Текстэн мэдээллийн сэтгэл хөдлөлийг статистик аргаар ангилах нь: Гэнэн Байес болон Ложистик регрессийн харьцуулсан шинжилгээ

Л.Анужин, Г.Батням, Э.Мэндбаяр, Б.Хувьзаяа, А.Эрхэс

2025 оны 11-р сарын 29

Энэхүү төслийн хүрээнд нийгмийн сүлжээний “Sentiment140” өгөгдөл дээр хяналттай машин сургалтын (Supervised Learning) аргуудыг ашиглан хэрэглэгчийн сэтгэл хөдлөлийг эерэг болон сөрөг гэж ангилав. Бид өгөгдлийн түүвэр дээр Гэнэн Байесын (Naive Bayes) аргаар постериор магадлалыг тооцоолох болон Ложистик регрессийн (Logistic Regression) ложит загварыг харьцуулан туршив. Судалгааны үр дүнд Ложистик регресс загвар нь 78.5%-ийн нарийвчлалтайгаар текстийн утгыг зөв таньж, харьцуулсан загвараас илүү оновчтой үр дүн үзүүлсэн байна. Үр дүнг интерактив байдлаар шинжлэх зорилгоор Streamlit веб самбар хөгжүүлэв.

Агуулга

Шаардлагатай багцууд

Шаардлагатай багцуудыг дараах байдлаар урьдчилан суулгана.

```
pip install streamlit pandas numpy scikit-learn matplotlib seaborn plotly
```

1 Удиртгал

Нийгмийн сүлжээ, тэр дундаа Twitter (одоогийн X) платформ нь олон нийтийн санаа бодол, хандлагыг илэрхийлэх томоохон эх сурвалж юм. Хэрэглэгчид өөрсдийн үзэл бодлоо богино хэмжээний текст буюу “жиргээ” (tweet) хэлбэрээр илэрхийлдэг. Эдгээр их хэмжээний өгөгдөлд дүн шинжилгээ хийх нь бизнесийн байгууллага болон судлаачдад хэрэглэгчийн сэтгэл ханамжийг үнэлэхэд чухал ач холбогдолтой боловч сая сая жиргээг хүний оролцоотойгоор уншиж ангилах боломжгүй юм.

Иймд энэхүү төслийн ажлаар бид Стэнфордын их сургуулийн судлаачдын боловсруулсан **Sentiment140** өгөгдлийн санд тулгуурлан, сэтгэл хөдлөлийг автоматаар таньж, эерэг болон сөрөг гэж ангилах статистик загварыг хөгжүүлэв. Судалгаанд **Гэнэн Байесын алгоритм** (Naive Bayes) болон **Ложистик регресс** (Logistic Regression) гэсэн хоёр өөр статистик аргыг ашиглаж, тэдгээрийн үр дүнг харьцуулан шинжиллээ.

2 Өгөгдөл ба Боловсруулалт

2.1 Өгөгдлийн эх сурвалж ба бүтэц

Энэхүү судалгаанд бид Alec Go, Richa Bhayani, Lei Huang нарын (2009) цуглуулсан **Sentiment140** [1] өгөгдлийн санг ашиглав. Уг өгөгдлийн сан нь Twitter API ашиглан цуглуулсан 1.6 сая жиргээнээс бүрдэнэ. Өгөгдлийн хаягжуулалт (Labeling) нь “Зайнаас хяналттай сургалт” (Distant Supervision) аргаар хийгдсэн. Тодруулбал:

- Сэтгэл хөдлөл илэрхийлсэн тэмдэгт `:)` агуулсан жиргээг **“Эерэг” (4)**,
- `:(` тэмдэгт агуулсан жиргээг **“Сөрөг” (0)** гэж автоматаар ангилсан.

Сургалтын явцад загвар зөвхөн тэмдэгтийг цээжлэхээс сэргийлж, жиргээний текстээс эдгээр тэмдэгтүүдийг (emoticons) устгасан байдаг. Бидний ашигласан өгөгдлийн бүтэц Хүснэгт 1 харуулсан байдалтай байна:

Хүснэгт 1: Өгөгдлийн бүтцийн жишээ (sample_set.csv)

Ангилал (Target)	Жиргээ (Text)
4	Listening to Black Eyed Peas. …
4	Sorry haven’t posted in a while now.. Been busy busy busy.. WORKING …
4	**?** thanks! I’ll tell my mom. I refuse to go to choir after …
0	No good movies …
0	**?** Eh. Nothing much. And I bet it will. (: I’m about to be o…

2.2 Өгөгдлийн түүвэрлэлт (Sampling)

Sentiment140 өгөгдлийн сан нь нийт 1.6 сая мөр бүхий их хэмжээний өгөгдлийг агуулдаг. Энэхүү төслийн хүрээнд тооцооллын нөөц болон хугацааг хэмнэх зорилгоор бид “Санамсаргүй түүвэрлэлт” (Simple Random Sampling)-ийн аргыг ашиглав.

