১. টেক্সট ক্লাসিফিকেশন কী? (What is Text Classification)

টেক্সট ক্লাসিফিকেশন হলো **NLP (Natural Language Processing)**-এর একটি গুরুত্বপূর্ণ কাজ, যেখানে কোনো লিখিত তথ্যকে তার অর্থ, বিষয়বস্তু ও প্রসঙ্গ বুঝে এক বা একাধিক পূর্বনির্ধারিত শ্রেণীতে (label/category) ভাগ করা হয়।

এখানে "টেক্সট" বলতে বোঝাতে পারে:

- একটি শব্দ
- একটি বাক্য
- একটি অনুচ্ছেদ
- একটি সম্পূর্ণ ডকুমেন্ট (যেমন: নিউজ আর্টিকেল, ইমেল, রিভিউ)

কেন টেক্সট ক্লাসিফিকেশন দরকার?

কম্পিউটার নিজে থেকে ভাষা বোঝে না। টেক্সট ক্লাসিফিকেশন কম্পিউটারকে শেখায়:

- লেখার মূল বিষয় কী
- লেখাটি কোন ধরনের
- লেখাটি কোন গ্রুপের মানুষ বা কাজের সাথে সম্পর্কিত

এটি কেন **Supervised Machine Learning?**

কারণ:

- মডেল শেখার সময় আগে থেকেই লেবেল দেয়া ডেটা থাকে
- উদাহরণ:
    - "This movie is amazing" → Positive
    - "Worst product ever" → Negative

এই ইনপুট + লেবেল দিয়েই মডেল শেখে কোন ধরণের লেখার সাথে কোন লেবেল যায়।

২. টেক্সট ক্লাসিফিকেশনের শ্রেণীর ধরন **(Types of Text Classification)**

২.১ বাইনারি ক্লাসিফিকেশন **(Binary Classification)**

সংজ্ঞা:
যখন সম্ভাব্য আউটপুট বা লেবেল মাত্র দুটি হয়, তখন সেটাকে বাইনারি ক্লাসিফিকেশন বলা হয়।

কেন **"Binary"** বলা হয়?

কারণ এটি অনেকটা **Yes/No**, **True/False**, **0/1** এর মতো।

বাস্তব উদাহরণ:

ইমেল স্প্যাম ডিটেকশন

ইমেল কনটেন্ট	আউটপুট
“Win a free iPhone now”	Spam
“Meeting at 10 AM tomorrow”	Not Spam

এখানে মডেল দেখে:

- সন্দেহজনক শব্দ আছে কি না
- লিংক বেশি আছে কি না
- আগে দেখা স্প্যাম শব্দের সাথে মিল আছে কি না

সব বিশ্লেষণ করে সিদ্ধান্ত নেয়: **Spam** না **Not Spam**

আরও উদাহরণ:

- রিভিউ → Positive / Negative
- SMS → Fraud / Not Fraud
- কমেন্ট → Toxic / Non-Toxic

২.২ মাল্টি-ক্লাস ক্লাসিফিকেশন **(Multi-class Classification)**

সংজ্ঞা:
যখন সম্ভাব্য লেবেল দুইয়ের বেশি, কিন্তু একটি টেক্সট শুধু একটি লেবেলেই পড়বে—তখন সেটি মাল্টি-ক্লাস ক্লাসিফিকেশন।

গুরুত্বপূর্ণ বিষয়:

- একাধিক অপশন থাকবে
- কিন্তু একটিই ফাইনাল আউটপুট

উদাহরণ: নিউজ ক্লাসিফিকেশন

ধরা যাক, ক্লাসগুলো হলো:

- খেলা
- বিজনেস
- রাজনীতি
- প্রযুক্তি

একটি নিউজ:

> "আজ শেয়ার বাজারে সূচক ২০০ পয়েন্ট বেড়েছে"

মডেল সিদ্ধান্ত নেবে:
→ বিজনেস

এটা একই সাথে "খেলা" বা "রাজনীতি" হবে না।

কেন এটা দরকার?

