Скам/Спам сэтгэгдэл танигч

С.Буян-Эрдэнэ О.Нямбаяр М.Ариунзаяа Б.Эрдэнэ-Очир

2025 оны 12-р сарын 5

?abstractname?

Энэхүү төслийн ажилд Монгол хэл дээр бичигдсэн Facebook сэтгэгдлийг ашиглан тухайн сэтгэгдэл spam эсэхийг ангилах машин сургалтын загвар боловсруулах бүх үе шатыг тусгав. Үүнд өгөгдөл цуглуулалт, урьдчилсан цэвэрлэгээ, шинжилгээний аргачлал, шинж тэмдгийн боловсруулалт, мөн сургасан моделуудын бүтэц, ажиллах зарчим болон гүйцэтгэлийн үнэлгээ зэрэг гол хэсгүүд багтана.

Агуулга

1 Оршил

Интернэт орчинд төөрөгдүүлсэн, сурталчилгааны зорилготой болон хуурамч мэдээлэл агуулсан spam сэтгэгдлүүд нь олон нийтийн мэдээллийн хэрэгслүүдийн аюулгүй байдлыг алдагдуулах томоохон асуудлын нэг юм. Тухайлбал, Facebook орчинд хийгддэг автомат сэтгэгдлийн спам нь хэрэглэгчийн туршлагыг муутгахын зэрэгцээ хуурамч зар сурталчилгааг түгээх, залилангийн эрсдэлийг нэмэгдүүлдэг. Тиймээс Facebook-ийн спам сэтгэгдлийг машин сургалтын аргаар илрүүлэх нь ач холбогдол өндөртэй бөгөөд энэ төслийн үндсэн зорилго болно. Энэхүү ажлын онолын үндэслэлүүдэд Decision Tree Learning [1] болон Naive Bayes Models [2] хамаарна.

2 Өгөгдөл цуглуулалт

Энэхүү төслийн зорилго нь Facebook дээрх хэрэглэгчдийн бичсэн сэтгэгдлүүдийг Spam болон Ham гэж автоматаар ангилах явдал юм. Өгөгдлийг Facebook-ийн олон нийтийн хуудас болон

нийтлэлүүдийн доорх сэтгэгдлүүдээс гараар цуглуулж, Google Sheets дээр нэгтгэн хадгалсан. Сэтгэгдэл бүр дараах шинжүүдтэй:

- label - Spam эсэх (spam / ham)
- Коммент бичсэн хүний нэр
- Raw comment - Тухайн бичсэн жинхэнэ текст
- Цэвэрлэсэн сэтгэгдэл - Текстийг боловсруулсны дараах хувилбар
- Постны агуулга
- Зураг агуулсан эсэх
- Нэрээ нууцалсан эсэх
- Монгол нэр эсэх
- Голдуу ашигласан үсэг (Latin / Cyrillic)
- Emoji-ний тоо
- Сэтгэгдлийн урт
- Email/Link/Утас агуулсан эсэх

Эдгээр шинжүүд нь:

- **Naive Bayes** — текстэн шинж дээр суурилсан модел
- **Decision Tree** — тоон ба логик шинжүүд дээр суурилсан модел

хоёуланг сургах боломж олгосон.

3 Өгөгдөл унших ба ачаалах процесс

Өгөгдлийг боловсруулахад зориулж **DataLoader** гэсэн класс бичсэн бөгөөд энэ нь:

- Google Sheets
- CSV файл
- Excel файл (.xlsx)

гэсэн гурван эх сурвалжаас ачаалах боломжтой.

Жишээ код:

```
df = pd.read_csv(url, engine='python', on_bad_lines='skip')
```

3.1 Ачаалах үеийн алхмууд

- Google Sheets линкээс CSV хэлбэрээр татаж уншсан
- Эхний мөрийг баганын нэр болгохоор шүүсэн
- SSL баталгаажуулалттай холбоотой алдааг түр хугацаанд алгассан
- **label** баганын утгуудыг: жижиг үсэг болгох, илүү хоосон зай арилгах, буруу өгөгдлүүдийг шүүх