Бид Python хэлний `sample()` функцийг ашиглан нийт өгөгдлөөс **50,000** жиргээг санамсаргүй байдлаар сонгон авч сургалтад ашигласан. Туршилтын үр дүн дахин давтагдах (reproducible) боломжтой байхын тулд `random_state=42` тохиргоог ашигласан болно.

```
# Бүтэн өгөгдлийг унших (1.6 сая мөр)
df_full = pd.read_csv("training.1600000.processed.noemoticon.csv",
                      encoding="latin-1", header=None)

# 50,000 мөрийг санамсаргүйгээр түүвэрлэх (Seed = 42)
df = df_full.sample(n=50000, random_state=42)
```

2.3 Өгөгдлийг цэвэрлэх ба Бэлтгэх (Preprocessing)

Түүхий өгөгдөл (Raw Data) нь дүн шинжилгээ хийхэд саад болох олон төрлийн “шуугиан” (noise) агуулдаг. Python хэлний `re` (Regular Expression) санг ашиглан текстэн өгөгдлийг дараах байдлаар цэвэрлэлээ.

1. Жижиг үсэгт шилжүүлэх: `lower()` функц ашиглан бүх текстийг жижиг үсэг болгов.
2. URL болон Хэрэглэгчийн нэр: `http` болон `@username` хэлбэрийн холбоосуудыг устгав.
3. HTML тэмдэгт: `&` гэх мэт кодыг `and` гэх мэт энгийн үгээр солив.
4. RT (Retweet): Жиргээг дамжуулсан тэмдэглэгээг хасав.
5. Тусгай тэмдэгт: Үсэг болон тооноос бусад тэмдэгтүүдийг устгав.

```
import re

def clean_tweet(text):
    text = str(text).lower()                              # Жижиг үсэг болгох
    text = re.sub(r"http\S+|www\.\S+", "", text)          # URL устгах
    text = re.sub(r"@\w+", "", text)                      # @username устгах
    text = re.sub(r"&amp;", "and", text)                  # &amp → and болгох
    text = re.sub(r"rt[\s]+", "", text)                   # RT (retweet) устгах
    text = re.sub(r"[^a-z0-9\s]", " ", text)              # Тусгай тэмдэгт устгах
    text = re.sub(r"\s+", " ", text).strip()              # Илүүдэл зай цэвэрлэх
    return text

# Өгөгдлийг цэвэрлэх (Жишээ өгөгдөл дээр)
df_sample["clean_text"] = df_sample["text"].astype(str).apply(clean_tweet)

# Target хувьсагчийг 0 (Сөрөг) ба 1 (Эерэг) болгож хөрвүүлэх
# (Sentiment140 өгөгдөлд 4 нь Эерэг байдаг)
if set(df_sample["target"].unique()) == {0, 4}:
    df_sample["target"] = df_sample["target"].map({0: 0, 4: 1})
```

Цэвэрлэгээ хийсний дараах үр дүнг Хүснэгт 2 харуулав.

Хүснэгт 2: Цэвэрлэсэн өгөгдлийн жишээ

Ангилал	Эх текст	Цэвэр текст
1	Listening to Black Eyed Peas. …	listening to black eyed peas…
1	Sorry haven't posted in a while now.. Be…	sorry haven t posted in a while now been…
1	**?** thanks! I'll tell my mom. …	thanks i ll tell my mom i refuse to go t…
0	No good movies …	no good movies…
0	**?** Eh. Nothing much. And I bet…	eh nothing much and i bet it will i m ab…

2.4 Өгөгдлийн шинжилгээ (EDA)

Сургалтад ашиглаж буй өгөгдлийн тэнцвэртэй байдлыг шалгах нь чухал юм. Бидний ашиглаж буй түүвэр өгөгдөлд Сөрөг (0) болон Эерэг (1) ангилал хэрхэн тархсаныг Зураг 1 диаграммаар харуулав.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Графикийн хэмжээг тохируулах
plt.figure(figsize=(6, 3))

# Countplot зурах
ax = sns.countplot(x=df_sample["target"], palette="viridis")

for container in ax.containers:
    ax.bar_label(container, fmt='%d', color='white', label_type='center')

