Бернуллийн Наив Байесын загварын хэрэглээ: Хэрэглэгчийн худалдан авах төлөвийг урьдчилан таамаглах нь

Б.Билгүүнтөгөлдөр (20B1NUM1087) О.Энхбаяр (21B1NUM0940)
Э.Алтаншагай (22B1NUM0331) Б.Намуундарь (23B1NUM0543)
Ч.Саранцацралт (24B1NUM0019)

2025 оны 12-р сарын 5

Энэхүү төслийн ажлаар Kaggle платформоос авсан Groceries өгөгдлийн санг ашиглан хэрэглэгчдийн худалдан авалтын төлөвийг урьдчилан таамаглах Бернуллийн Наив Байес (Bernoulli Naive Bayes) ангиллын загварыг математик үндэслэлтэйгээр бүтээв. Судалгаанд 9,835 худалдан авалтын түүх, 169 өвөрмөц бүтээгдэхүүн агуулсан өгөгдлийг боловсруулж, “whole milk” (сүү) худалдан авах эсэхийг таамаглах загвар бүтээв. Загварыг 5-фолд кросс-валидациар үнэлэхэд дундаж нарийвчлал 74.2% байв.

Агуулга

Шаардлагатай багцууд

Шаардлагатай багцуудыг дараах байдлаар урьдчилан суулгана.

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

1 Удиртгал

1.1 Судалгааны хэрэгцээ, шаардлага

Орчин үеийн жижиглэн худалдааны салбарт хэрэглэгчдийн худалдан авах зан төлөвийг урьдчилан таамаглах нь маркетингийн стратеги, бараа материалын менежмент, хувийн санал болгох систем зэрэгт чухал ач холбогдолтой Leskovec et al. (2014). Машин сургалтын аргуудыг ашиглан хэрэглэгчдийн өмнөх худалдан авалтын түүх дээр үндэслэн ирээдүйн худалдан авалтыг таамаглах боломжтой.

Байесын зарчимд суурилсан ангиллын алгоритмууд нь статистикийн хүчтэй үндэслэлтэй, тайлбарлагдах боломжтой, бага өгөгдөлтэй ч сайн ажилладаг зэрэг давуу талтай Murphy (2012). Бернуллийн Наив Байес нь ялангуяа хоёртын (binary) шинж чанаруудтай өгөгдөлд тохиромжтой бөгөөд текст ангилал, худалдан авалтын дүн шинжилгээнд өргөн хэрэглэгддэг.

1.2 Зорилго

Энэхүү төслийн зорилго нь:

1. Бернуллийн Наив Байесын алгоритмыг математик үндэслэлтэйгээр ойлгох, батлах
2. Тус алгоритмыг Python хэлээр эхнээс нь бүтээх
3. Groceries өгөгдлийн сан дээр загварыг сургаж, үнэлгээ хийх
4. Хэрэглэгчийн худалдан авах магадлалыг нөхцөлт магадлалын аргаар тооцоолох
5. Sklearn сангийн загвартай харьцуулж, баталгаажуулах

1.3 Өгөгдлийн эх сурвалж

Судалгаанд Kaggle платформ дээрх “Groceries” өгөгдлийн санг ашиглав Hahsler et al. (2006). Тус өгөгдөл нь хүнсний дэлгүүрийн 9,835 худалдан авалтын түүхийг агуулсан бөгөөд худалдан авсан бүтээгдэхүүнүүдийн жагсаалт байна. Өгөгдлийг https://www.kaggle.com/datasets/irfanasrullah/groceries хаягаас татаж авав.

2 Онолын үндэслэл

2.1 Байесын теорем

Тодорхойлолт 2.1 (Нөхцөлт магадлал): A ба B үзэгдлүүдийн хувьд B өгөгдсөн үеийн A-ийн нөхцөлт магадлалыг дараах байдлаар тодорхойлно Murphy (2012):

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0 \tag{1}$$

Теорем 2.1 (Байесын теорем): A ба B үзэгдлүүдийн хувьд дараах тэнцэтгэл биелнэ:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{2}$$

Баталгаа: Нөхцөлт магадлалын тодорхойлолтоос:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \text{ба} \quad P(B|A) = \frac{P(A \cap B)}{P(A)}$$

Хоёр дахь тэнцэтгэлээс $P(A \cap B) = P(B|A) \cdot P(A)$ гэж бичвэл:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad \blacksquare$$

Байесын теоремын бүрэлдэхүүн хэсгүүд:

- $P(A|B)$ — **Постериор магадлал** (posterior probability): B нотолгоо өгөгдсөн үед A-ийн магадлал
- $P(B|A)$ — **Үнэний хувь** (likelihood): A таамаглал үнэн үед B нотолгоо ажиглагдах магадлал
- $P(A)$ — **Приор магадлал** (prior probability): Нотолгооноос өмнөх A-ийн магадлал
- $P(B)$ — **Нотолгооны магадлал** (evidence): Нормчлолын тогтмол

2.2 Наив Байесын нөхцөлт үл хамаарлын таамаглал

Асуудал: $P(\mathbf{X}|C) = P(X_1, X_2, ..., X_n|C)$ хамтын нөхцөлт тархалтыг шууд үнэлэх бэрхшээлтэй. Хэрэв $X_i \in \{0, 1\}$ бол 2^n параметр үнэлэх шаардлагатай.

Тодорхойлолт 2.2 (Нөхцөлт үл хамаарал): $X_1, ..., X_n$ санамсаргүй хувьсагчид C өгөгдсөн үед нөхцөлт хамааралгүй гэж нэрлэгдэнэ, хэрэв:

$$P(X_1, ..., X_n|C) = \prod_{i=1}^{n} P(X_i|C) \tag{3}$$

Тэмдэглэгээ: $X_i \perp X_j|C, \quad \forall i \neq j$

Математик утга: Энэ таамаглал нь n шинж чанарын хамтын тархалтыг n ширхэг нэг хэмжээст тархалт руу задлах боломжийг олгодог. Үүнээр үнэлэх параметрүүдийн тоог $O(2^n)$-ээс $O(n)$ болгон бууруулдаг.