নিউজ ওয়েবসাইট বা অ্যাপগুলো:

- ইউজারকে সঠিক ক্যাটাগরিতে নিউজ দেখাতে পারে
- আলাদা আলাদা সেকশনে কন্টেন্ট সাজাতে পারে

আরও উদাহরণ:

- ইমেল টাইপ: Promotions / Social / Primary
- প্রশ্ন: Math / Physics / Biology
- ডকুমেন্ট: Legal / Medical / Technical

২.৩ মাল্টি-লেবেল ক্লাসিফিকেশন **(Multi-label Classification)**

সংজ্ঞা:
যখন একটি টেক্সট একই সাথে একাধিক লেবেল পেতে পারে, তখন সেটিকে মাল্টি-লেবেল ক্লাসিফিকেশন বলা হয়।

মাল্টি-ক্লাস থেকে পার্থক্য:

- মাল্টি-ক্লাস → একটি লেবেল
- মাল্টি-লেবেল → একাধিক লেবেল একসাথে

উদাহরণ:

একটি নিউজ:

> “শচীন টেন্ডুলকার আজ ক্রিকেট ইতিহাসে নতুন রেকর্ড গড়লেন”

সম্ভাব্য লেবেল:

- ক্রিকেট
- শচীন টেন্ডুলকার
- স্পোর্টস নিউজ

এখানে তিনটিই সঠিক একসাথে।

বাস্তব ব্যবহার:

- ইউটিউব ভিডিও ট্যাগিং
- ব্লগ পোস্ট ট্যাগ
- রিসার্চ পেপার ক্যাটাগরাইজেশন

টেকনিক্যাল পার্থক্য:

মাল্টি-লেবেলে:

- প্রতিটি লেবেল আলাদা বাইনারি ডিসিশন
- আউটপুট হতে পারে:
    - [1, 0, 1, 1]

---

৩. বাস্তব জীবনে টেক্সট ক্লাসিফিকেশনের ব্যবহার (Real-life Applications)

৩.১ ইমেল স্প্যাম ডিটেকশন

- লক্ষ লক্ষ ইমেল থেকে দরকারি ইমেল আলাদা করা
- ব্যবসায়িক নিরাপত্তার জন্য খুব গুরুত্বপূর্ণ

৩.২ সেন্টিমেন্ট অ্যানালাইসিস

রিভিউ বা কমেন্ট দেখে বোঝা:

- মানুষ খুশি না রাগান্বিত
- প্রোডাক্ট ভালো না খারাপ

উদাহরণ:

> “এই ফোনের ক্যামেরা খুবই বাজে”

আউটপুট:
→ Negative

৩.৩ কাস্টমার সাপোর্ট টিকেট রাউটিং

একটি ইমেল:

> “পেমেন্ট করার পরেও অর্ডার কনফার্ম হয়নি”

মডেল বুঝবে:
→ Billing / Payment Team

ফলে:

- সময় বাঁচে
- ভুল টিমে টিকেট যায় না

৩.৪ ভাষা শনাক্তকরণ **(Language Detection)**

একটি টেক্সট:

> “আমি আজ কলেজে যাব না”

মডেল আউটপুট:
→ Bengali

ব্যবহার:

- গুগল ট্রান্সলেট
- মাল্টিল্যাঙ্গুয়াল চ্যাটবট

৩.৫ ফেক নিউজ ও ম্যালিশিয়াস কন্টেন্ট ডিটেকশন

- ভুল তথ্য ছড়ানো আটকানো
- সোশ্যাল মিডিয়া সেফ রাখা

উদাহরণ:

> “এই ওষুধ খেলে একদিনে করোনা সেরে যায়”

মডেল চেক করে:
→ Fake / Harmful

৩.৬ টপিক বা ট্যাগ রিকমেন্ডেশন

একটি ব্লগ পোস্ট:

> “Python দিয়ে Machine Learning শেখার গাইড”

সম্ভাব্য ট্যাগ:

- Python
- Machine Learning
- Data Science

এটি মাল্টি-লেবেল ক্লাসিফিকেশনের একটি ক্লাসিক উদাহরণ।

ঠিক আছে। নিচে সাধারণ টেক্সট–ক্লাসিফিকেশন পাইপলাইনটা আমি খুব ডিটেইলসে, ধাপে ধাপে, সহজ ভাষায়, উদাহরণসহ বুঝিয়ে দিচ্ছি। তুমি যেন একেবারে শূন্য লেভেল থেকে পুরো স্লোটা কল্পনা করতে পারো—এইভাবে লেখা হয়েছে।

সাধারণ টেক্সট–ক্লাসিফিকেশন পাইপলাইন **(Step by Step)**

একটি টেক্সট ক্লাসিফিকেশন সিস্টেম বানাতে গেলে সাধারণত এই ৬টি ধাপ অনুসরণ করা হয়:

1. ডেটা সংগ্রহ
2. টেক্সট প্রি-প্রসেসিং
3. ভেক্টরাইজেশন
4. মডেলিং
5. ইভ্যালুয়েশন
6. ডিপ্লয়মেন্ট

এখন একে একে সবকিছু বিস্তারিতভাবে বোঝাই।

---

১. ডেটা সংগ্রহ **(Data Acquisition)**

ডেটা সংগ্রহ বলতে কী বোঝায়?