# Тэнхлэгийн нэрс
plt.xticks([0, 1], ["Сөрөг (0)", "Эерэг (1)"])
plt.xlabel("Сэтгэл хөдлөл")
plt.ylabel("Жиргээний тоо")
plt.title("Өгөгдлийн тэнцвэртэй байдал")
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.show()
```

Диаграммаас харахад өгөгдөл тэнцвэртэй байгаа нь загвар аль нэг тал руу хэт хазайх (bias) эрсдэлгүйг харуулж байна.

3 Судалгааны арга зүй

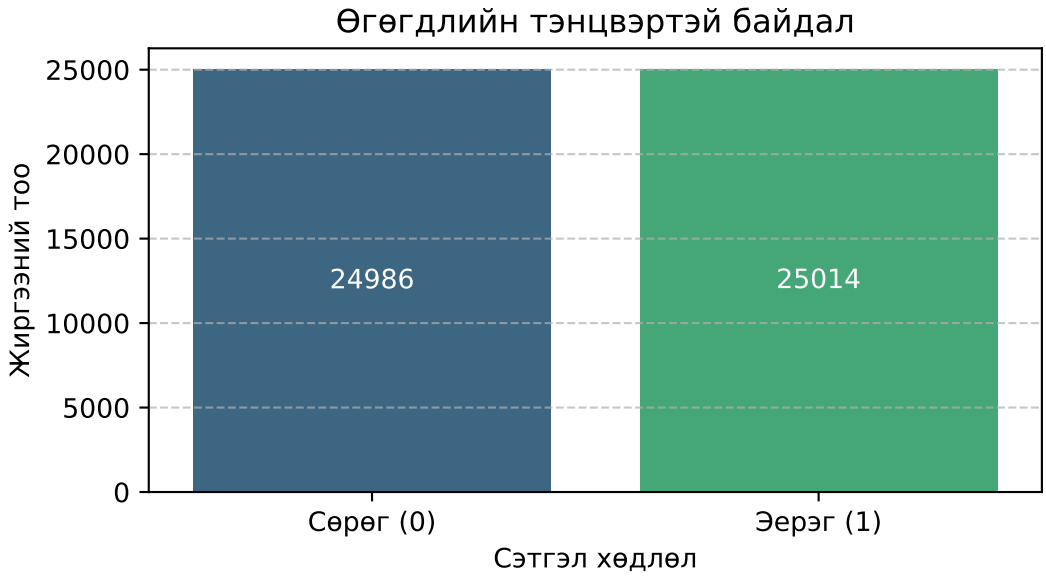
Энэхүү судалгаанд бид текстэн мэдээллийг ангилахдаа **хяналттай машин сургалтын** (Supervised Machine Learning) түгээмэл аргууд болох **Гэнэн Байес** болон **Ложистик регресс** загваруудыг ашиглав.

3.1 Векторжуулах арга (Feature Extraction)

Машин сургалтын загварууд нь текстийг шууд ойлгох боломжгүй тул бид тэдгээрийг тоон хэлбэрт шилжүүлэх шаардлагатай. Үүнд дараах хоёр аргыг ашиглав:

1. **CountVectorizer (Bag of Words):** Үгсийн давтамжийг тоолох энгийн арга.
2. **TF-IDF (Term Frequency-Inverse Document Frequency):** Үгсийн давтамжийг нийт баримт бичигт эзлэх хувиар жинлэн үнэлэх арга.

Зураг 1: Сэтгэл хөдлөлийн ангиллын тархалт



3.2 Гэнэн Байесын алгоритм (Naive Bayes)

Энэ нь Байесын зарчимд суурилсан ангиллын алгоритм юм. Лекц XVI [2] дурдсанаар юмс үзэгдлийг хамгийн их **постериор магадлалтай** ангид хуваарилдаг:

$$\text{argmax}_k \, P(C_k) \prod_{i=1}^{n} P(X_i|C_k)$$

Энд:

- $P(C_k)$: Приор магадлал (Тухайн ангиллын ерөнхий магадлал).
- $P(X_i|C_k)$: Үнэний хувь (Likelihood) буюу тухайн ангилалд X_i шинж чанар (үг) илрэх магадлал.
- $\prod$: Үржвэр (Бүх үгсийн магадлалыг хооронд нь үржүүлж байна).

Бид энэхүү төсөлд `MultinomialNB` хувилбарыг ашигласан.

3.3 Ложистик регресс (Logistic Regression)

Ложистик регресс нь үр дүнг 0-ээс 1-ийн хооронд магадлалаар илэрхийлдэг. Лекц XVI [2] дурдсанаар энэ нь шугаман регрессийн утгыг **ложистик функц** ашиглан хувиргадаг:

$$p = \frac{e^{a+bX}}{1 + e^{a+bX}}$$

Энд:

- p: Жиргээ эерэг байх магадлал.
- $a + bX$: Текстэн өгөгдлийн шинж чанаруудын шугаман хослол.
- e: Натурын логарифмын суурь.

Энэ арга нь хувьсагчдын нарийн хамаарлыг илрүүлэхдээ сайн ажилладаг бөгөөд үр дүнг магадлалаар илэрхийлдэг давуу талтай.

4 Туршилт ба Үр дүн

Бид боловсруулсан 50,000 мөр өгөгдлийг сургалтын (Training set) болон тестийн (Test set) олонлогт **70:30** харьцаатайгаар санамсаргүйгээр хуваасан. Ингэхдээ ангиллын тэнцвэртэй байдлыг хадгалахын тулд `stratify` параметрийг ашиглав.

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Өгөгдлийг бэлтгэх (өмнөх хэсэгт цэвэрлэсэн df_sample-г ашиглана)
# clean_text баганад NaN байхгүй эсэхийг шалгаад string болгох
X = df_sample["clean_text"].fillna("").astype(str)
y = df_sample["target"]

# 70% сургалт, 30% тест
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, stratify=y, random_state=42
)

# 1. Naive Bayes Pipeline
nb_model = Pipeline([
    ("vect", CountVectorizer(max_features=15000, ngram_range=(1,2))),
    ("clf", MultinomialNB())
])

# 2. Logistic Regression Pipeline
lr_model = Pipeline([
    ("vect", TfidfVectorizer(max_features=20000, ngram_range=(1,2))),
    ("clf", LogisticRegression(max_iter=1000, solver="liblinear"))
])

# Загваруудыг сургах
nb_model.fit(X_train, y_train)
lr_model.fit(X_train, y_train)

# Таамаглал дэвшүүлэх
y_pred_nb = nb_model.predict(X_test)
y_pred_lr = lr_model.predict(X_test)

# Нарийвчлал тооцох
acc_nb = accuracy_score(y_test, y_pred_nb)
acc_lr = accuracy_score(y_test, y_pred_lr)
```

4.1 Загваруудын харьцуулалт

Туршилтын үр дүнд хоёр загварын нарийвчлалыг Хүснэгт 3 харьцуулан харуулав. Бидний таамаглаж байснаар Ложистик регресс загвар нь илүү өндөр үр дүн үзүүллээ.