Лемм 2.1: Наив таамаглал ёсоор магадлалын гинжин дүрэм хялбарчлагдана:

$$P(X_1, ..., X_n|C) = P(X_1|C) \cdot P(X_2|X_1, C) \cdot ... \cdot P(X_n|X_1, ..., X_{n-1}, C)$$

Наив таамаглал: $P(X_i|X_j, C) = P(X_i|C)$ тул:

$$P(X_1, ..., X_n|C) = \prod_{i=1}^{n} P(X_i|C) \quad \blacksquare$$

2.3 Бернуллийн Наив Байес загвар

Тодорхойлолт 2.3 (Бернуллийн тархалт): $X \in \{0, 1\}$ санамсаргүй хувьсагч нь θ параметртэй Бернуллийн тархалттай гэж нэрлэгдэнэ, хэрэв:

$$P(X = x) = \theta^x (1 - \theta)^{1-x}, \quad x \in \{0, 1\} \tag{4}$$

Үүнд $\theta = P(X = 1)$ нь амжилтын магадлал.

Шалгалт:

- $x = 1$ үед: $P(X = 1) = \theta^1 (1 - \theta)^0 = \theta$ □
- $x = 0$ үед: $P(X = 0) = \theta^0 (1 - \theta)^1 = 1 - \theta$ □
- Нийлбэр: $\theta + (1 - \theta) = 1$ □

Теорем 2.2 (Бернуллийн Наив Байес ангилагч): $\mathbf{X} = (X_1, ..., X_n) \in \{0, 1\}^n$ хоёртын шинж чанарууд, $C \in \{0, 1\}$ хоёртын анги өгөгдсөн үед Бернуллийн Наив Байес ангилагчийн таамаглал:

$$\hat{C} = \arg\max_{c \in \{0,1\}} \left[P(C = c) \prod_{i=1}^{n} P(X_i = 1|C = c)^{X_i} \cdot (1 - P(X_i = 1|C = c))^{1-X_i} \right] \tag{5}$$

Баталгаа: Байесын теорем + Наив таамаглал + Бернуллийн тархалт:

$$P(C|\mathbf{X}) \propto P(C) \prod_{i=1}^{n} P(X_i|C)$$

$X_i|C \sim \text{Bernoulli}(\theta_{ic})$ үед:

$$P(X_i|C) = \theta_{ic}^{X_i}(1-\theta_{ic})^{1-X_i}$$

Орлуулбал:

$$P(C|\mathbf{X}) \propto P(C) \prod_{i=1}^{n} \theta_{ic}^{X_i}(1-\theta_{ic})^{1-X_i} \quad \blacksquare$$

2.4 Лапласын тэгшитгэл

Асуудал: Maximum Likelihood үнэлгээ (MLE):

$$\hat{\theta}_{ic}^{MLE} = \frac{N_{ic}}{N_c} \tag{6}$$

Үүнд N_{ic} нь $C = c$ ангид $X_i = 1$ байсан тоо, N_c нь $C = c$ ангийн нийт тоо.

Хэрэв $N_{ic} = 0$ бол $\hat{\theta}_{ic} = 0$ болж, бүтээгдэхүүний магадлал тэг болно. Энэ нь бусад бүх шинж чанаруудын мэдээллийг устгана (тэг үржүүлбэл тэг).

Шийдэл (Байесын үнэлгээ): Beta prior тархалт ашиглана:

$$\theta_{ic} \sim \text{Beta}(\alpha, \alpha)$$

Posterior дундаж (MAP үнэлгээ):

$$\hat{\theta}_{ic}^{MAP} = \frac{N_{ic} + \alpha}{N_c + 2\alpha} \tag{7}$$

Тайлбар:

- α — smoothing параметр (pseudo-count)
- Тоологч: $N_{ic} + \alpha$ (бодит + хуурамч $X_i = 1$ ажиглалт)
- Хуваагч: $N_c + 2\alpha$ (нийт + хуурамч $X_i = 1$ + хуурамч $X_i = 0$)

2.5 Лог магадлалын хувиргалт

Асуудал (Numerical underflow): n шинж чанартай үед магадлалуудын үржвэр маш бага утга авч, компьютерт 0 болж хувирна.

Шийдэл: Лог хувиргалт. log функц нь strictly monotone increasing тул:

$$\arg\max_c f(c) = \arg\max_c \log f(c)$$

Теорем 2.3 (Лог магадлалын томьёо):

$$\log P(C|\mathbf{X}) \propto \log P(C) + \sum_{i=1}^{n} [X_i \log \theta_{ic} + (1 - X_i) \log(1 - \theta_{ic})] \tag{8}$$

3 Өгөгдлийн боловсруулалт

3.1 Шаардлагатай сангуудыг ачаалах

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
import warnings
warnings.filterwarnings('ignore')

# График тохиргоо
plt.rcParams['figure.figsize'] = (6, 4)
plt.rcParams['axes.unicode_minus'] = False

# Санамсаргүй байдлын seed тогтоох (reproducibility)
np.random.seed(42)

print("Сангуудыг амжилттай ачааллаа!")
print(f"NumPy хувилбар: {np.__version__}")
print(f"Pandas хувилбар: {pd.__version__}")
```

```
Сангуудыг амжилттай ачааллаа!
NumPy хувилбар: 1.26.4
Pandas хувилбар: 2.1.4
```

3.2 Өгөгдлийг ачаалах

```
transactions = []

# CSV файлыг ачаалах
with open('groceries.csv', 'r', encoding='utf-8') as f:
    lines = f.readlines()

# Эхний мөр нь header - устгана
```

```
lines = lines[1:]

for line in lines:
    parts = [x.strip() for x in line.strip().split(',') if x.strip()]

    if len(parts) <= 1:
        continue

    cleaned_items = parts[1:]  # Эхний багана алгасах
    if cleaned_items:
        transactions.append(cleaned_items)

print(f"Худалдан авалтын тоо: {len(transactions)}")
print("\nЭхний 5 худалдан авалт:")
for i, t in enumerate(transactions[:5], 1):
    print(f"{i}. {t}")
```

```
Худалдан авалтын тоо: 9835

Эхний 5 худалдан авалт:
1. ['citrus fruit', 'semi-finished bread', 'margarine', 'ready soups']
2. ['tropical fruit', 'yogurt', 'coffee']
3. ['whole milk']
4. ['pip fruit', 'yogurt', 'cream cheese', 'meat spreads']
5. ['other vegetables', 'whole milk', 'condensed milk', 'long life bakery product']
```

3.3 Өгөгдлийн анхны шинжилгээ

```
# Бүх бүтээгдэхүүнийг цуглуулах
all_items = [item for trans in transactions for item in trans]
unique_items = sorted(set(all_items))
item_counts = Counter(all_items)

print(f"Өвөрмөц бүтээгдэхүүний тоо: {len(unique_items)}")
print(f"Нийт худалдан авалтын тоо: {len(all_items):,}")