ডেটা সংগ্রহ মানে হলো মডেল শেখানোর জন্য টেক্সট ডেটা জোগাড় করা, যেখানে প্রতিটি টেক্সটের সাথে তার সঠিক লেবেল থাকবে।

মডেল শেখে উদাহরণ দেখে। উদাহরণ যত ভালো হবে, মডেল তত ভালো শিখবে।

ডেটার সাধারণ সোর্স

১. ওয়েব স্ক্র্যাপিং

ওয়েবসাইট থেকে লেখা সংগ্রহ করা।

- নিউজ ওয়েবসাইট
- ই-কমার্স রিভিউ
- ব্লগ পোস্ট

উদাহরণ:

- একটি ই-কমার্স সাইট থেকে হাজার হাজার প্রোডাক্ট রিভিউ স্ক্র্যাপ করা
- পরে সেগুলোকে Positive / Negative লেবেল দেয়া

---

২. **API**

অনেক কোম্পানি ডেটা অ্যাক্সেসের জন্য API দেয়।

- Twitter API → টুইট
- Reddit API → পোস্ট
- YouTube API → কমেন্ট

উদাহরণ:

- টুইট সংগ্রহ করে Sentiment Analysis করা

---

৩. লেবেল করা ডেটাসেট **(Ready-made Corpus)**

এগুলো আগে থেকেই পরিষ্কার ও লেবেল করা থাকে।

উদাহরণ:

- IMDB Movie Review Dataset → Positive / Negative

- AG News Dataset → News category
- Spam SMS Dataset → Spam / Ham

এগুলো শেখার জন্য সবচেয়ে ভালো।

---

৪. লোকাল ডাটাবেস / কোম্পানির ডেটা

- কাস্টমার ইমেল
- সাপোর্ট টিকেট
- চ্যাট লগ

বাস্তব প্রজেক্টে সাধারণত এখান থেকেই ডেটা আসে।

---

২. টেক্সট প্রি-প্রসেসিং **(Text Preprocessing)**

প্রি-প্রসেসিং কেন দরকার**?**

raw টেক্সট খুব নোংরা হয়:

- বড় হাতের অক্ষর
- অপ্রয়োজনীয় চিহ্ন
- বানানভেদ
- ফাঁকা শব্দ

এইসব থাকলে মডেল ভুল শেখে।

প্রি-প্রসেসিং মানে:
টেক্সট পরিষ্কার **+** একরকম করা

২.১ **Lowercase** করা

সব অক্ষর ছোট হাতের করা।

উদাহরণ:

- "Movie is GOOD"
- "movie is good"

কম্পিউটার এগুলো আলাদা শব্দ ভাবে, তাই lowercase করা জরুরি।

২.২ পাঙ্কচুয়েশন ও বিশেষ চিহ্ন সরানো

অপ্রয়োজনীয় জিনিস:

- ! @ # $ %
- , . ?
- extra symbols

উদাহরণ:

- "Great!!! movie??"
  → "great movie"

২.৩ টোকেনাইজেশন **(Tokenization)**

টেক্সটকে শব্দে ভাগ করা

উদাহরণ:

- "i love this movie"
  → ["i", "love", "this", "movie"]

এটাই NLP-এর ভিত্তি।

২.৪ স্টপওয়ার্ড রিমুভাল

স্টপওয়ার্ড মানে এমন শব্দ যেগুলো খুব বেশি ব্যবহৃত হয় কিন্তু অর্থ কম দেয়।

উদাহরণ:

- is, am, are
- the, this, that
- আমি, তুমি, এটা

উদাহরণ:

- ["i", "love", "this", "movie"]
  → ["love", "movie"]

সবসময় দরকার হয় না, তবে অনেক ক্ষেত্রে কাজের।

২.৫ স্টেমিং ও লেমমাটাইজেশন

স্টেমিং

শব্দ কেটে মূল অংশ রাখা।

- playing → play
- loved → love

লেমমাটাইজেশন

ব্যাকরণ অনুযায়ী মূল শব্দ বের করা।

- better → good

লেমমাটাইজেশন বেশি সঠিক, কিন্তু সময় বেশি লাগে।

২.৬ বাংলা বা ভাষা-নিরপেক্ষ কাজ

বাংলা টেক্সটে বিশেষ সমস্যা থাকে:

- যুক্তাক্ষর
- ভিন্ন ভিন্ন ইউনিকোড ফর্ম
- বিভক্তি (যাব, যাবে, যাচ্ছি)

এগুলো নরমালাইজ না করলে মডেল বিভ্রান্ত হয়।

---

৩. ভেক্টরাইজেশন (Text → Numbers)

কেন টেক্সটকে সংখ্যায় রূপান্তর করতে হয়?