3.1.1 Давуу тал

- Олон төрлийн эх үүсвэрээс нэг ижил логикоор өгөгдөл ачаална
- `on_bad_lines='skip'` - буруу мөр илэрвэл код зогсохгүй
- Онлайн Google Sheets өгөгдлийг шууд уншина

4 Өгөгдөл цэвэрлэх ба бэлтгэх

4.1 Naive Bayes — текстэн өгөгдлийн цэвэрлэгээ

`prepare_for_naive_bayes()` функц нь текстийг дараах 3 сонголтоор бэлтгэх боломжтой:

- **raw** — анхны бичсэн текст
- **transliterated** — кирилл рүү хөрвүүлж, цэвэрлэсэн текст
- **both** — raw + cleaned нэгтгэсэн текст

4.1.1 Цэвэрлэгээний үеийн алхмууд

- Хоосон label/text мөрүүдийг хассан
- Цэвэрлэсэн текст байхгүй үед **fallback** хийдэг
- Эцэст нь сургалтанд бэлэн **x, y** Series буцаана:
- x -> текст
- y -> label

```
return df_clf['text'], df_clf['label']
```

4.2 Decision Tree — тоон ба логик өгөгдөл бэлтгэх

Decision Tree модел нь текст бус дараах төрлийн өгөгдлийг ашигладаг:

- Boolean (yes/no, true/false)
- Тоон үзүүлэлтүүд
- Label (spam / ham)

```
df[col] = df[col].map({
    True: 1, False: 0,
    "yes": 1, "no": 0,
    "  ": 1, "  ": 0
}).fillna(0)
```

Boolean багануудыг 0/1 болгон хөрвүүлж, тоон багануудыг numeric болгож, label-ийг цэвэрлэсэн.

```
df[col] = pd.to_numeric(df[col], errors='coerce').fillna(0)
```

Label цэвэрлэгээ: - Spam -> spam - Ham -> ham

5 Өгөгдөл хуваах

Загварын гүйцэтгэлийг үнэлэхийн тулд өгөгдлийг: - **Train set — 80%**, - **Test set — 20%** хувиар хуваав. Хуваахдаа ангиллын тэнцвэрт байдлыг хадгалахын тулд **stratified split** ашигласан.

Naive Bayes-д зориулсан өгөгдөлд:

```
from sklearn.model_selection import train_test_split

X, y = data_loader.prepare_for_naive_bayes(text_source='raw')

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

Decision Tree-д зориулсан өгөгдөлд:

```
df_features, attributes, label_col = data_loader.prepare_for_decision_tree()

train_df, test_df = train_test_split(
    df_features, test_size=0.2, stratify=df_features[label_col], random_state=42
)
```

DataLoader нь ML pipeline-д хэрэгтэй өгөгдлийг бүрэн бэлтгэж өгдөг.

Naive Bayes : - text - abel - токенчлол хийхэд шууд бэлэн - lower-case - stopwords устгах боломжтой

Decision Tree : - Boolean шинжүүд - тоон үзүүлэлтүүд - label - attribute жагсаалт

Ингэснээр өгөгдөл Naive Bayes болон Decision Tree моделуудыг сургахад 100% бэлэн болсон.

6 Өгөгдлийн тархалтын дүрслэл

Тархалтын графикууд нь spam болон normal сэтгэгдлүүдийн үгийн давтамж, урт, хамгийн түгээмэл үгсийн ялгааг ажиглахад тусалсан. Үүнд: - Сэтгэгдлийн урт
- Emoji-ийн тоо
- Голдуу ашигласан үсгийн төрөл (Latin / Cyrillic)
- Spam ангилалд хамгийн түгээмэл үгс
- Ham ангилалд хамгийн түгээмэл үгс багтана

Дүрслэлүүдээс харахад spam сэтгэгдэлд: - Spam сэтгэгдэлд мөнгө, зээл, чат гэх мэт үгс давамгайлсан. - Ham сэтгэгдэлд илүү энгийн, сэдвийн дагуу текст давамгайлдаг. - Emoji болон бусад тусгай тэмдэгтүүд spam-д илүү их агуулагдсан. - Энэ дүрслэл нь Naive Bayes болон Decision Tree загваруудын оновчлолд тус болно.

7 Машин сургалтын загварууд

Энэхүү төсөлд хоёр загвар хэрэглэв. Текстэн өгөгдлийн онцлогоос хамааран Naive Bayes болон Decision Tree алгоритмууд тохиромжтой гэж үзсэн.

7.1 Naive Bayes (Unigram & Bigram) модель

Энэхүү төслийн spam/ham ангиллын загвар нь Multinomial Naive Bayes зарчмыг ашигладаг.

N-gram токенчлол :

