🛠️ Mini Project: Sentiment Analysis

Can we teach a machine to tell whether a sentence is positive, negative, or neutral?

```
# Step 1: Sample Data
reviews = [
    "This movie was amazing!",
    "Not bad, but not great either.",
    "Worst movie I've ever seen.",
    "Absolutely loved the story!",
    "I didn't like the acting.",
    "It was okay, nothing special.",
    "Totally blown away by the visuals.",
    "Terrible plot and weak script."
]

sentiments = [
    "Positive",
    "Neutral",
    "Negative",
    "Positive",
    "Negative",
    "Neutral",
    "Positive",
    "Negative"
]
```

Preprocess the Text

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(reviews)
```

🧠 Step 3: Train a Classifier

We'll keep it simple with Logistic Regression or Naive Bayes — both are great for text classification!

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, sentiments, test_size=0.2)

model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

🔍 Step 4: Evaluate Your Model

Use accuracy or even better: Confusion Matrix to see how well it performs.

```
from sklearn.metrics import classification_report, confusion_matrix

y_pred = model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[0 2]
 [0 0]]
              precision    recall  f1-score   support

    Negative       0.00      0.00      0.00       2.0
    Positive       0.00      0.00      0.00       0.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-d
```

```
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-d
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Recall is ill-d
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

🎉 Final Touch: Try Your Own Sentence!

```
sentence = ["I totally loved that scene!"]
sent_vec = vectorizer.transform(sentence)
print(model.predict(sent_vec))
```

```
['Positive']
```