# Худалдан авалтын хэмжээний статистик
trans_sizes = [len(trans) for trans in transactions]
print(f"\nХудалдан авалтын хэмжээ:")
print(f"  - Дундаж: {np.mean(trans_sizes):.2f} бүтээгдэхүүн")
print(f"  - Медиан: {np.median(trans_sizes):.0f} бүтээгдэхүүн")
print(f"  - Хамгийн бага: {min(trans_sizes)} бүтээгдэхүүн")
print(f"  - Хамгийн их: {max(trans_sizes)} бүтээгдэхүүн")

# Хамгийн түгээмэл 10 бүтээгдэхүүн
print(f"\nХамгийн түгээмэл 10 бүтээгдэхүүн:")
for item, count in item_counts.most_common(10):
    pct = count / len(transactions) * 100
    print(f"  - {item}: {count:,} ({pct:.1f}%)")
```

```
Өвөрмөц бүтээгдэхүүний тоо: 169
Нийт худалдан авалтын тоо: 43,367

Худалдан авалтын хэмжээ:
  - Дундаж: 4.41 бүтээгдэхүүн
```

```
  - Медиан: 3 бүтээгдэхүүн
  - Хамгийн бага: 1 бүтээгдэхүүн
  - Хамгийн их: 32 бүтээгдэхүүн

Хамгийн түгээмэл 10 бүтээгдэхүүн:
  - whole milk: 2,513 (25.6%)
  - other vegetables: 1,903 (19.3%)
  - rolls/buns: 1,809 (18.4%)
  - soda: 1,715 (17.4%)
  - yogurt: 1,372 (14.0%)
  - bottled water: 1,087 (11.1%)
  - root vegetables: 1,072 (10.9%)
  - tropical fruit: 1,032 (10.5%)
  - shopping bags: 969 (9.9%)
  - sausage: 924 (9.4%)
```

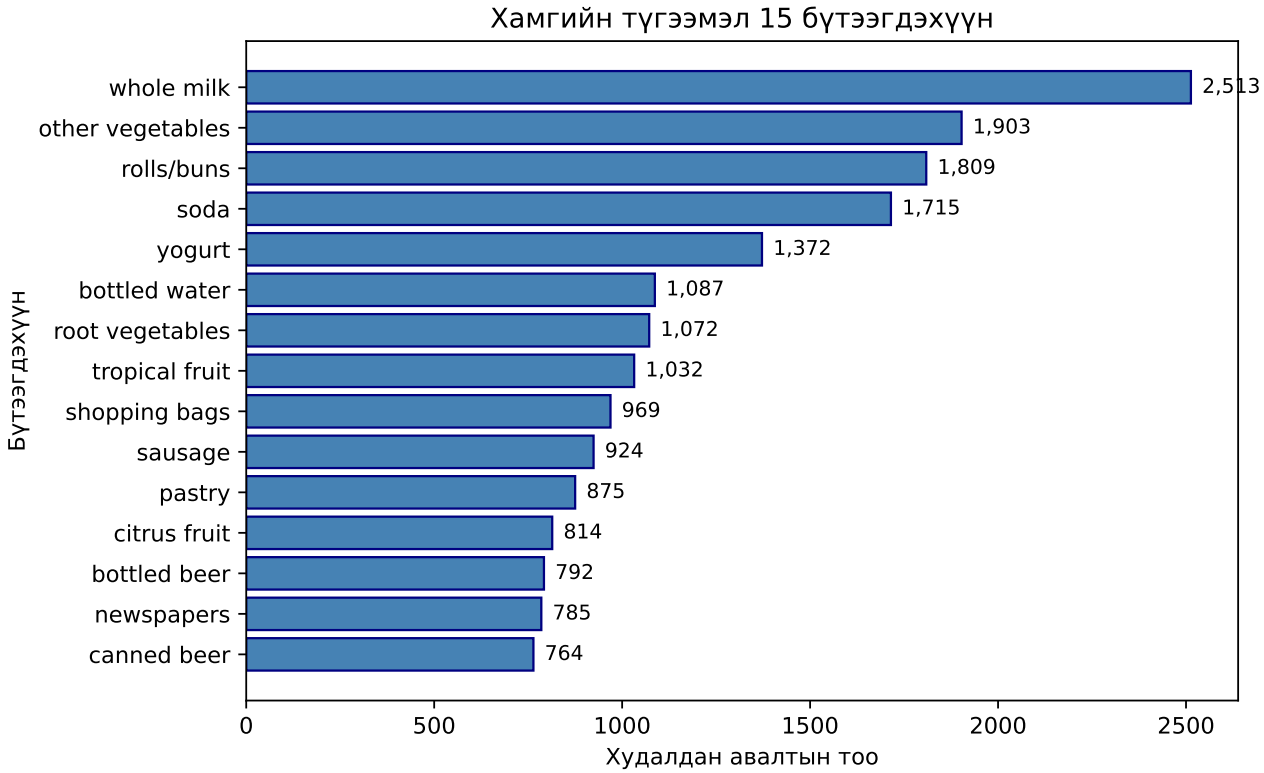
```python
top_15 = item_counts.most_common(15)
items = [x[0] for x in top_15]
counts = [x[1] for x in top_15]

plt.figure(figsize=(8, 5))
bars = plt.barh(items[::-1], counts[::-1], color='steelblue', edgecolor='navy')
plt.xlabel('Худалдан авалтын тоо')
plt.ylabel('Бүтээгдэхүүн')
plt.title('Хамгийн түгээмэл 15 бүтээгдэхүүн')

for bar, count in zip(bars, counts[::-1]):
    plt.text(bar.get_width() + 30, bar.get_y() + bar.get_height()/2,
             f'{count:,}', va='center', fontsize=9)

plt.tight_layout()
plt.show()
```

3.4 Хоёртын матриц үүсгэх

Машин сургалтын загварт оруулахын тулд гүйлгээний өгөгдлийг хоёртын шинж чанарын матриц болгон хувиргана.