```
from naive_bayes_model import ngram_tokenize

text = "hello world nice day"
ngrams = ngram_tokenize(text, ngram_range=(1,2))
print(ngrams)
```

- Unigram: нэг үг бүр ([“hello”,“world”,“nice”,“day”])
- Bigram: хоёр үгийн дараалал ([“hello world”,“world nice”,“nice day”])
- ngram_range=(1,2) - Unigram + Bigram токенчлолыг нэгэн зэрэг гаргана.

Энэ нь текстийн контекстийг сайжруулахад ашиглагдсан.

MyMultinomialNB класс :

- fit(X, y) → сургалт
- predict(X) → шинэ текстийг ангилах

- predict_single(text) → нэг текстийн ангилал
- save_vocabulary(filename) → vocabulary болон class тус бүрийн top n-grams-г файлд хадгалах

7.1.1 Сургалтын алхам

```
from naive_bayes_model import MyMultinomialNB
from sklearn.model_selection import train_test_split

# 1.
X, y = data_loader.prepare_for_naive_bayes(text_source='both')

# 2.     /
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# 3.          (Unigram+Bigram)
nb_model = MyMultinomialNB(alpha=1.0, ngram_range=(1,2))
nb_model.fit(X_train, y_train)

# 4.
y_pred = nb_model.predict(X_test)

# 5. Vocabulary-
nb_model.save_vocabulary("vocabulary.txt")
```

7.2 Decision Tree (Spam Feature Classifier)

Decision Tree нь текст бус логик болон тоон шинжүүд дээр суурилсан ангиллын модель юм.

Энгийн алгоритм (ID3) : - Энтропи (Entropy) тооцоолох - Information Gain тооцоолох - Багана тус бүрээр хуваалт хийх - Рекурсив сурах

MyDecisionTree wrapper :

- fit(df, attributes, target) → Decision Tree сургах
- predict(df) → бүх dataset-д таамаглал хийх
- predict_single(sample_dict) → нэг дээжинд таамаглал
- print_tree() → модыг хүснэгт маягаар хэвлэх

7.2.1 Сургалтын алхам

```
from decision_tree_model import MyDecisionTree

# 1.
```

```
df, attributes, target = data_loader.prepare_for_decision_tree()

# 2.
tree_model = MyDecisionTree(max_depth=8)
tree_model.fit(df, attributes, target)

# 3.
tree_model.print_tree()

# 4.
y_pred = tree_model.predict(df)

# 5.
sample = df.iloc[0][attributes].to_dict()
print("Sample prediction:", tree_model.predict_single(sample))
```

7.3 Моделийг сургах болон үнэлэх

ModelEvaluator нь дараах зорилготой: - Моделийг сургах (train_naive_bayes, train_decision_tree) - Үнэлгээ хийх (evaluate) - Нэг сэтгэгдэлд таамаглал гаргах (predict_comment)

7.3.1 Naive Bayes сургалт

Train-test хуваалт

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)
```

- 70% сургалт, 30% шалгалт
- random_state-ээр дахин давтагдах үр дүн баталгаажна

Модель үүсгэх

```
from naive_bayes_model import MyMultinomialNB
model = MyMultinomialNB(alpha=1.0, ngram_range=(1,2))
model.fit(X_train, y_train)
```

- alpha → Laplace smoothing
- ngram_range → unigram/bigram хослуулсан токенчлол

Сургалтын үр дүн

- self.X_train, self.X_test, self.y_train, self.y_test хадгалагдана
- Модель бэлэн болж, self.model_type = ‘naive_bayes’ болно

7.3.2 Decision Tree сургалт

Train-test хуваалт

```
train_df, test_df = train_test_split(
    df, test_size=0.2, random_state=42, stratify=df[target]
)
```