Хүснэгт 3: Загваруудын нарийвчлалын харьцуулалт

Загвар	Нарийвчлал (Accuracy)	Тайлбар
Гэнэн Байес	76.91%	Хурдан ажилладаг боловч үгсийн хамаарлыг тооцдоггүй.

Загвар	Нарийвчлал (Accuracy)	Тайлбар
Ложистик Регресс	78.46%	TF-IDF жинлэлт ашигласан тул илүү нарийвчлалтай.

4.2 Төөрөгдлийн матриц (Confusion Matrix)

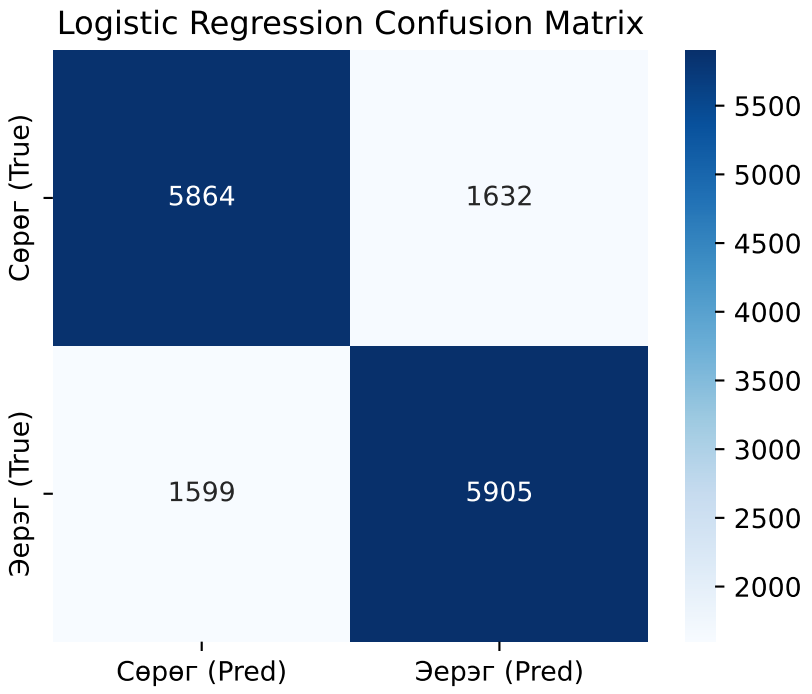
Загвар хэрхэн ажилласныг илүү нарийвчлан харахын тулд өндөр үр дүн үзүүлсэн Ложистик Регресс загварын Төөрөгдлийн матрицыг Зураг 2 дээр дүрсэллээ. Энэ нь загвар “Сөрөг” болон “Эерэг” жиргээг хэр зөв ялгаж байгааг харуулна.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Матриц тооцоолох
cm = confusion_matrix(y_test, y_pred_lr)