```python
# Зорилтот бүтээгдэхүүн
TARGET_ITEM = 'whole milk'

# Бүтээгдэхүүний индекс үүсгэх
feature_items = [item for item in unique_items if item != TARGET_ITEM]
item_to_idx = {item: idx for idx, item in enumerate(feature_items)}
n_features = len(feature_items)

print(f"Шинж чанаруудын тоо: {n_features}")

# Хоёртын матриц үүсгэх
X = np.zeros((len(transactions), n_features), dtype=np.int8)
y = np.zeros(len(transactions), dtype=np.int8)

for i, trans in enumerate(transactions):
    if TARGET_ITEM in trans:
        y[i] = 1
    for item in trans:
```

Зураг 1: Хамгийн түгээмэл 15 бүтээгдэхүүн




```
        if item != TARGET_ITEM and item in item_to_idx:
            X[i, item_to_idx[item]] = 1

print(f"Шинж чанарын матрицын хэмжээ: {X.shape}")
print(f"Зорилтот вектор хэмжээ: {y.shape}")
print(f"\nЗорилтот хувьсагчийн тархалт:")
print(f"  - '{TARGET_ITEM}' АВААГҮЙ (y=0): {np.sum(y==0):,} ({np.mean(y==0)*100:.1f}%)")
print(f"  - '{TARGET_ITEM}' АВСАН (y=1): {np.sum(y==1):,} ({np.mean(y==1)*100:.1f}%)")
```

```
Шинж чанаруудын тоо: 168
Шинж чанарын матрицын хэмжээ: (9835, 168)
Зорилтот вектор хэмжээ: (9835,)

Зорилтот хувьсагчийн тархалт:
  - 'whole milk' АВААГҮЙ (y=0): 7,322 (74.4%)
  - 'whole milk' АВСАН (y=1): 2,513 (25.6%)
```

3.5 Өгөгдлийг хуваах

```
def train_test_split(X, y, test_size=0.2, random_state=42):
    """Өгөгдлийг сургалтын ба тестийн хэсэгт хуваах."""
    np.random.seed(random_state)
    n_samples = len(y)
    indices = np.random.permutation(n_samples)

    test_count = int(n_samples * test_size)
```

```
    test_indices = indices[:test_count]
    train_indices = indices[test_count:]

    return X[train_indices], X[test_indices], y[train_indices], y[test_indices]

# Өгөгдлийг 80/20 харьцаагаар хуваах
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Сургалтын өгөгдөл: {X_train.shape[0]:,} жишээ")
print(f"Тестийн өгөгдөл: {X_test.shape[0]:,} жишээ")
print(f"\nСургалтын зорилтот тархалт:")
print(f"  - y=0: {np.sum(y_train==0):,} ({np.mean(y_train==0)*100:.1f}%)")
print(f"  - y=1: {np.sum(y_train==1):,} ({np.mean(y_train==1)*100:.1f}%)")
```

```
Сургалтын өгөгдөл: 7,868 жишээ
Тестийн өгөгдөл: 1,967 жишээ

Сургалтын зорилтот тархалт:
  - y=0: 5,826 (74.0%)
  - y=1: 2,042 (26.0%)
```

4 Бернуллийн Наив Байес загварыг бүтээх

4.1 Загварын класс

Энэ хэсэгт Equation 5 теоремд үндэслэн Бернуллийн Наив Байес алгоритмыг Python хэлээр эхнээс нь бичнэ.

```
class BernoulliNaiveBayes:
    """
    Бернуллийн Наив Байес ангилагч.

    Математик үндэслэл:
    - Таамаглал: Ĉ = argmax_c P(C=c) ∏ᵢ P(Xᵢ|C=c)
    - Бернуллийн тархалт: P(Xᵢ|C) = θᵢc^Xᵢ · (1-θᵢc)^(1-Xᵢ)
    - Лапласын тэгшитгэл: θᵢc = (Nᵢc + α) / (Nc + 2α)
    """

    def __init__(self, alpha=1.0):
        self.alpha = alpha
        self.class_prior_ = None
        self.feature_prob_ = None
        self.classes_ = None
        self.n_features_ = None

    def fit(self, X, y):
        """Загварыг сургах - параметрүүдийг үнэлэх."""
        X = np.array(X)
        y = np.array(y)

        self.classes_ = np.unique(y)
        n_classes = len(self.classes_)
        n_samples, self.n_features_ = X.shape

        # P(C) үнэлэх
        self.class_prior_ = np.zeros(n_classes)
        for idx, c in enumerate(self.classes_):
```

```
            self.class_prior_[idx] = np.sum(y == c) / n_samples

        # P(Xᵢ=1|C) үнэлэх - Лапласын тэгшитгэлтэй
        self.feature_prob_ = np.zeros((n_classes, self.n_features_))

        for idx, c in enumerate(self.classes_):
            X_c = X[y == c]
            N_c = X_c.shape[0]

            for i in range(self.n_features_):
                N_ic = np.sum(X_c[:, i])
                self.feature_prob_[idx, i] = (N_ic + self.alpha) / (N_c + 2 * self.alpha)

        return self

    def _compute_log_likelihood(self, X):
        """Лог likelihood тооцоолох."""
        n_samples = X.shape[0]
        n_classes = len(self.classes_)
        log_prob = np.zeros((n_samples, n_classes))

        for idx, c in enumerate(self.classes_):
            log_prior = np.log(self.class_prior_[idx])
            theta = self.feature_prob_[idx]
            log_theta = np.log(theta)
            log_1_theta = np.log(1 - theta)

            log_likelihood = X @ log_theta + (1 - X) @ log_1_theta
            log_prob[:, idx] = log_prior + log_likelihood

        return log_prob

    def predict(self, X):
        """Ангилал таамаглах."""
        X = np.array(X)
        log_prob = self._compute_log_likelihood(X)
        return self.classes_[np.argmax(log_prob, axis=1)]

    def predict_proba(self, X):
        """Класс магадлалуудыг тооцоолох."""
        X = np.array(X)
        log_prob = self._compute_log_likelihood(X)

        # Log-sum-exp trick
        log_prob_max = np.max(log_prob, axis=1, keepdims=True)
        log_prob_shifted = log_prob - log_prob_max
        prob = np.exp(log_prob_shifted)
        prob = prob / np.sum(prob, axis=1, keepdims=True)

        return prob

    def get_feature_importance(self, feature_names):
        """Шинж чанаруудын ач холбогдлыг тооцоолох."""
        log_ratio = np.abs(np.log(self.feature_prob_[1]) - np.log(self.feature_prob_[0]))

        importance_df = pd.DataFrame({
            'feature': feature_names,
            'importance': log_ratio,
            'prob_class_0': self.feature_prob_[0],
            'prob_class_1': self.feature_prob_[1]
        })

        return importance_df.sort_values('importance', ascending=False)
```

```
print("BernoulliNaiveBayes класс амжилттай үүсгэгдлээ!")
```

```
BernoulliNaiveBayes класс амжилттай үүсгэгдлээ!
```

4.2 Загварыг сургах

```
# Загвар үүсгэх (α=1 Лапласын тэгшитгэлтэй)
model = BernoulliNaiveBayes(alpha=1.0)