- 80% сургалт, 20% шалгалт
- stratify=df[target] → label-ийн тархалтыг хадгалах

Модель үүсгэх

```
from decision_tree_model import MyDecisionTree
tree_model = MyDecisionTree(max_depth=8)
tree_model.fit(train_df, attributes, target)
```

- max_depth → модны гүн
- Сургалтын өгөгдлөөс boolean ба numeric шинжүүдийг ашиглана

7.3.3 Үнэлгээ хийх

Таамаглал гаргах

```
if model_type == 'naive_bayes':
    y_pred = model.predict(X_test)
elif model_type == 'decision_tree':
    y_pred = model.predict(X_test)
```

Accuracy :

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
```

Confusion Matrix :

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred, labels=sorted(set(y_test)))
```

Precision, Recall, F1-Score

Class бүрд тооцоолно:

Precision = TP / (TP + FP) Recall = TP / (TP + FN) F1-Score = 2 * (Precision*Recall)/(Precision+Recall)

Нэг сэтгэгдэлд таамаглал гаргах :

Naive Bayes:

```
pred = model.predict_single(comment)
```

Decision Tree:

```
pred = model.predict_single(features_dict)
```

- ModelEvaluator нь Naive Bayes (unigram/bigram) болон Decision Tree загваруудыг сургалт, үнэлгээ, prediction-д нэгтгэн ашиглахад тохиромжтой.
- Train-test хуваалт, accuracy, confusion matrix, precision/recall/F1 зэрэг бүх үндсэн үнэлгээний метрик автомат гарна.
- Эдгээр алхмуудыг ашигласнаар моделийг хялбар, дахин давтагдах байдлаар сургаж, үнэлж, сэтгэгдлийг ангилах боломжтой.

8 Өгөгдлийг визуализаци хийх

Visualizer нь: - Өгөгдлийг бэлдэх (_prepare_data) - Текст, emoji, binary болон numeric шинжүүдийг дүрслэх - Spam/Ham ангилалд хэрхэн тархсаныг харах

Дүрслэлүүд: - Сэтгэгдлийн урт (Comment length) тархалт - Emoji тоо - Үсгийн төрөл (Script types) - Binary features тархалт - Feature=1 үед Spam/Ham тархалт - Correlation matrix

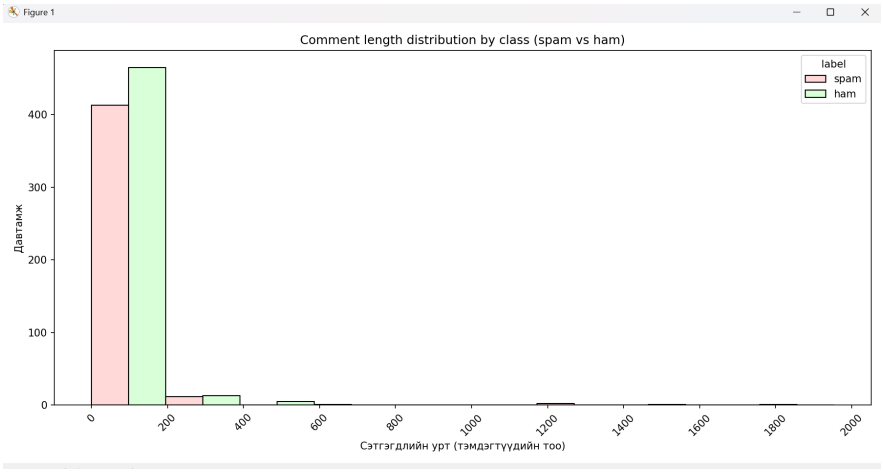
8.1 Сэтгэгдлийн урт (Comment length) тархалт

```
viz.plot_comment_length()
```

- Histogram дүрслэл: Spam vs Ham
- bins=10, multiple='dodge'
- X-тэмдэгтийн урт, Y-д давтамж

