```
import numpy as np, pandas as pd, matplotlib.pyplot as plt
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, confusion_matrix

# ==== Set these if auto-detect fails ====
# CSV_PATH  = ""customer_review.csv""
# TEXT_COL  = None   # e.g., "Review" or "Text"
# LABEL_COL = None   # e.g., "Sentiment" or "Score"
# ========================================

df = pd.read_csv("customer_review.csv")

def pick_col(df, exact, contains):
    cols = {c.lower(): c for c in df.columns}
    for e in exact:
        if e.lower() in cols: return cols[e.lower()]
    for c in df.columns:
        if any(sub in c.lower() for sub in contains): return c
    return None

text_col  = TEXT_COL  or pick_col(df, ["Review","Text","Summary","reviewText"],
["review","text","summary","body","content","comment"])
label_col = LABEL_COL or pick_col(df, ["Sentiment","Score","Rating","Stars","Label","overall"],
["sentiment","score","rating","star","label","polarity","class"])
assert text_col and label_col, "Set TEXT_COL and LABEL_COL to your column names."

X_raw = df[text_col].astype(str)

def map_label(v):
    # numeric ratings (e.g., 1–5)
    if pd.api.types.is_number(v): return "Negative" if v<=2 else ("Neutral" if v==3 else "Positive")
    t = str(v).strip().lower()
    if any(x in t for x in ["pos","good","great","5","4","excellent"]): return "Positive"
    if any(x in t for x in ["neu","neutral","3","mixed"]):              return "Neutral"
    if any(x in t for x in ["neg","bad","poor","1","2","terrible"]):    return "Negative"
    return np.nan

y_raw = df[label_col].apply(map_label)
keep = y_raw.notna() & X_raw.str.strip().ne("")
X, y = X_raw[keep], y_raw[keep]
```

```
print("Using columns -> Text:", text_col, "| Label:", label_col)
print("Class counts:\n", y.value_counts(), "\n")

# Split (stratify if possible)
try:
    Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
except ValueError:
    Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=42)

# TF-IDF + Naive Bayes
vec = TfidfVectorizer(stop_words="english", max_features=10000)
Xtrv, Xtev = vec.fit_transform(Xtr), vec.transform(Xte)
clf = MultinomialNB().fit(Xtrv, ytr)

# Evaluate
yp = clf.predict(Xtev)
print(f"Accuracy: {accuracy_score(yte, yp):.4f}\n")
print("Classification Report:\n", classification_report(yte, yp, digits=4, zero_division=0))

# Confusion matrix (plain matplotlib)
labels = [c for c in ["Negative","Neutral","Positive"] if c in y.unique()]
cm = confusion_matrix(yte, yp, labels=labels)
fig, ax = plt.subplots(figsize=(6,5))
im = ax.imshow(cm, interpolation="nearest"); ax.set_title("Confusion Matrix"); fig.colorbar(im)
ticks = np.arange(len(labels)); ax.set_xticks(ticks); ax.set_yticks(ticks)
ax.set_xticklabels(labels, rotation=45, ha="right"); ax.set_yticklabels(labels)
thr = cm.max()/2 if cm.max()>0 else .5
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, f"{cm[i,j]:d}", ha="center", va="center",
                color=("white" if cm[i,j]>thr else "black"))
ax.set_ylabel("Actual"); ax.set_xlabel("Predicted")
plt.tight_layout(); plt.show()
```