মেশিন লার্নিং অ্যালগরিদম:

- শব্দ বোঝে না
- সংখ্যা বোঝে

তাই টেক্সট → নাম্বার করা বাধ্যতামূলক।

৩.১ **Bag of Words (BoW)**

ধারণা**:**

শব্দ আছে কি নেই, বা কতবার আছে—এটাই দেখা।

Vocabulary = ["i", "love", "movie"]

টেক্সট: "i love this movie"

ভেক্টর:

- i → 1
- love → 1
- movie → 1

Output: [1, 1, 1]

সমস্যা:

- শব্দের অর্থ বোঝে না
- শব্দের ক্রম বোঝে না

৩.২ **TF–IDF**

TF = Term Frequency
IDF = Inverse Document Frequency

সহজভাবে:

- যেসব শব্দ সব জায়গায় আছে, সেগুলোর গুরুত্ব কম
- যেসব শব্দ নির্দিষ্ট লেখায় গুরুত্বপূর্ণ, সেগুলোর ওজন বেশি

উদাহরণ:

- “movie” → কম গুরুত্বপূর্ণ
- “masterpiece” → বেশি গুরুত্বপূর্ণ

TF-IDF BoW-এর চেয়ে স্মার্ট।

৩.৩ **Word Embeddings (Word2Vec, GloVe)**

প্রতিটি শব্দকে একটি ঘন ভেক্টর দেওয়া হয়।

উদাহরণ:

- king − man + woman ≈ queen

মানে:
শব্দের অর্থ ও সম্পর্ক শেখে।

সমস্যা:

- context বুঝে না
- একই শব্দ সব জায়গায় একই ভেক্টর

৩.৪ **Contextual Embeddings (BERT, Transformer)**

এখানে শব্দের অর্থ বাক্যের উপর নির্ভর করে।

উদাহরণ:

- "bank of river"
- "bank account"

BERT বুঝতে পারে:

- একই শব্দ, কিন্তু অর্থ আলাদা

এই কারণেই BERT সবচেয়ে শক্তিশালী।

---

৪. মডেলিং **(Modeling)**

৪.১ **Traditional ML Models**

Naive Bayes

- খুব দ্রুত
- ছোট ডেটায় ভালো
- ব্যাসলাইন হিসেবে ব্যবহারযোগ্য

Logistic Regression

- TF-IDF-এর সাথে খুব ভালো কাজ করে
- সহজ ও ব্যাখ্যাযোগ্য

SVM

- টেক্সট ডেটায় শক্তিশালী
- উচ্চ-ডাইমেনশন ভালো হ্যান্ডেল করে

৪.২ **Deep Learning Models**

CNN

- লোকাল প্যাটার্ন ধরে
- ছোট বাক্যে ভালো

LSTM

- শব্দের ক্রম বোঝে
- লম্বা টেক্সটে ভালো

BERT

- context-aware
- state-of-the-art পারফরম্যান্স
- কিন্তু computationally expensive

কখন কোনটা নেব**?**

- ছোট ডেটা → Logistic Regression / SVM
- মাঝারি ডেটা → LSTM / CNN
- বড় ও জটিল → BERT

---

৫. ইভ্যালুয়েশন **(Evaluation)**

Accuracy

মোট কতটা ঠিক করেছে।

সমস্যা:

- imbalanced data-তে ভুল ধারণা দেয়।

Precision

যেগুলো Positive বলেছে, তার কতটা সত্যি।

Recall

আসল Positive-এর কতটা ধরতে পেরেছে।

F1-score

Precision ও Recall-এর ব্যালেন্স।

Confusion Matrix

ভুল কোথায় হচ্ছে সেটা বোঝার সবচেয়ে ভালো উপায়।

---

৬. ডিপ্লয়মেন্ট (Deployment)

কীভাবে ডিপ্লয় করা হয়?