Үзүүлэлт: Spam сэтгэгдэл ихэвчлэн богино, Ham сэтгэгдэл харьцангуй урт

Figure 1: Сэтгэгдлийн уртын тархалт

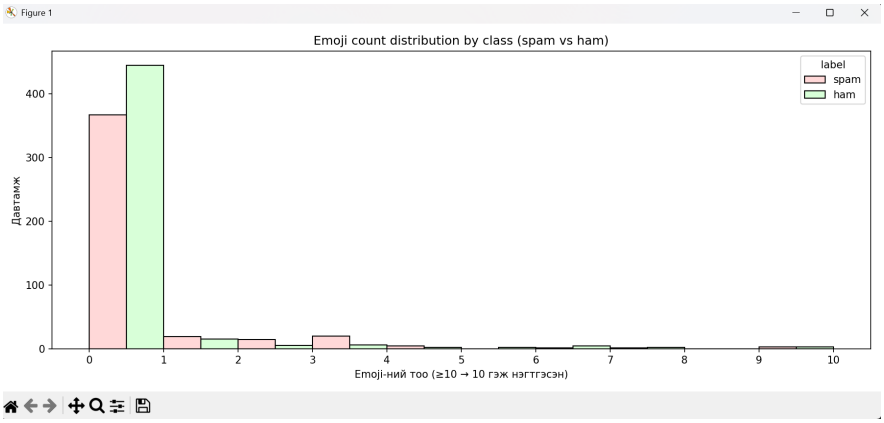


8.2 Emoji тоо

```
viz.plot_emoji_count()
```

- Emoji тоо ≤10 гэж нэгтгэж histogram үүсгэнэ
- Spam болон Ham-д хэрхэн тархсан харагдана

Figure 2: Emoji

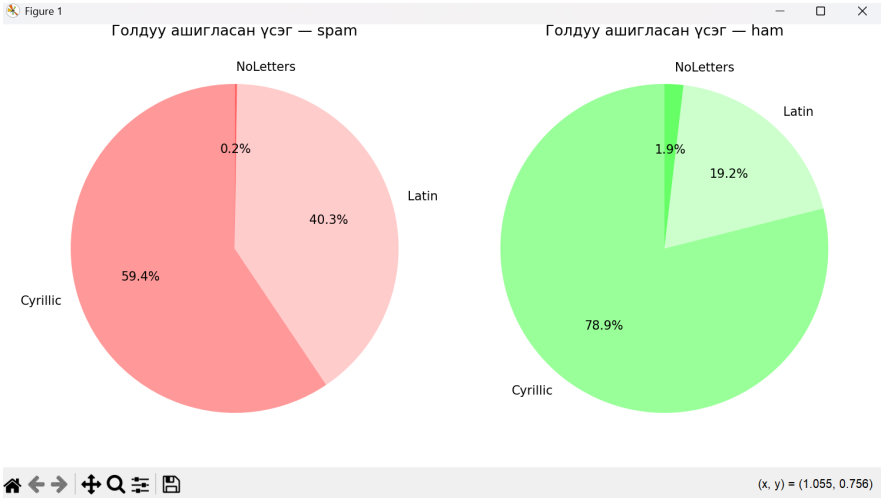


8.3 Үсгийн төрөл (Script types)

```
viz.plot_script_types()
```

- Pie chart: голдуу ашигласан үсгийн төрөл
- Spam/Ham тусад нь харуулна
- Script төрөл ба хувьсгалтай тархалт харагдана

Figure 3: Script types



8.4 Binary features тархалт

```
viz.plot_binary_distribution()
```

- Binary багануудын хувь (%) тархалт
- 0/1 утгуудын нийт хувь, count
- Spam/Ham аль нь илүү давамгай байгааг харах боломжтой

