive reviews."
elif num_negative > num_positive:
    overall_sentiment = "Negative"
    recommendation = "We do not recommend this product based on the negative reviews."
else:
    overall_sentiment = "Mixed"
    recommendation = "The reviews are mixed. Consider additional factors before deciding."
```

Step 10: Print summary of the analysis

```
print("\n--- Overall Analysis ---")
print(f"Total Reviews Analyzed: {total_reviews}")
print(f"Positive Reviews: {num_positive} ({positive_percentage:.1f}%)")
print(f"Negative Reviews: {num_negative} ({negative_percentage:.1f}%)")
print(f"Overall Sentiment: {overall_sentiment}")
print(f"Recommendation: {recommendation}")
```

Output:

```
--- Overall Analysis ---

Total Reviews Analyzed: 6

Positive Reviews: 4 (66.7%)

Negative Reviews: 2 (33.3%)

Overall Sentiment: Positive

Recommendation: We recommend this product based on the positive reviews.
```