BE-NIG — News Insight Generator

Step-by-step project guide (A → Z) — Ready to follow

Prepared for: □□□□□ AIML Project (2000+ samples)

১. প্রকল্প সারসংক্ষেপ (Overview)

প্রকল্প নাম : BE-NIG — Bangla/English News Insight Generator। উদ্দেশ্য : একটি টেক্সট ইনপুট (খবরের হেডলাইন বা ছোট বডি) নিয়ে নিম্নলিখিত ইনফরমেশন জেনারেট করবে: • শ্রেণি (Category) — Politics, Sports, Tech, Business, Entertainment, Health, Education ইত্যাদি • ১–২ লাইনের সংক্ষিপ্ত সারসংক্ষেপ (Summary) • টপ কীওয়ার্ডস • ৩টি রিলেটেড /সদৃশ খবর (similar headlines) এই গাইডটি তোমাকে ডেটা সংগ্রহ থেকে শুরু করে মডেল ট্রেন , সেভ, এবং Streamlit ডেমো পর্যন্ত প্রতিটি ধাপে নিয়ে যাবে।

২. ডেলিভারেবলস (What you'll submit)

প্রস্তাবিত ডেলিভারেবলস : - notebooks/01_eda_clean.ipynb (EDA & cleaning) - notebooks/02_train_eval.ipynb (training + evaluation) - app/streamlit_app.py (ডেমো) - models/ (vectorizer, classifier, label encoder saved) - data/news_clean.parquet (cleaned dataset) - README.md এবং report.pdf (রিপোর্ট ও সেটআপ নির্দেশনা)

৩. ধাপে ধাপে করণের তালিকা (Step-by-step plan)

- Step 0 — পরিবেশ তৈরি (Environment): • Python 3.8+ ভার্চুয়াল এনভায়রনমেন্ট তৈরি করো এবং প্রয়োজনীয় প্যাকেজ ইনস্টল করো।

- Step 1 — ক্যাটেগরি নির্ধারণ (Decide categories): • পরামর্শ : Politics, Sports, Business, Tech, Entertainment, Health, Education — মোট 7টি • প্রতিটি ক্যাটেগরিতে লক্ষ্য : 250–400 স্যাম্পল → মোট 2000+

- Step 2 — ডেটা সংগ্রহ (Collect data): • Kaggle/news datasets অথবা নিজে scrape করে CSV তৈরি করো। • Minimum columns: id, title, text, category

- Step 3 — ডেটা পরিষ্কার (Data cleaning): • title এবং text কে একত্র করে clean_text() প্রয়োগ। • অপ্রয়োজনীয় whitespace, punctuation, digits, অতি ছোট টেক্সট রিমুভ।

- Step 4 — ভেক্টরাইজেশন (TF-IDF): • TfidfVectorizer(ngram_range=(1,2), min_df=2, max_df=0.9, sublinear_tf=True)

- Step 5 — মডেল ট্রেনিং (Train): • Stratified 80/20 split। • Model: Linear SVM (SGDClassifier(loss='hinge')) অথবা LogisticRegression • মূল্যায়ন : Accuracy, Macro F1, Confusion matrix

- Step 6 — ইনসাইট জেনারেশন : • Summary: TextRank (sumy) বা sentence selection • Keywords: TF-IDF top terms • Similar: cosine_similarity with corpus TF-IDF

- Step 7 — অ্যাপ (Demo): • Streamlit app: text input → category, summary, keywords, similar headlines

- Step 8 — রিপোর্ট ও মূল্যায়ন (Report): • Confusion matrix, per-class metrics, few error-analysis examples, limitations, future work

৪. ফাইল স্ট্রাকচার (Recommended folder layout)

project_root/ ├─ data/ │ └─ news_raw.csv (raw collected data) ├─ notebooks/ ├─ app/ │ ├─ streamlit_app.py │ └─ utils.py ├─ models/ ├─ requirements.txt └─ README.md