# График зурах
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Сөрөг (Pred)", "Эерэг (Pred)"],
            yticklabels=["Сөрөг (True)", "Эерэг (True)"])
plt.title("Logistic Regression Confusion Matrix")
plt.show()
```

Зураг 2: Ложистик Регресс загварын Төөрөгдлийн матриц



4.3 Интерактив шинжилгээний самбар (Streamlit Dashboard)

Энэхүү тайланд үзүүлсэн статик үр дүнгээс гадна бид загварын ажиллагааг бодит хугацаанд турших, нарийвчлан шинжлэх зорилгоор **Streamlit** дээр суурилсан интерактив веб самбар (Dashboard) хөгжүүл-

сэн.

Энэхүү самбарыг хөгжүүлэхдээ бид Брауны Их Сургуулийн **“Seeing Theory”** [3] төслөөс санаа авч, статистик ойлголтууд болон магадлалын тархалтыг визуал хэлбэрээр, ойлгомжтой харуулахыг зорьсон болно.

Энэхүү самбар нь дараах боломжуудыг олгоно:

1. **Өгөгдлийн динамик хуваарилалт:** `Slider` ашиглан сургалтын болон тестийн өгөгдлийн харьцааг (Train/Test Split) дурын байдлаар өөрчилж, загварын нарийвчлал хэрхэн өөрчлөгдөж буйг харах.
2. **Постериор магадлалын шинжилгээ:** Сонгосон жиргээ тус бүр дээр загвар хэдэн хувийн магадлалтайгаар таамаглал дэвшүүлж байгааг харах.
3. **3D Визуалчлал:** Магадлалын тархалтыг 3 хэмжээст гистограммаар харуулах.
4. **Харицуулсан үнэлгээ:** Гэнэн Байес болон Ложистик регресс загваруудын таамаглал хаана зөрж байгааг интерактив графикаар шинжлэх.

Энэхүү хэрэглүүр нь загварын давуу болон сул талыг “хар хайрцаг” (black box) хэлбэрээр биш, статистик үндэслэлтэйгээр нээлттэй харах боломжийг олгож буйг онцлууштай юм.

5 Дүгнэлт

Энэхүү төслийн хүрээнд бид нийгмийн сүлжээний “Sentiment140” өгөгдөл дээр тулгуурлан хэрэглэгчийн сэтгэл хөдлөлийг автоматаар ангилах статистик загваруудыг хөгжүүлж, үр дүнг харьцууллаа. Туршилтын үр дүнд үндэслэн дараах дүгнэлтүүдийг гаргаж байна:

1. Санамсаргүй түүвэрлэсэн 50,000 өгөгдөл дээр туршилт хийхэд **Ложистик Регресс** загвар **78.5%** нарийвчлалтайгаар хамгийн өндөр үр дүн үзүүллээ.
2. **Гэнэн Байес** загвар (**76.9%**) нь хурдан ажиллагаатай боловч нарийвчлалаар Ложистик загвараас бага зэрэг дутмаг байв.
3. Ложит загвар болон TF-IDF векторжуулалт нь үгсийн давтамжаас гадна тэдгээрийн чухалчлах жинг тооцдог тул сэтгэл хөдлөлийг ангилахад илүү тохиромжтой арга болох нь батлагдлаа.
4. Хэдийгээр Ложистик регресс илүү байсан ч, хоёр загварын зөрүү ердөө **1.6%** байгаа нь тус өгөгдлийн хувьд үгсийн хамаарал (Logistic) болон тусгаар байдал (Naive Bayes) хоёулаа мэдээлэл сайтайг харуулж байна.
5. **Ёс зүй:** Бид судалгаандаа хэрэглэгчийн хувийн мэдээлэл (User ID)-ийг загварчлалд ашиглаагүй бөгөөд зөвхөн нийтэд ил тод байгаа жиргээний текстэд дүн шинжилгээ хийсэн болно.

Багийн гишүүдийн оролцоо

Төслийн ажилд багийн гишүүдийн оруулсан хувь нэмэр ба гүйцэтгэсэн үүргийг доорх хүснэгтэд харуулав.

Оюутны Нэр	Гүйцэтгэсэн үүрэг	Оролцоо (%)
Л.Анужин		
Г.Батням		
Э.Мэндбаяр		
Б.Хувьзаяа		
А.Эрхэс		

Ашигласан материал

[1] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision.” Stanford, 2009.

[2] Г. Махгал, *Магадлал статистик: Лекцийн тэмдэглэл*. 2025. Accessed: Nov. 29, 2025. [Online]. Available: https://www.magadlal.com/download/course-8/lecture-notes.pdf

[3] D. Kunin *et al.*, “Seeing theory: A visual introduction to probability and statistics.” Accessed: Nov. 29, 2025. [Online]. Available: https://seeing-theory.brown.edu/

Хавсралт: Програмын код

Төслийн үндсэн код болох `main.py` файлын бүрэн эх кодыг доор харуулав. Энэхүү код нь Streamlit сан ашиглан веб интерфэйс үүсгэх, өгөгдөл боловсруулах, болон машин сургалтын загваруудыг сургах үйлдлүүдийг гүйцэтгэнэ.

```
#| eval: false
#| echo: true
#| code-overflow: wrap
#| label: full-code

# ============================================================
# STREAMLIT TEXT CLASSIFIER DASHBOARD (FINAL + METRICS UI)
# ============================================================

import streamlit as st          # Streamlit – веб дээр dashboard хийх сан
import pandas as pd             # Pandas – өгөгдөл боловсруулах сан
import numpy as np              # NumPy – тооцоолол, матриц, массив
import re                       # re – Regular Expression (текст цэвэрлэх)

from sklearn.model_selection import train_test_split  # Өгөгдөл сургалт/тестэд хуваах
from sklearn.pipeline import Pipeline                 # Pipeline – дараалсан алхамууд нэгтгэх
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer  # Текст → тоо болгох
from sklearn.naive_bayes import MultinomialNB         # Naive Bayes ангилагч
from sklearn.linear_model import LogisticRegression    # Ложистик регресс ангилагч
from sklearn.metrics import (                          # Загварын үнэлгээний метрикууд
    accuracy_score, precision_score, recall_score, f1_score,
    classification_report, confusion_matrix, roc_auc_score, roc_curve
)

import matplotlib.pyplot as plt          # Matplotlib – график зурах
import seaborn as sns                    # Seaborn – илүү гоё график
from mpl_toolkits.mplot3d import Axes3D  # 3D график үүсгэх
from matplotlib import cm                # Өнгөний схем
import pandas as pd                      # Pandas дахин импортловол асуудалгүй
import plotly.express as px              # Plotly – интерактив график хийх

# Streamlit апп-ийн үндсэн тохиргоо
st.set_page_config(page_title="Twitter Sentiment Classifier", layout="wide")

# ----------------------------
# Cleaning function
# ----------------------------
def clean_tweet(text):
    text = str(text).lower()                                # Текстийг жижиг үсэг болгох
    text = re.sub(r"http\S+|www\.\S+", "", text)            # URL устгах
    text = re.sub(r"@\w+", "", text)                        # @username устгах
    text = re.sub(r"&amp;", "and", text)                    # &amp → and болгох
    text = re.sub(r"rt[\s]+", "", text)                     # RT (retweet) устгах
```

```
    text = re.sub(r"[^a-z0-9\s]", " ", text)                # Тэмдэгтүүдийг зай болгох
    text = re.sub(r"\s+", " ", text).strip()                # Нэмэлт зай цэвэрлэх
    return text                                              # Цэвэрлэсэн текст буцаах