8.5 Feature=1 үед Spam/Ham тархалт

```
viz.plot_spam_ham_by_feature()
```

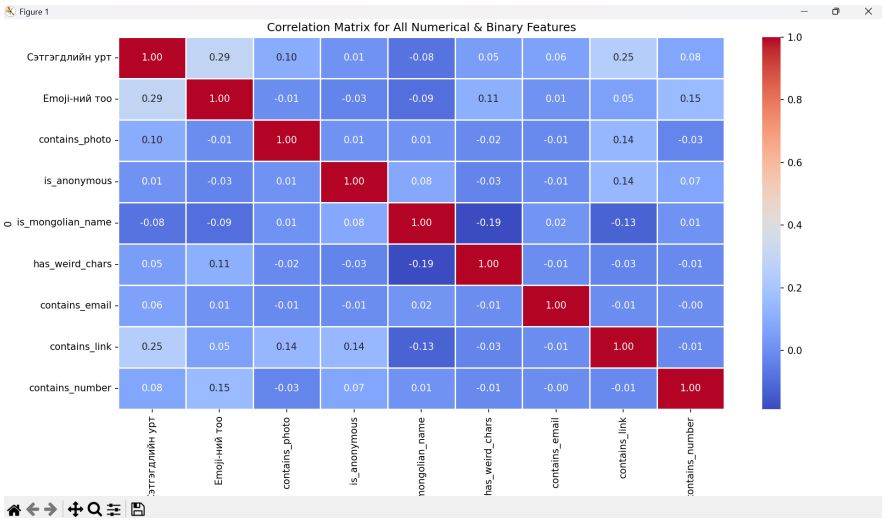
- Binary feature=1 үед Spam vs Ham хувь
- Жишээ: contains_link=1 → spam илүү давтамжтай

8.6 Correlation matrix

```
viz.plot_correlation_matrix()
```

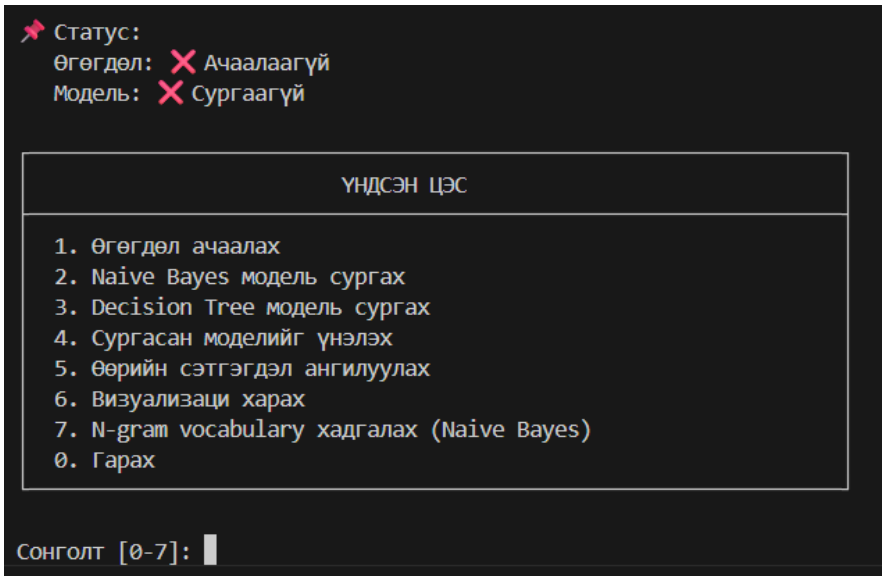
- Тоон болон binary шинжүүдийн хамаарлыг heatmap-аар дүрслэнэ
- Feature хоорондын хамаарлыг 0–1 хүрээнд харах боломжтой

Figure 4: Correlation Matrix



Visualizer нь өгөгдлийг цэвэрлэх, хувиргах, дүрслэх бүх алхмыг нэгтгэсэн бөгөөд Spam/Ham ангиллын тархалт, feature хувьсгал, correlation-ийг харах боломжтой. Үүнийг ашигласнаар текст ба binary feature анализ хийх, тархалтын судалгаа гаргах үйл явцыг хялбар болгоно.

Figure 5: Console App



9 Консол application — SpamClassifierApp

SpamClassifierApp нь хэрэглэгчийн оролтод тулгуурлан дараах үйлдлүүдийг гүйцэтгэнэ:

9.1 Өгөгдөл ачаалах

- Google Sheets, CSV, XLSX файлаас өгөгдөл уншина.
- DataLoader ашиглан өгөгдлийг ачаална.
- Visualizer-д өгөгдлийг дамжуулж визуализацийг бэлдэнэ.
- Label тархалт, нийт мөрийг дэлгэцэнд харуулна.