৫. কোড স্নিপেট — ডেটা ক্লিনিং ও মডেল ট্রেন (core.py)

```
import re, string, joblib, numpy as np, pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import SGDClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix
from joblib import dump
BN_PUNCT = "“”‘’—–…"
PUNCT_TABLE = str.maketrans("", "", string.punctuation + BN_PUNCT)
def clean_text(s):
    s = str(s)
    s = s.replace("\u200c"," ").replace("\u200d"," ")
    s = s.lower()
    s = re.sub(r"\d+", " ", s)
    s = s.translate(PUNCT_TABLE)
    s = re.sub(r"\s+", " ", s).strip()
    return s
df = pd.read_csv("data/news_raw.csv")  # id,title,text,category
df["text_clean"] = df["title"].fillna("") + ". " + df["text"].fillna("")
df["text_clean"] = df["text_clean"].apply(clean_text)
le = LabelEncoder()
y = le.fit_transform(df["category"])
X_train, X_test, y_train, y_test = train_test_split(
    df["text_clean"], y, test_size=0.2, stratify=y, random_state=42
)
pipe = Pipeline([
    ("tfidf", TfidfVectorizer(ngram_range=(1,2), min_df=2, max_df=0.9, sublinear_tf=True)),
    ("clf",   SGDClassifier(loss="hinge", alpha=1e-4, random_state=42))
])
pipe.fit(X_train, y_train)
pred = pipe.predict(X_test)
print(classification_report(y_test, pred, target_names=le.classes_))
dump(pipe, "models/news_clf.joblib")
dump(le, "models/label_encoder.joblib")
```

□. □□□ □□□□□□□ — □□□□□□ □□□□□□ (app/utils.py)

```
import numpy as np, pandas as pd, joblib
from sumy.summarizers.text_rank import TextRankSummarizer
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
pipe = joblib.load("models/news_clf.joblib")
le   = joblib.load("models/label_encoder.joblib")
tfidf_only = joblib.load("models/tfidf_only.joblib")
M = joblib.load("models/corpus_sparse.joblib")
meta = pd.read_parquet("data/corpus_meta.parquet")
def predict_info(text: str, topk_sim=3):
    proba = None
    if hasattr(pipe.named_steps["clf"], "predict_proba"):
        proba = pipe.predict_proba([text])[0]
    pred = pipe.predict([text])[0]
    cat  = le.inverse_transform([pred])[0]
    # summary
    try:
        parser = PlaintextParser.from_string(text, Tokenizer("english"))
        summ = TextRankSummarizer()
        sents = [str(s) for s in summ(parser.document, 2)]
        summary = " ".join(sents) if sents else text[:180]
    except:
        summary = text[:180]
    # keywords
    v = tfidf_only.transform([text])
    terms = np.array(tfidf_only.get_feature_names_out())
    top_idx = np.asarray(v.tocoo().col)
    top_data = np.asarray(v.tocoo().data)
    order = np.argsort(-top_data)[:5]
```

```
    keywords = terms[top_idx[order]].tolist()
    # similar docs
    from sklearn.metrics.pairwise import cosine_similarity
    sims = cosine_similarity(v, M).ravel()
    top = sims.argsort()[-topk_sim:][::-1]
    recs = meta.iloc[top].to_dict(orient="records")
    return cat, summary, keywords, proba, recs
```

□. Streamlit □□□□ (app/streamlit_app.py) — □□□ □□□□□□

```
import streamlit as st
from utils import predict_info
st.set_page_config(page_title="News Insight Generator", layout="centered")
st.title("■ BE-NIG — News Insight Generator")
text = st.text_area("Paste a news text/headline:", height=180)
if st.button("Analyze"):
    if not text.strip():
        st.warning("Please paste some text.")
    else:
        cat, summary, keywords, proba, recs = predict_info(text)
        st.success(f"Predicted Category: **{cat}**")
        st.write("**Summary:** ", summary)
        st.write("**Keywords:** ", ", ".join(keywords) if keywords else "-")
        st.subheader("Similar news")
        for r in recs:
            st.write(f"• {r['title']}  — _{r['category']}_")
```

□. □□□□□ □□□ □□□□□□ □ □□□□□□□□

```
# Virtual environment
python -m venv venv
source venv/bin/activate   # Windows: venv\Scripts\activate
pip install -r requirements.txt
# Run training script / notebook
python notebooks/core.py
# Run demo
streamlit run app/streamlit_app.py
```

□□□□ □□□□ □□□□ □□□□□□□□ □□□□□ (Immediate next steps): 1) □□□□□□□□□□□□□□ □□□□□□ □□□ (□□□□ □□□□□□□□□□ 7□□ □□□ □□□)□ 2) data/news_raw.csv □□□□ □□□□ □□□: columns = id,title,text,category 3) □□□ □□□, □□□ □□□□□ □□□□ □□□ sample dataset (100-300 □□□□) □□□□ □□□ □□□□ □□□□ 4) □□□□ '□□□ □□□□□□ □□□' □□□□□ □□□ training □□□ □□□□□□ sample □□□□ □□□□ □□□ □□□□□□□ □□ □□□□□□□ □□□□ A→Z □□□□ □□□ — □□□□□ □□□ □□□□ □□□□ sample dataset □ □□□□□□ □□□□□□□ □□□ □□□ □□□□ □□□□□□□