# Сургах
model.fit(X_train, y_train)

print("Загвар амжилттай сургагдлаа!")
print(f"\nӨмнөх магадлал P(C):")
print(f"  - P(y=0) = {model.class_prior_[0]:.4f}")
print(f"  - P(y=1) = {model.class_prior_[1]:.4f}")

# Зарим нөхцөлт магадлалыг харуулах
print(f"\nЗарим нөхцөлт магадлал P(X=1|C):")
sample_features = ['yogurt', 'other vegetables', 'rolls/buns', 'soda', 'tropical fruit']
for feat in sample_features:
    if feat in item_to_idx:
        idx = item_to_idx[feat]
        print(f"  - P({feat}=1 | y=0) = {model.feature_prob_[0, idx]:.4f}")
        print(f"  - P({feat}=1 | y=1) = {model.feature_prob_[1, idx]:.4f}")
```

```
Загвар амжилттай сургагдлаа!

Өмнөх магадлал P(C):
  - P(y=0) = 0.7405
  - P(y=1) = 0.2595

Зарим нөхцөлт магадлал P(X=1|C):
  - P(yogurt=1 | y=0) = 0.1158
  - P(yogurt=1 | y=1) = 0.2172
  - P(other vegetables=1 | y=0) = 0.1604
  - P(other vegetables=1 | y=1) = 0.2911
  - P(rolls/buns=1 | y=0) = 0.1712
  - P(rolls/buns=1 | y=1) = 0.2197
  - P(soda=1 | y=0) = 0.1795
  - P(soda=1 | y=1) = 0.1546
  - P(tropical fruit=1 | y=0) = 0.0858
  - P(tropical fruit=1 | y=1) = 0.1639
```

5 Загварын үнэлгээ

5.1 Үнэлгээний хэмжигдэхүүнүүд

Ангиллын загварын гүйцэтгэлийг үнэлэхэд дараах хэмжигдэхүүнүүдийг ашиглана James et al. (2021):

- **Нарийвчлал (Accuracy)**: $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision**: $\frac{TP}{TP+FP}$
- **Recall**: $\frac{TP}{TP+FN}$
- **F1 Score**: $2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$

```
def confusion_matrix(y_true, y_pred):
    """Төөрөгдлийн матриц тооцоолох"""
    tp = np.sum((y_true == 1) & (y_pred == 1))
    tn = np.sum((y_true == 0) & (y_pred == 0))
    fp = np.sum((y_true == 0) & (y_pred == 1))
    fn = np.sum((y_true == 1) & (y_pred == 0))
    return np.array([[tn, fp], [fn, tp]])

def accuracy_score(y_true, y_pred):
    return np.mean(y_true == y_pred)

def precision_score(y_true, y_pred):
    tp = np.sum((y_true == 1) & (y_pred == 1))
    fp = np.sum((y_true == 0) & (y_pred == 1))
    return tp / (tp + fp) if (tp + fp) > 0 else 0

def recall_score(y_true, y_pred):
    tp = np.sum((y_true == 1) & (y_pred == 1))
    fn = np.sum((y_true == 1) & (y_pred == 0))
    return tp / (tp + fn) if (tp + fn) > 0 else 0

def f1_score(y_true, y_pred):
    prec = precision_score(y_true, y_pred)
    rec = recall_score(y_true, y_pred)
    return 2 * prec * rec / (prec + rec) if (prec + rec) > 0 else 0
```

5.2 Таамаглал ба үнэлгээ

```
# Тест дээр таамаглах
y_test_pred = model.predict(X_test)
y_test_proba = model.predict_proba(X_test)

# Төөрөгдлийн матриц
cm = confusion_matrix(y_test, y_test_pred)

print("ЗАГВАРЫН ҮНЭЛГЭЭ")
print("=" * 50)

print("\n1. ТӨӨРӨГДЛИЙН МАТРИЦ:")
print("                    Таамагласан")
print("                  0 (Аваагүй)  1 (Авсан)")
print(f"Бодит  0 (Аваагүй)     {cm[0,0]:>4}          {cm[0,1]:>3}")
print(f"       1 (Авсан)        {cm[1,0]:>3}          {cm[1,1]:>3}")

acc = accuracy_score(y_test, y_test_pred)
prec = precision_score(y_test, y_test_pred)
rec = recall_score(y_test, y_test_pred)
f1 = f1_score(y_test, y_test_pred)

print("\n2. ҮНЭЛГЭЭНИЙ ХЭМЖИГДЭХҮҮНҮҮД:")
print(f"  - Нарийвчлал (Accuracy):  {acc:.4f}")
print(f"  - Precision:              {prec:.4f}")
print(f"  - Recall:                 {rec:.4f}")
```

```
print(f"  - F1 Score:              {f1:.4f}")