Жишээ үр дүн:

Нийт мөр: 1200 Label тархалт: ham 900 spam 300

9.2 Naive Bayes модель сургах

- Text эх үүсвэрийг сонгоно: raw, transliterated, эсвэл хоёулаа.
- Alpha (Laplace smoothing) болон n-gram төрлийг тохируулна.
- Test set-ийн хувь (default 30%) зааж өгнө.
- `ModelEvaluator.train_naive_bayes` дуудагдаж, модель сургана.
- Сургалтын болон тест set-ийн мөрийн тоо, vocabulary хэмжээ хэвлэгдэнэ.
- Үүссэн n-gram vocabulary-г файлд хадгалах боломжтой.

Жишээ үр дүн:

Train set: 840 мөр Test set: 360 мөр Vocabulary хэмжээ: 2450

9.3 Decision Tree модель сургах

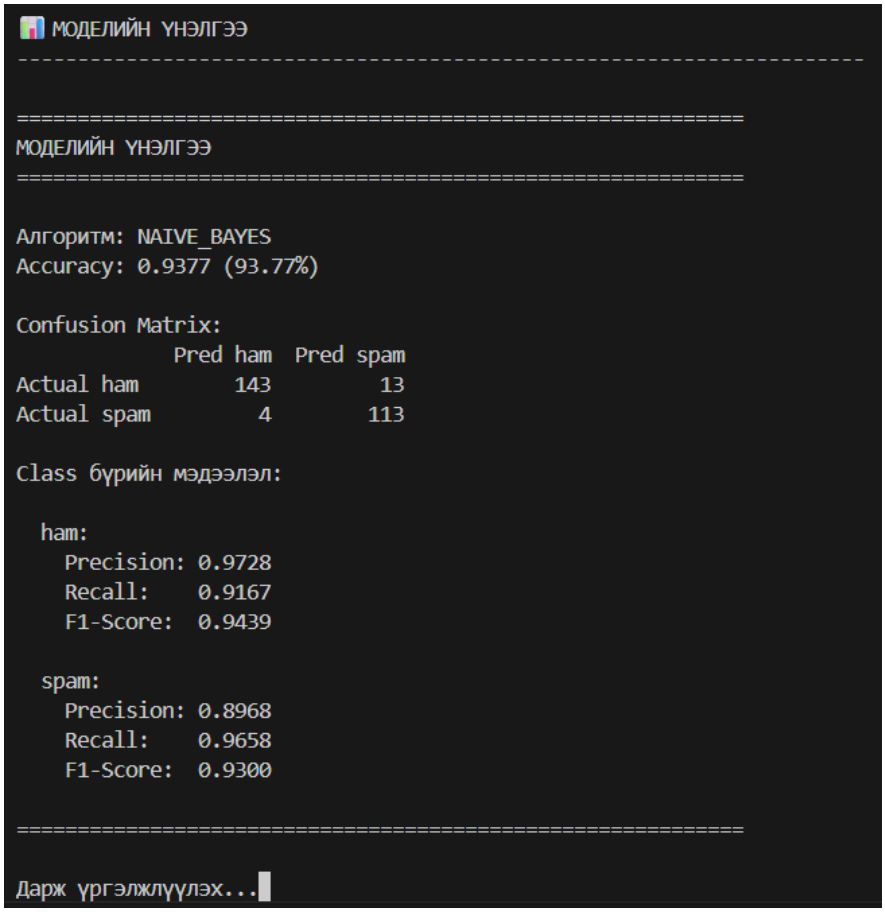
- Maximum depth болон test set-ийн хувийг тохируулна.
- `ModelEvaluator.train_decision_tree` дуудагдаж, модель сургагдана.
- Сургалтын болон тест set-ийн мөрийн тоо хэвлэгдэнэ.
- Хэрэглэгч хүсвэл tree бүтцийг console дээр харах боломжтой.

9.4 Модел үнэлэх

- `ModelEvaluator.evaluate()` дуудаж, сургасан моделийг үнэлнэ.
- Үүнд:
  - Accuracy
  - Confusion Matrix
  - Class-wise Precision, Recall, F1-Score

Жишээ тайлан:

Figure 6: Report



9.5 Өөрийн сэтгэгдэл ангилуулах

- Naive Bayes: Текст оруулахад spam/ham-г таамаглана.

