```
import re
import string
import pandas as pd
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 joblib

def clean_text(text):
    """
    Basic text cleaning:
    - Remove URLs, punctuation, and convert to lowercase.
    """
    # Remove URLs
    text = re.sub(r"http\S+|www\S+|https\S+", '', text, flags=re.MULTILINE)
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Lowercase
    text = text.lower()
    return text

def load_and_preprocess_data(csv_path):
    """
    Loads a CSV with 'article' as text and 'sentiment' as label.
    Returns the preprocessed text (X) and labels (y).
    """
    df = pd.read_csv(csv_path)

    # Rename columns to match expected format
    df = df.rename(columns={'article': 'text', 'sentiment': 'label'})

    # Basic cleaning
    df['cleaned_text'] = df['text'].apply(clean_text)

    # Return the text and label
    X = df['cleaned_text']
    y = df['label']
    return X, y

def train_sentiment_model(csv_path="sample_data.csv", model_out="sentiment_model.pkl"):
    """
    Train a Naive Bayes classifier on the provided dataset
    and save the model + TF-IDF vectorizer as .pkl files.
    """
    X, y = load_and_preprocess_data(csv_path)

    # Check if the dataset is too small for splitting
    if len(X) < 2:
        print("[WARNING] Not enough samples to perform train-test split. Training on full dataset.")
        X_train, y_train = X, y
        X_test, y_test = X, y
    else:
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Convert text to TF-IDF features
    vectorizer = TfidfVectorizer()
    X_train_tfidf = vectorizer.fit_transform(X_train)
    X_test_tfidf = vectorizer.transform(X_test)

    # Train classifier
    clf = MultinomialNB()
    clf.fit(X_train_tfidf, y_train)

    # Evaluate on test set
    y_pred = clf.predict(X_test_tfidf)
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy on test set: {:.2f}".format(accuracy))
    print("Classification Report:")
    print(classification_report(y_test, y_pred))

    # Save model and vectorizer
    joblib.dump(clf, model_out)
    joblib.dump(vectorizer, "vectorizer.pkl")
    print(f"[INFO] Model saved to {model_out}")
    print(f"[INFO] Vectorizer saved to vectorizer.pkl")
```

```
def load_sentiment_pipeline(model_path="sentiment_model.pkl", vectorizer_path="vectorizer.pkl"):
    """
    Utility function to load the trained model and vectorizer.
    Returns a prediction function.
    """
    clf = joblib.load(model_path)
    vectorizer = joblib.load(vectorizer_path)

    def predict_sentiment(text):
        """Predict sentiment for raw text input."""
        cleaned = clean_text(text)
        X_tfidf = vectorizer.transform([cleaned])
        return clf.predict(X_tfidf)[0]

    return predict_sentiment

if __name__ == "__main__":
    # Train the model (if running this file directly)
    train_sentiment_model(csv_path="sample_data.csv")
```

```
[WARNING] Not enough samples to perform train-test split. Training on full dataset.
Accuracy on test set: 1.00
Classification Report:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1

[INFO] Model saved to sentiment_model.pkl
[INFO] Vectorizer saved to vectorizer.pkl
```