```python
from typing import List, Dict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

class TextClassifier:
    def __init__(self, vectorizer):
        self.vectorizer = vectorizer
        self._model = None

    def fit(self, texts: List[str], labels: List[int]):
        X = self.vectorizer.fit_transform(texts)
        self._model = LogisticRegression(solver='liblinear')
        self._model.fit(X, labels)

    def predict(self, texts: List[str]) -> List[int]:
        X = self.vectorizer.transform(texts)
        return self._model.predict(X)

    def evaluate(self, y_true: List[int], y_pred: List[int]) -> Dict[str, float]:
        return {
            "accuracy": accuracy_score(y_true, y_pred),
            "precision": precision_score(y_true, y_pred),
            "recall": recall_score(y_true, y_pred),
            "f1_score": f1_score(y_true, y_pred)
    }
```

```python
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
# from src.models.text_classifier import TextClassifier

texts = [
    "This movie is fantastic and I love it!",
    "I hate this film, it's terrible.",
    "The acting was superb, a truly great experience.",
    "What a waste of time, absolutely boring.",
    "Highly recommend this, a masterpiece.",
    "Could not finish watching, so bad."
]
labels = [1, 0, 1, 0, 1, 0]

X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)

vectorizer = TfidfVectorizer()

classifier = TextClassifier(vectorizer)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

metrics = classifier.evaluate(y_test, y_pred)
print(metrics)
```

```
{'accuracy': 0.5, 'precision': 0.5, 'recall': 1.0, 'f1_score': 0.6666666666666666}
```

```python
from google.colab import files

uploaded = files.upload()
```

Chọn tệp sentiments.csv
**sentiments.csv**(text/csv) - 479938 bytes, last modified: 23/10/2025 - 100% done

```
Saving sentiments.csv to sentiments.csv
```

```python
from typing import List, Dict
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

class TextClassifier:
    def __init__(self, vectorizer):
        self.vectorizer = vectorizer
        self._model = None
    def fit(self, texts: List[str], labels: List[int]):
        X = self.vectorizer.fit_transform(texts)
        self._model = LogisticRegression(solver='liblinear', multi_class='auto')
        self._model.fit(X, labels)

    def predict(self, texts: List[str]) -> List[int]:
        X = self.vectorizer.transform(texts)
        return self._model.predict(X)

    def evaluate(self, y_true: List[int], y_pred: List[int]) -> Dict[str, float]:
        return {
            "accuracy": accuracy_score(y_true, y_pred),
            "precision_macro": precision_score(y_true, y_pred, average='macro'),
            "recall_macro": recall_score(y_true, y_pred, average='macro'),
            "f1_macro": f1_score(y_true, y_pred, average='macro')
        }
```

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

train = pd.read_csv("sent_train.csv")
valid = pd.read_csv("sent_valid.csv")

train = train.rename(columns={"text": "sentence"})
valid = valid.rename(columns={"text": "sentence"})

train = train.dropna(subset=["sentence", "label"])
valid = valid.dropna(subset=["sentence", "label"])

train["label"] = train["label"].astype(int)
valid["label"] = valid["label"].astype(int)

X_train = train["sentence"].tolist()
y_train = train["label"].tolist()

X_valid = valid["sentence"].tolist()
y_valid = valid["label"].tolist()

vectorizer = TfidfVectorizer(max_features=5000)
classifier = TextClassifier(vectorizer)

classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_valid)

metrics = classifier.evaluate(y_valid, y_pred)
print(metrics)
```

```
{'accuracy': 0.7902010050251256, 'precision_macro': 0.785494203048906, 'recall_macro': 0.6191089030595229, 'f1_macro': 0.6660357
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in v
  warnings.warn(
```

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
from pyspark.ml.feature import Tokenizer, StopWordsRemover, HashingTF, IDF
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

def main():
    spark = SparkSession.builder \
        .appName("SentimentAnalysis") \
        .getOrCreate()

    data_path = "sentiments.csv"
    df = spark.read.csv(data_path, header=True, inferSchema=True)

    df = df.withColumn("label", (col("sentiment").cast("integer") + 1) / 2)

    df = df.dropna(subset=["text", "label"])

    train_df, test_df = df.randomSplit([0.8, 0.2], seed=42)

    tokenizer = Tokenizer(inputCol="text", outputCol="words")
    remover = StopWordsRemover(inputCol="words", outputCol="filtered_words")
    hashingTF = HashingTF(inputCol="filtered_words", outputCol="raw_features", numFeatures=10000)
    idf = IDF(inputCol="raw_features", outputCol="features")

    lr = LogisticRegression(maxIter=20, regParam=0.01, featuresCol="features", labelCol="label")

    pipeline = Pipeline(stages=[tokenizer, remover, hashingTF, idf, lr])

    model = pipeline.fit(train_df)

    predictions = model.transform(test_df)

    evaluator = MulticlassClassificationEvaluator(metricName="accuracy", labelCol="label", predictionCol="prediction")
    accuracy = evaluator.evaluate(predictions)

    print(f"Accuracy: {accuracy:.4f}")

    spark.stop()


if __name__ == "__main__":
    main()
```

```
Accuracy: 0.7394
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

def test_model_improvement():

    df = pd.read_csv("sentiments.csv")
    df["label"] = (df["sentiment"] + 1) // 2
    df = df.dropna(subset=["text", "label"])

    X_train, X_test, y_train, y_test = train_test_split(
        df["text"], df["label"], test_size=0.2, random_state=42
    )

    vectorizer = TfidfVectorizer(stop_words="english", ngram_range=(1,2))
    X_train_vec = vectorizer.fit_transform(X_train)
    X_test_vec = vectorizer.transform(X_test)

    clf = MultinomialNB()
    clf.fit(X_train_vec, y_train)

    preds = clf.predict(X_test_vec)
    acc = accuracy_score(y_test, preds)

    print("\nAccuracy:", acc)
    print("\nClassification Report:\n", classification_report(y_test, preds))

    assert acc > 0.5

test_model_improvement()
```

```
Accuracy: 0.7066436583261432

Classification Report:
               precision    recall  f1-score   support

           0       0.88      0.23      0.37       427
           1       0.69      0.98      0.81       732

    accuracy                           0.71      1159
   macro avg       0.79      0.61      0.59      1159
weighted avg       0.76      0.71      0.65      1159
```