- মডেলকে API বানানো
- Flask / FastAPI ব্যবহার
- Docker দিয়ে প্যাকেজ করা

ডিপ্লয়মেন্টে কী ভাবতে হয়?

- কত দ্রুত রেসপন্স দেয় (Latency)
- কত ইউজার একসাথে হ্যান্ডেল করবে (Scaling)
- নতুন ডেটা এলে আপডেট কিভাবে হবে

মডেল তুলনা — সংক্ষেপে

- Naive Bayes → দ্রুত, ছোট ডেটা
- Logistic / SVM → TF-IDF + টেক্সট

- Random Forest → হ্যান্ডক্রাফটেড ফিচার
- LSTM / CNN → সিকোয়েন্স শেখা
- BERT → সেরা পারফরম্যান্স, বেশি খরচ

ঠিক আছে। নিচে আমি এই পুরো অংশটা খুব সুন্দর করে, ধাপে ধাপে, সংখ্যা ও বাস্তব উদাহরণ দিয়ে, সহজ ভাষায় বুঝিয়ে দিচ্ছি—যেন Evaluation আর Practical দিকটা একদম পরিষ্কার হয়ে যায়।

---

১. ইভ্যালুয়েশন কী এবং কেন দরকার?

ইভ্যালুয়েশন মানে হলো—
আমাদের বানানো মডেল কতটা ভালো কাজ করছে সেটা মাপা।

শুধু "ঠিক করেছে / ভুল করেছে" জানলেই হয় না, কারণ:

- কিছু ক্ষেত্রে ভুল করা খুব ভয়ংকর
- কিছু ক্ষেত্রে ভুল সহনীয়

তাই আমরা Accuracy ছাড়াও **Precision, Recall, F1-score** ব্যবহার করি।

---

২. **Confusion Matrix —** সব কিছুর ভিত্তি

ধরা যাক, আমরা একটি **Spam Detection** মডেল বানিয়েছি।

বাস্তব অবস্থা	মডেল বলেছে	নাম
Spam	Spam	True Positive (TP)
Not Spam	Spam	False Positive (FP)

Spam	Not Spam	False Negative (FN)
Not Spam	Not Spam	True Negative (TN)

এখন তোমার দেয়া সংখ্যাগুলো ধরছি:

- TP = 70
- FP = 10
- FN = 20

---

৩. **Precision** — “যেগুলোকে **Positive** বলেছি, তার কতটা সত্যি?”

সংজ্ঞা

Precision বলে দেয়:

> মডেল যতগুলো Positive বলেছে, তার মধ্যে কয়টা আসলে Positive ছিল?

ফর্মুলা

Precision = TP / (TP + FP)

হিসাব

Precision = 70 / (70 + 10)

= 70 / 80

= 0.875

= 87.5%

বাস্তব অর্থ

মডেল যখন বলছে “এইটা Spam” —

- 87.5% সময় সে ঠিক বলছে
- 12.5% সময় ভুল করে ভালো ইমেলকে Spam বানাচ্ছে

Precision কখন গুরুত্বপূর্ণ?

- Spam Detection
- Fraud Detection
- Medical Diagnosis (ভুল Positive বিপদজনক)

---

৪. **Recall —** “আসল **Positive-**এর কতটা ধরতে পেরেছি?”

সংজ্ঞা

Recall বলে দেয়:

> আসল Positive যত ছিল, তার কতটা মডেল ধরতে পেরেছে?

ফর্মুলা

Recall = TP / (TP + FN)

হিসাব

Recall = 70 / (70 + 20)

= 70 / 90

≈ 0.7778

= 77.78%

বাস্তব অর্থ

সব Spam-এর মধ্যে:

- মডেল 77.78% ধরতে পেরেছে
- 22.22% Spam মিস করেছে (Inbox-এ চলে গেছে)

Recall কখন গুরুত্বপূর্ণ?

- Disease Detection
- Terrorist Content
- Fraud / Scam Detection

কারণ এখানে কিছু মিস হওয়া খুব বিপজ্জনক।

---

৫. **F1-score — Precision** আর **Recall-**এর ব্যালেন্স

অনেক সময়:

- Precision ভালো
- Recall খারাপ
  বা উল্টোটা

তাই দরকার একটা ব্যালেন্স স্কোর → F1-score

ফর্মুলা

F1 = 2 × (Precision × Recall) / (Precision + Recall)

হিসাব

F1 = 2 × (0.875 × 0.7778) / (0.875 + 0.7778)

≈ 2 × 0.6806 / 1.6528

≈ 0.8235

= 82.35%

বাস্তব অর্থ

মডেলটি:

- Precision এবং Recall দুটোই মোটামুটি ভালো ব্যালেন্সে রেখেছে

---

৬. **Precision** বাড়ালে কী হয়? **Recall** বাড়ালে কী হয়?