conclusion.append(f"Загварын нарийвчлал {acc*100:.1f}%, F1 оноо {f1:.4f} байна.")
```

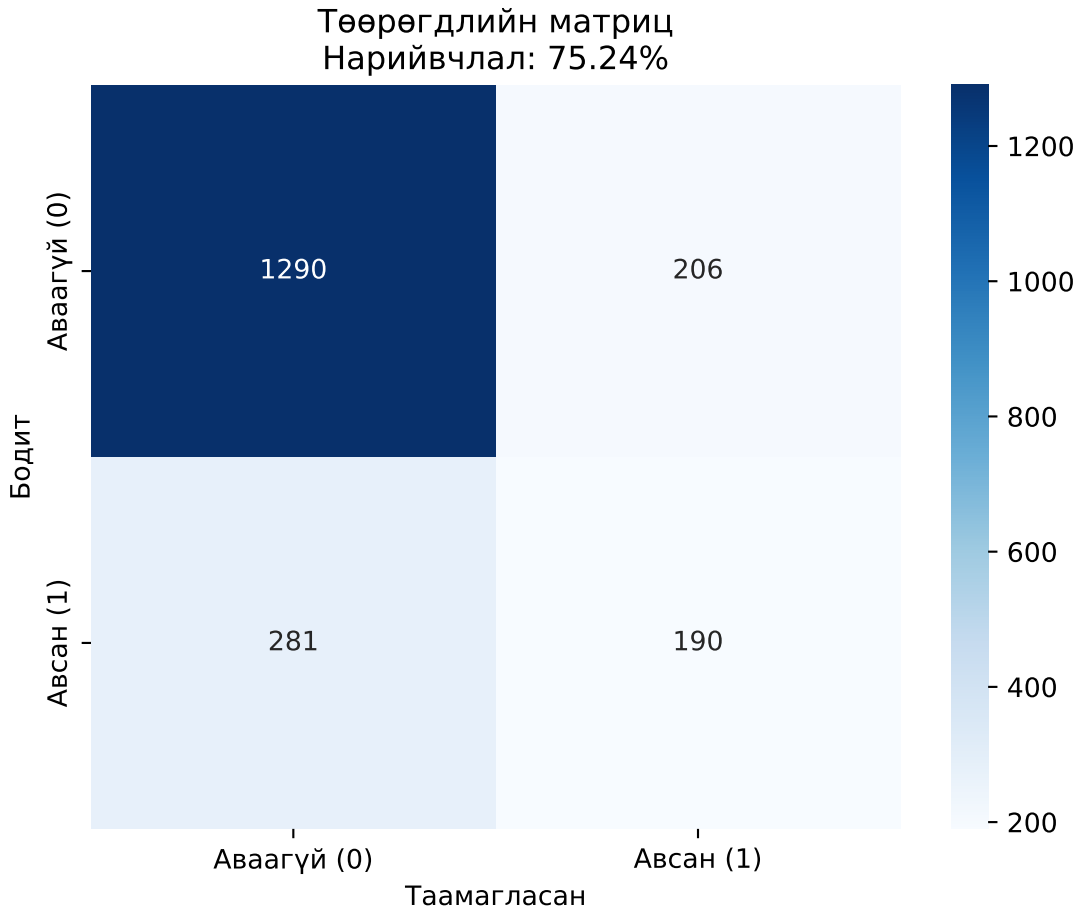
```
ЗАГВАРЫН ҮНЭЛГЭЭ
==================================================

1. ТӨӨРӨГДЛИЙН МАТРИЦ:
                    Таамагласан
                  0 (Аваагүй)  1 (Авсан)
Бодит  0 (Аваагүй)     1290          206
       1 (Авсан)        281          190

2. ҮНЭЛГЭЭНИЙ ХЭМЖИГДЭХҮҮНҮҮД:
  - Нарийвчлал (Accuracy):  0.7524
  - Precision:              0.4798
  - Recall:                 0.4034
  - F1 Score:               0.4383
```

```
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Аваагүй (0)', 'Авсан (1)'],
            yticklabels=['Аваагүй (0)', 'Авсан (1)'])
plt.xlabel('Таамагласан')
plt.ylabel('Бодит')
plt.title(f'Төөрөгдлийн матриц\nНарийвчлал: {acc:.2%}')
plt.tight_layout()
plt.show()
```

5.3 ROC муруй ба AUC

```
def roc_curve(y_true, y_score, n_thresholds=100):
    """ROC муруйн цэгүүдийг тооцоолох"""
    thresholds = np.linspace(0, 1, n_thresholds)
    tpr_list, fpr_list = [], []

    for thresh in thresholds:
        y_pred = (y_score >= thresh).astype(int)
        tp = np.sum((y_true == 1) & (y_pred == 1))
        fn = np.sum((y_true == 1) & (y_pred == 0))
        fp = np.sum((y_true == 0) & (y_pred == 1))
        tn = np.sum((y_true == 0) & (y_pred == 0))

        tpr = tp / (tp + fn) if (tp + fn) > 0 else 0
        fpr = fp / (fp + tn) if (fp + tn) > 0 else 0
        tpr_list.append(tpr)
        fpr_list.append(fpr)

    return np.array(fpr_list), np.array(tpr_list), thresholds

def auc_score(fpr, tpr):
    sorted_indices = np.argsort(fpr)
    return np.trapz(tpr[sorted_indices], fpr[sorted_indices])
```

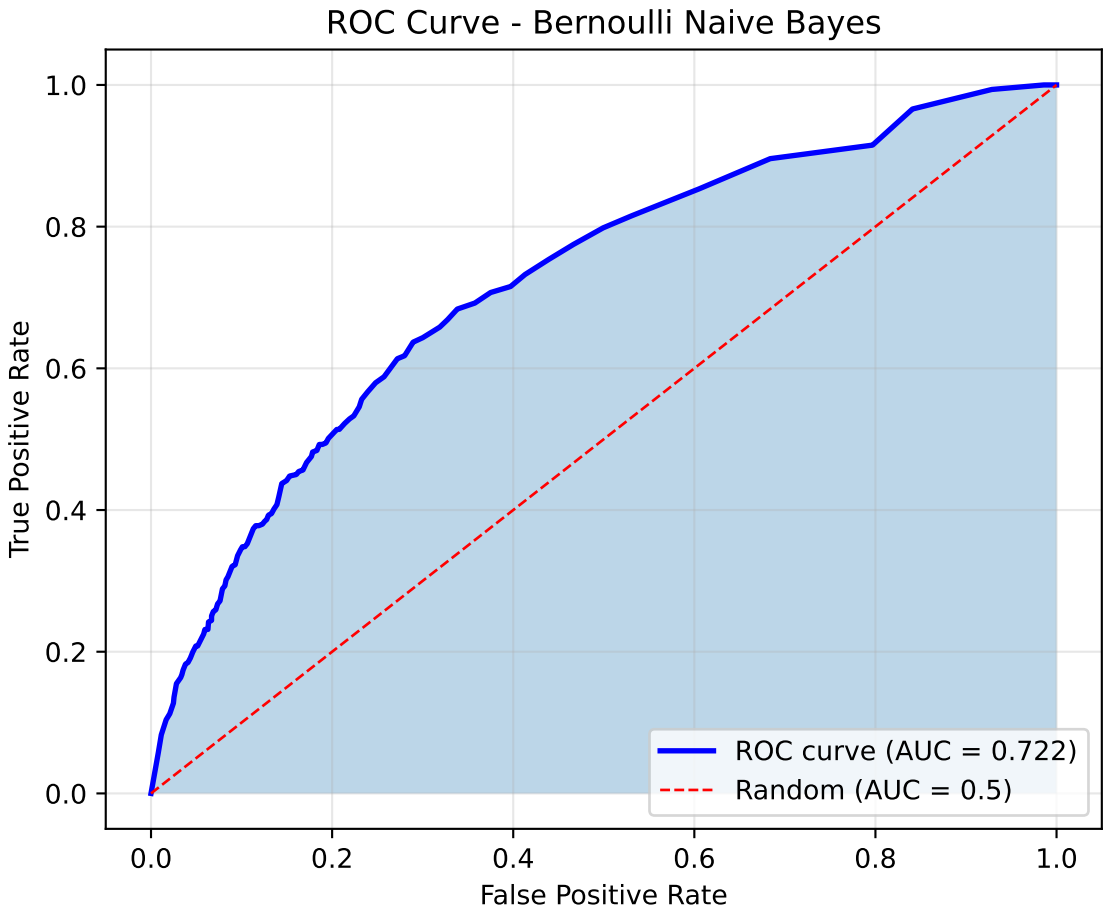
Зураг 2: Төөрөгдлийн матриц (Confusion Matrix)