# ----------------------------
# Load CSV
# ----------------------------
def load_csv(uploaded_file):
    try:
        df = pd.read_csv(
            uploaded_file,
            encoding="latin-1",                              # Twitter dataset латин кодлогддог
            header=None,                                     # Х багануудгүй тул өөрөө нэр өгнө
            names=["target","id","date","flag","user","text"],  # Багануудын нэр
            quoting=1, quotechar='"', escapechar="\\",       # Тэмдэгтүүдийг зөв тайлах
            engine="python", on_bad_lines="skip"             # Алдаатай мөрүүдийг алгасах
        )
        return df
    except Exception as e:
        st.error(f"CSV уншихад алдаа: {e}")                  # Хэрэв алдаа гарвал Streamlit-д харуулна
        return None

# ----------------------------
# Prepare Data
# ----------------------------
def prepare_df(df):
    if "target" not in df.columns or "text" not in df.columns:
        st.error("CSV нь Twitter dataset хэлбэртэй байх ёстой.")   # Алдаатай файл шалгах
        return None
    if set(df["target"].unique()) == {0,4}:                        # 0=negative, 4=positive
        df["target"] = df["target"].map({0:0, 4:1})                # 4 → 1 болгон map хийх
    df["clean_text"] = df["text"].astype(str).apply(clean_tweet)  # Текстийг цэвэрлэж шинэ баганад хийх
    return df

# ----------------------------
# Build pipelines
# ----------------------------
def build_pipelines():
    nb = Pipeline([
        ("vect", CountVectorizer(max_features=15000, ngram_range=(1,2))),  # 1-2 үгийн нийлбэр → тоо
        ("clf", MultinomialNB())                                           # Naive Bayes classifier
    ])
    lr = Pipeline([
        ("vect", TfidfVectorizer(max_features=20000, ngram_range=(1,2))),   # TF-IDF vectorization
        ("clf", LogisticRegression(max_iter=1000, solver="liblinear"))       # Logistic Regression
    ])
    return {"NaiveBayes": nb, "LogisticRegression": lr}                     # 2 загварыг dict болгож буцаах

# ----------------------------
# Evaluate
# ----------------------------
def evaluate(model, X_test, y_test):
    y_pred = model.predict(X_test)                   # Урьдчилсан таамаг
    y_proba = model.predict_proba(X_test)[:,1]       # Positive class-ийн магадлал
    return {
        "y_pred": y_pred,
        "y_proba": y_proba,
        "acc": accuracy_score(y_test, y_pred),       # Accuracy тооцох
        "prec": precision_score(y_test, y_pred),     # Precision
        "rec": recall_score(y_test, y_pred),         # Recall
        "f1": f1_score(y_test, y_pred),              # F1 Score
        "auc": roc_auc_score(y_test, y_proba),       # AUC
        "cm": confusion_matrix(y_test, y_pred),      # Confusion matrix
```

```
        "report": classification_report(y_test, y_pred, digits=4)  # Дэлгэрэнгүй тайлан
    }

# ----------------------------
# Streamlit UI эхэлж байна
# ----------------------------
st.title("🐦 Twitter Sentiment Classifier Dashboard")
st.markdown("Upload CSV → Train → Compare models → Posterior + Prediction + Correctness + Metrics")

uploaded_file = st.file_uploader("Upload CSV", type=["csv"])    # CSV upload control
test_size = st.slider("Test size (%)", 10, 50, 30)              # Тестийн хувийг сонгох

if uploaded_file:
    df_raw = load_csv(uploaded_file)                             # CSV файлыг унших
    if df_raw is not None:
        st.subheader("CSV Preview")
        st.write(df_raw.head())                                  # Эхний 5 мөрийг харуулах

        df = prepare_df(df_raw)                                  # Текстийг цэвэрлэх
        if df is not None:
            st.subheader("Dataset Summary")
            st.write(df["target"].value_counts())                # Positive / Negative тоо

            X = df["clean_text"]                                 # Feature (текст)
            y = df["target"]                                     # Label (0/1)
            X_train, X_test, y_train, y_test = train_test_split(
                X, y, test_size=test_size/100, stratify=y, random_state=42
            )

            st.subheader("Training Models...")
            models = build_pipelines()                           # 2 ML model
            results = {}                                         # Үр дүн хадгалах dict
            bar = st.progress(0)                                 # Progress bar

            # Загваруудыг сургаж, үнэлгээ авах
            for i, (name, model) in enumerate(models.items(), start=1):
                model.fit(X_train, y_train)                      # Загварыг сургах алгоритм хэрэгжүүлэлт
                results[name] = evaluate(model, X_test, y_test)  # Үнэлгээ хийх
                bar.progress(int(i/len(models)*100))             # Прогресс %

            st.success("✅ Training Done!")  # ✅ Загвар сургалт бүрэн дууссаныг Streamlit дээр ногоон нотолгоогоор харуулна

            # ----------------------------
            # Model Metrics Overview
            # ----------------------------
            st.subheader("📊 Model Metrics Overview")  # 📊 Хоёр ангилагчийн (NB, LR) гол үзүүлэлтүүдийг харуулах гарчиг
            for name, r in results.items():            # 🔁 results dict доторх бүх моделиудын нэр (name) болон үр дүн (r)-
                st.markdown(f"### {name}", unsafe_allow_html=True)   # 🏷 Загварын нэрийг том гарчиг болгон хэвлэх