- Precision বাড়ালে → False Positive কমে
- Recall বাড়ালে → False Negative কমে

সব প্রজেক্টে একটাই লক্ষ্য হয় না—
Business problem বুঝে সিদ্ধান্ত নিতে হয়।

---

৭. ডেটা সমস্যা ও বাস্তব সমাধান

৭.১ **Imbalanced Dataset**

সমস্যা

এক ক্লাস অনেক বেশি, অন্যটা খুব কম।

উদাহরণ:

- 95% Not Spam

- 5% Spam

মডেল সবসময় “Not Spam” বললেই:

- Accuracy = 95%
  কিন্তু মডেল একদমই কাজের না।

সমাধান

- Oversampling (SMOTE) → কম ক্লাস বাড়ানো
- Undersampling → বেশি ক্লাস কমানো
- Class weights → ভুলের দাম বাড়ানো
- Stratified split → ট্রেন/টেস্টে ক্লাস ব্যালেন্স রাখা
- Focal loss → কঠিন উদাহরণে বেশি ফোকাস

৭.২ **Noisy Labels**

সমস্যা

ভুল লেবেল দেয়া ডেটা।

উদাহরণ:

- “This movie is amazing” → Negative (ভুল)

সমাধান

- ম্যানুয়াল রিভিউ
- সন্দেহজনক ডেটা বাদ দেয়া
- Active learning

৭.৩ **Domain Shift**

সমস্যা

ট্রেন ডেটা এক রকম, বাস্তব ডেটা আরেক রকম।

উদাহরণ:

- ট্রেন: English movie reviews

- টেস্ট: YouTube comments

সমাধান

- Transfer learning
- Fine-tuning
- Domain-specific data যোগ করা

৭.৪ **Short Text Problem**

সমস্যা

টেক্সট খুব ছোট।

উদাহরণ:

- “bad”
- “awesome”

Context কম, তথ্য কম।

সমাধান

- Pretrained embeddings
- Character n-grams
- Context-aware models (BERT)

---

৮. প্র্যাকটিক্যাল অ্যাডভাইস — বাস্তবে কাজ শুরু করলে

১. সবসময় **Baseline** বানাও

প্রথমে:

- TF-IDF + Logistic Regression

এটা দিয়ে বুঝবে:

- সমস্যা সহজ না কঠিন

---

২. **Feature Engineering** করো

- Word n-grams
- Character n-grams
- Domain keywords
- Sentiment lexicon

অনেক সময় মডেলের চেয়ে ফিচার বেশি কাজ করে।

---

৩. **Accuracy-**তে আটকে থেকো না

বিশেষ করে imbalanced ডেটায়:

- F1-score
- Macro-F1
  এইগুলো দেখো।

---

৪. **Cross-validation** ব্যবহার করো

একবার train-test split যথেষ্ট না।

- Stratified k-fold ব্যবহার করো
- ফলাফল stable হয়

---

৫. ডেটা বাড়ানোর চেষ্টা করো

- Weak supervision
- Distant supervision
- Manual labeling campaign

ডেটা বাড়লে মডেল ভালো হয়।

---

৬. প্রোডাকশন চিন্তা করো

- BERT ভালো কিন্তু ধীর
- DistilBERT / Small models অনেক সময় ভালো অপশন

- ONNX / Quantization latency কমায়

---

৭. বাংলা ও লোকাল ভাষার জন্য

- আলাদা tokenizer দরকার
- বাংলা stopwords আলাদা
- Pretrained Bengali models ব্যবহার করলে ফল ভালো

---

৯. ছোট প্র্যাকটিক্যাল কাজের পরিকল্পনা **(Beginner-friendly)**

১. ডেটা: IMDB বা নিজের রিভিউ
২. প্রি-প্রসেসিং: lowercase, tokenize
৩. ভেক্টরাইজেশন: TF-IDF (unigram + bigram)
৪. মডেল: Logistic Regression
৫. ইভ্যালুয়েশন: Precision, Recall, F1 (Stratified CV)
৬. উন্নতি: Word2Vec / BERT দিয়ে তুলনা