```
# ROC муруй тооцоолох
fpr, tpr, thresholds = roc_curve(y_test, y_test_proba[:, 1])
auc = auc_score(fpr, tpr)

print(f"AUC-ROC Score: {auc:.4f}")

# Диаграмм
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, 'b-', linewidth=2, label=f'ROC curve (AUC = {auc:.3f})')
plt.plot([0, 1], [0, 1], 'r--', linewidth=1, label='Random (AUC = 0.5)')
plt.fill_between(fpr, tpr, alpha=0.3)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Bernoulli Naive Bayes')
plt.legend(loc='lower right')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

conclusion.append(f"AUC-ROC үзүүлэлт {auc:.4f} байна.")
```

```
AUC-ROC Score: 0.7222
```

Зураг 3: ROC муруй ба AUC үзүүлэлт



5.4 Кросс-валидаци

Загварын тогтвортой байдлыг шалгахын тулд 5-фолд кросс-валидаци хийнэ.

```python
def k_fold_cross_validation(X, y, k=5, alpha=1.0, random_state=42):
    """K-фолд кросс-валидаци хийх"""
    np.random.seed(random_state)
    n_samples = len(y)
    indices = np.random.permutation(n_samples)
    fold_size = n_samples // k

    scores = {'accuracy': [], 'precision': [], 'recall': [], 'f1': []}

    for fold in range(k):
        start_idx = fold * fold_size
        end_idx = start_idx + fold_size if fold < k - 1 else n_samples
        test_idx = indices[start_idx:end_idx]
        train_idx = np.concatenate([indices[:start_idx], indices[end_idx:]])

        X_train_fold, X_test_fold = X[train_idx], X[test_idx]
        y_train_fold, y_test_fold = y[train_idx], y[test_idx]

        model_fold = BernoulliNaiveBayes(alpha=alpha)
        model_fold.fit(X_train_fold, y_train_fold)
        y_pred_fold = model_fold.predict(X_test_fold)
```

```
        scores['accuracy'].append(accuracy_score(y_test_fold, y_pred_fold))
        scores['precision'].append(precision_score(y_test_fold, y_pred_fold))
        scores['recall'].append(recall_score(y_test_fold, y_pred_fold))
        scores['f1'].append(f1_score(y_test_fold, y_pred_fold))

    return scores

# 5-фолд кросс-валидаци
cv_scores = k_fold_cross_validation(X, y, k=5, alpha=1.0)

print("5-ФОЛД КРОСС-ВАЛИДАЦИЙН ҮР ДҮН")
print("=" * 50)

print("\nFold бүрийн үр дүн:")
for i in range(5):
    print(f"Fold {i+1}: Accuracy={cv_scores['accuracy'][i]:.4f}, F1={cv_scores['f1'][i]:.4f}")

cv_acc_mean = np.mean(cv_scores['accuracy'])
cv_acc_std = np.std(cv_scores['accuracy'])
cv_f1_mean = np.mean(cv_scores['f1'])

print(f"\nДундаж үр дүн:")
print(f"Accuracy: {cv_acc_mean:.4f} ± {cv_acc_std:.4f}")
print(f"F1 Score: {cv_f1_mean:.4f} ± {np.std(cv_scores['f1']):.4f}")

conclusion.append(f"5-фолд кросс-валидацийн дундаж нарийвчлал {cv_acc_mean*100:.1f}% (± {cv_acc_std*100:.1f}%) байна.")
```

```
5-ФОЛД КРОСС-ВАЛИДАЦИЙН ҮР ДҮН
==================================================

Fold бүрийн үр дүн:
Fold 1: Accuracy=0.7524, F1=0.4383
Fold 2: Accuracy=0.7443, F1=0.4291
Fold 3: Accuracy=0.7397, F1=0.4565
Fold 4: Accuracy=0.7336, F1=0.4279
Fold 5: Accuracy=0.7417, F1=0.4573

Дундаж үр дүн:
Accuracy: 0.7423 ± 0.0061
F1 Score: 0.4418 ± 0.0128
```

5.5 Шинж чанаруудын ач холбогдол

```
feature_importance = model.get_feature_importance(feature_items)

print("Топ 10 нөлөөтэй бүтээгдэхүүн:")
print("-" * 60)
for i, (_, row) in enumerate(feature_importance.head(10).iterrows(), 1):
    print(f"{i:2d}. {row['feature']:<25} | "
          f"P(X=1|y=0)={row['prob_class_0']:.4f} | "
          f"P(X=1|y=1)={row['prob_class_1']:.4f}")

# Визуализаци
top_n = 15
top_features = feature_importance.head(top_n)
```

```
plt.figure(figsize=(8, 6))
colors = ['green' if row['prob_class_1'] > row['prob_class_0'] else 'red'
          for _, row in top_features.iterrows()]