- Decision Tree: Шинжүүдийг оруулахад spam/ham-г таамаглана.
- Үр дүнг тодорхой icon-оор (SPAM, HAM) харуулна.

9.6 Визуализаци харах

- Сэтгэгдлийн урт, emoji count, script types, binary feature тархалт, correlation matrix-г харуулна.
- Хэрэглэгч хүссэн графикаа сонгож эсвэл бүх график-г нэг дор үзэж болно.

9.7 N-gram vocabulary хадгалах

- Naive Bayes сурагдсан vocabulary-г текст файлд хадгална.
- Vocabulary хэмжээ болон файлын нэрийг дэлгэцэнд харуулна.

9.8 Гүйцэтгэсэн процесс

1. Өгөгдөл ачаална → DataFrame ба Visualizer бэлдэнэ
2. Модел сургана → Naive Bayes эсвэл Decision Tree
3. Модел үнэлнэ → Accuracy, Confusion Matrix, Precision/Recall/F1
4. Өөрийн сэтгэгдэл ангилна → Single prediction
5. Визуализаци хийж анализлана → Histogram, Pie chart, Heatmap
6. Vocabulary-г файлд хадгална (Naive Bayes)

10 Дүгнэлт

Энэхүү Spam сэтгэгдэл илрүүлэгч жижиг систем нь одоогийн байдлаар зөвхөн зээлтэй холбоотой спам сэтгэгдлийг өндөр нарийвчлалтайгаар илрүүлэх боломжтой . Энэ нь хэрэглэгчдэд өдөр тутмын пост, комментуудаас зээлийн шинжтэй сэжигтэй мэдээллийг хурдан таньж, аюулгүй байдлаа хангахад үр дүнтэй шийдэл болж байна. Гэсэн хэдий ч бүх төрлийн спам (сурталчилгаа, залилан, фишинг, залилангийн холбоосууд гэх мэт)–ыг өндөр нарийвчлалтай илрүүлэхийн тулд илүү олон төрөл, олон эх сурвалжаас бүрдсэн том хэмжээний сургалтын өгөгдлийн сан шаардлагатай. Иймээс ирээдүйд системийг дараах чиглэлээр өргөжүүлэх боломжтой: Спамын төрөл бүрийн өгөгдлийг өргөн хүрээтэй цуглуулах Мэдээлэл боловсруулалтын нарийвчилсан аргачлал сайжруулах Илүү хөгжингүй ML/DL загвар ашиглан бүх төрлийн спамыг илрүүлэх чадварыг нэмэгдүүлэх Эцэст нь, энэхүү төсөл нь анхан шатны түвшинд бодит хэрэглээнд ашиглагдах боломжтой, цаашид хөгжүүлбэл олон төрлийн spam-ийг илрүүлдэг бүрэн хэмжээний ухаалаг систем болох боломжтой гэж дүгнэж байна.

11 Багийн гишүүдийн оролцоо

11.1 Хийх ажлын жагсаалт:

- Өгөгдлөө цуглуулах (1 хүний 125 ham, 125 spam) : Бүгд
- Өгөгдлөө унших, цэвэрлэх, хуваах (CSV, excel format) : Нямбаяр
- Тархалтуудыг дүрслэх : Эрдэнэ-Очир, Нямбаяр
- PPT бэлдэх : Ариунзаяа
- Тайлан бичих (Quarto, qmd file) : Буян-Эрдэнэ

11.2 Хийсэн моделууд:

- Naive Bayes classifier (Unigram) - Буян-Эрдэнэ
- Naive Bayes classifier (Bigram) - Эрдэнэ-Очир
- Decision tree classifier - Ариунзаяа
- Unified model - Нямбаяр

Ашигласан материал

[1] S. Russell and P. Norvig, “Decision tree learning,” in *Artificial intelligence: A modern approach*, 3rd ed., Prentice Hall, 2010, pp. 653–662.

[2] S. Russell and P. Norvig, “Naive bayes models,” in *Artificial intelligence: A modern approach*, 3rd ed., Prentice Hall, 2010, pp. 758–760.