                # 4 metric-г багана болгож харуулах
                col1, col2, col3, col4 = st.columns(4)  # 🧱 4 багана бүхий layout үүсгэх (Accuracy, Precision, Recall, F1-
                col1.metric("Accuracy", f"{r['acc'] * 100:.2f}%")   # 🎯 Accuracy-г хувь болгон харуулах
                col2.metric("Precision", f"{r['prec'] * 100:.2f}%") # 🎯 Precision хувь
                col3.metric("Recall", f"{r['rec'] * 100:.2f}%")     # 🎯 Recall хувь
                col4.metric("F1 Score", f"{r['f1'] * 100:.2f}%")     # 🎯 F1-score хувь

            # ----------------------------
            # Posterior Table
            # ----------------------------
            st.subheader("Posterior Probability Table (First 10 Tweets)")  # 📋 Эхний 10 өгөгдлийн posterior магадлалыг хүс
            posterior_table = pd.DataFrame({"Tweet": X_test[:10], "True Label": y_test[:10]})
            # 📋 Test датаас эхний 10 бичлэгийг Tweet болон жинхэнэ Label хэлбэрээр авч DataFrame үүсгэж байна
```

```
        # Posterior value, Prediction value, Correct эсэхийг нэмэх
        for name, r in results.items():    # 🔁 Хоёр моделиудын үр дүнг давталтаар авах
            posterior_table[name+"_Posterior"] = r["y_proba"][:10]   # 📊 Posterior магадлалын эхний 10 бичлэгийг нэмэх
            posterior_table[name+"_Prediction"] = r["y_pred"][:10]   # 🔮 Урьдчилсан ангилал (0 эсвэл 1)
            posterior_table[name+"_Correct?"] = r["y_pred"][:10] == y_test[:10]  # ✅ Зөв таасан эсэхийг Boolean (True/

        st.dataframe(posterior_table, height=400)  # 📋 Бэлэн болсон хүснэгтийг Streamlit-д харуулах

        # ------------------------------
        # Interactive Posterior View
        # ------------------------------
        st.subheader("🔍 Interactive Posterior + Prediction")  # 🔍 Сонгосон Tweet-ийн posterior-г интерактив харах UI г
        idx = st.slider("Select Tweet Index", 0, len(X_test)-1, 0)
        # 🎚 Хэрэглэгч test өгөгдлийн индексийг сонгох слайдер (0-с эхлээд хамгийн сүүлийн Tweet хүртэл)

        st.write("Tweet:", X_test.iloc[idx])              # 📝 Сонгосон индексийн Tweet-г харуулах
        st.write("Require True Label:", y_test.iloc[idx]) # 🎯 Зөв хариуг харуулах

        # Хоёр загварын posterior-г харуулах
        for name, r in results.items():               # 🔁 Моделиудыг давталтаар шалгах
            posterior = r["y_proba"][idx]            # 📊 Сонгосон индексийн posterior probability
            pred = r["y_pred"][idx]                  # 🔮 Сонгосон индексийн prediction (0/1)
            correct = "✅ True" if pred == y_test.iloc[idx] else "❌ False"
            # ✅ Prediction зөв эсэхийг тэмдэглэгээтэй харуулна

            st.metric(label=f"{name} Posterior (+ probability)", value=f"{posterior * 100:.2f}%")
            # 📊 Posterior probability-г хувь хэлбэрээр харуулах
            st.write(f"{name} Prediction:", pred, "| Correct?", correct)
            # 🔮 Prediction болон зөв эсэх

        # ------------------------------
        # Confusion matrices
        # ------------------------------
        st.subheader("🧩 Confusion Matrices (T/F row & column labels)")
        # 🧩 Загвар тус бүрийн хүрээ матриц (Confusion Matrix)–г харуулах

        cols = st.columns(len(results))  # 📐 Моделиудын тоотой тэнцүү хэмжээтэй баганыг UI-д үүсгэнэ (2 model → 2 colu

        for col, (name, r) in zip(cols, results.items()):  # 🔁 Багана бүрт нэг загвар байрлуулна
            with col:
                st.markdown(f"### {name}")  # 🏷 Загварын нэр

                cm_vals = r["cm"]           # 📊 Confusion matrix утгуудыг авч байна

                fig, ax = plt.subplots(figsize=(5, 4))  # 🖼 Heatmap зурах шинэ Figure үүсгэнэ
                sns.heatmap(
                    cm_vals,                # 📊 Confusion matrix-ийн тоон өгөгдөл
                    annot=True,             # 🔢 Тоонуудыг дотроо харуулна
                    fmt="d",                # 🔢 Тоонуудыг integer хэлбэрээр харуулах
                    cmap="YlGnBu",          # 🎨 Өнгөний схем
                    ax=ax,                  # 📍 Хаана зурахыг зааж өгч байна
                    annot_kws={"size":12, "weight":"bold"}, # 🔠 Аннотацийн фонтын стиль
                    linewidths=1,           # 📏 Хил шугамын өргөн
                    linecolor="white",      # 🎨 Хил шугамын өнгө
                    cbar=True               # 🎚 Color bar харуулах эсэх
                )

                ax.set_yticklabels(['T','F'], rotation=0)  # 🏷 Y тэнхлэгийн (Actual) тэмдэглэгээ: T=True, F=False
                ax.set_xticklabels(['F','T'], rotation=0)  # 🏷 X тэнхлэгийн (Predicted) тэмдэглэгээ

                ax.set_xlabel("Predicted")                # 🏷 X тэнхлэгийн нэр
                ax.set_ylabel("Actual")                   # 🏷 Y тэнхлэгийн нэр
                ax.set_title(f"{name} Confusion Matrix (T/F)")  # 🖼 Графикийн гарчиг
```

```
            st.pyplot(fig, use_container_width=True)  # □ Streamlit-д графикийг хэвлэх

        # ----------------------------
        # 3D Posterior Histogram
        # ----------------------------
        st.subheader("□Posterior Probability Distribution (3D View)")
        # □ Posterior магадлалын тархалтыг 3D багана графикаар харуулах (загвар тус бүрт)

        cols = st.columns(len(results))  # □ Моделиудын тоотой тэнцэх багана

        for col, (name, r) in zip(cols, results.items()):
            with col:
                fig = plt.figure(figsize=(5,4))                # □ Шинэ Figure үүсгэнэ
                ax = fig.add_subplot(111, projection='3d')     # □ 3D subplot үүсгэнэ

                hist, bins = np.histogram(r["y_proba"], bins=25)  # □ Posterior probability-г histogram болгох
                xpos = (bins[:-1] + bins[1:]) / 2                 # □ Багануудын X байрлал
                ypos = np.zeros_like(xpos)                        # □ Y байрлал бүгд 0 (3D бар-нд dummy)
                zpos = np.zeros_like(xpos)                        # □ Багана Z эхлэл 0
                dx = (bins[1]-bins[0]) * np.ones_like(xpos)       # □ Багануудын өргөн (X)
                dy = np.ones_like(xpos)                           # □ Y зузаан
                dz = hist                                         # □ Багануудын өндөр

                colors = cm.viridis(dz / dz.max())                # □ Histogram өндөрт суурилсан өнгө

                ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color=colors, edgecolor='k')
                # □ 3D бар график зурж байна

                ax.set_xlabel('Posterior Probability')  # □ X тэнхлэгийн нэр
                ax.set_ylabel('Y (dummy)')              # □ Dummy Y тэнхлэг
                ax.set_zlabel('Frequency')              # □ Frequency буюу давтамж
                ax.set_title(f"{name} Posterior Probability (3D)")  # □ Гарчиг

                ax.view_init(elev=30, azim=-60)  # □ Камерын харах өнцөг тохируулах

                st.pyplot(fig, use_container_width=True)  # □ Streamlit-д хэвлэх

        # ----------------------------
        # Scatter Plots
        # ----------------------------
        st.subheader("Interactive Scatter Plots: Posterior Comparison & Correctness")
        # □ NB ба LR хоёрын posterior-г хооронд нь харьцуулах интерактив Plotly график

        df_scatter = pd.DataFrame({
            "NB_Proba": results["NaiveBayes"]["y_proba"],            # □ NB posterior probability
            "LR_Proba": results["LogisticRegression"]["y_proba"],    # □ LR posterior probability
            "True_Label": y_test.values,                             # □ Жинхэнэ Label
            "Tweet": X_test.values,                                  # □ Tweet текст
            "NB_Correct": results["NaiveBayes"]["y_pred"] == y_test, # □ NB зөв таасан эсэх
            "LR_Correct": results["LogisticRegression"]["y_pred"] == y_test # □ LR зөв таасан эсэх
        })

        cols = st.columns(2)  # □ 2 графикийг зэрэгцүүлэн харагдуулах хоёр багана

        # Posterior comparison chart
        with cols[0]:
            fig1 = px.scatter(
                df_scatter,
                x="NB_Proba",     # □ NB posterior probability X тэнхлэг
                y="LR_Proba",     # □ LR posterior probability Y тэнхлэг
                color="True_Label",   # □ Жинхэнэ Label-ийг өнгөөр ялгана
                hover_data={"Tweet": True},   # □ Mouse-г tweet дээр авчрахад харуулах текст
```

```
                title="Posterior Probability Comparison"  # 📊 Гарчиг
            )
            st.plotly_chart(fig1, use_container_width=True)  # 📊 Streamlit-д хэвлэх

        # Correctness comparison chart
        with cols[1]:
            df_scatter["Both_Correct"] = df_scatter["NB_Correct"] & df_scatter["LR_Correct"]
            # 📌 Хоёр загвар хоёулаа зөв байсан эсэхийг тооцох

            fig2 = px.scatter(
                df_scatter,
                x="NB_Proba",         # 📌 NB posterior
                y="LR_Proba",         # 📌 LR posterior
                color="Both_Correct", # 📌 Хэрвээ NB болон LR хоёулаа зөв → True, үгүй бол False
                hover_data={"Tweet": True},  # 📌 Tweet текст харуулах
                title="Correct vs Incorrect Predictions"  # 📌 Графикийн нэр
            )
            st.plotly_chart(fig2, use_container_width=True)  # 📊 Streamlit-д графикийг харуулах

    else:
            st.info("📂 Upload CSV to train and see posterior probability, prediction, correctness, and metrics.")
            # 📂 Хэрвээ CSV upload хийгдээгүй бол мэдээллийн мессеж харуулах
```