bars = plt.barh(range(top_n), top_features['importance'].values, color=colors, alpha=0.7)
plt.yticks(range(top_n), top_features['feature'].values)
plt.gca().invert_yaxis()
plt.xlabel('Ач холбогдол |log(P(X|y=1)/P(X|y=0))|')
plt.title(f'Топ {top_n} нөлөөтэй бүтээгдэхүүн')
plt.grid(True, alpha=0.3, axis='x')
plt.tight_layout()
plt.show()
```

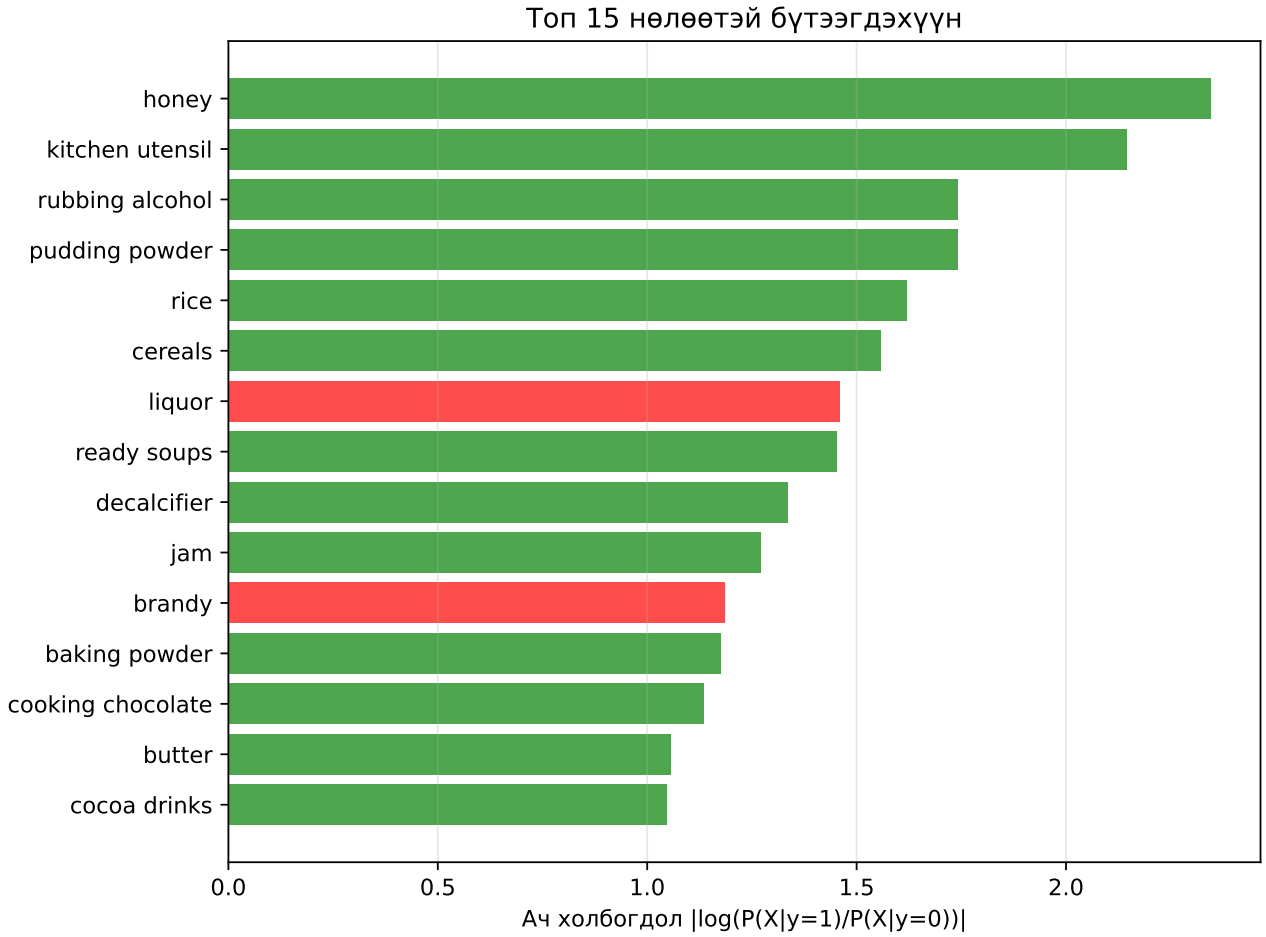
```
Топ 10 нөлөөтэй бүтээгдэхүүн:
------------------------------------------------------------
 1. honey                     | P(X=1|y=0)=0.0005 | P(X=1|y=1)=0.0054
 2. kitchen utensil           | P(X=1|y=0)=0.0002 | P(X=1|y=1)=0.0015
 3. rubbing alcohol           | P(X=1|y=0)=0.0005 | P(X=1|y=1)=0.0029
 4. pudding powder            | P(X=1|y=0)=0.0009 | P(X=1|y=1)=0.0049
 5. rice                      | P(X=1|y=0)=0.0038 | P(X=1|y=1)=0.0191
 6. cereals                   | P(X=1|y=0)=0.0031 | P(X=1|y=1)=0.0147
 7. liquor                    | P(X=1|y=0)=0.0148 | P(X=1|y=1)=0.0034
 8. ready soups               | P(X=1|y=0)=0.0010 | P(X=1|y=1)=0.0044
 9. decalcifier               | P(X=1|y=0)=0.0010 | P(X=1|y=1)=0.0039
10. jam                       | P(X=1|y=0)=0.0034 | P(X=1|y=1)=0.0122
```

6 Sklearn-тэй харьцуулалт

Манай бүтээсэн загварыг sklearn номын сангийн BernoulliNB загвартай харьцуулна Pedregosa et al. (2011).

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score as sklearn_accuracy

# Sklearn загвар сургах
sklearn_model = BernoulliNB(alpha=1.0)
sklearn_model.fit(X_train, y_train)
sklearn_pred = sklearn_model.predict(X_test)
sklearn_proba = sklearn_model.predict_proba(X_test)

# Манай загвар
our_pred = model.predict(X_test)
our_proba = model.predict_proba(X_test)

print("SKLEARN BERNOULLINB-ТЭЙ ХАРЬЦУУЛАЛТ")
print("=" * 50)

print(f"\n1. Нарийвчлал (Accuracy):")
print(f"   - Манай загвар:    {sklearn_accuracy(y_test, our_pred):.6f}")
print(f"   - Sklearn загвар:  {sklearn_accuracy(y_test, sklearn_pred):.6f}")

print(f"\n2. Таамаглалын тохирол:")
matching = np.sum(our_pred == sklearn_pred)
total = len(y_test)
print(f"   - Ижил таамаглал:  {matching:,} / {total:,} ({matching/total*100:.2f}%)")
```

Зураг 4: Шинж чанаруудын ач холбогдол (Топ 15)




```
print(f"\n3. Магадлалын ялгаа:")
prob_diff = np.abs(our_proba - sklearn_proba)
print(f"   - Дундаж ялгаа:    {np.mean(prob_diff):.8f}")
print(f"   - Хамгийн их ялгаа: {np.max(prob_diff):.8f}")

if matching/total > 0.99:
    print("\n✓ Манай загвар sklearn-тэй бараг 100% ижил таамаглал өгч байна!")
    conclusion.append("Манай бүтээсэн загвар sklearn-ийн BernoulliNB-тэй бараг 100% ижил үр дүн өгч байна.")
```

```
SKLEARN BERNOULLINB-ТЭЙ ХАРЬЦУУЛАЛТ
==================================================

1. Нарийвчлал (Accuracy):
   - Манай загвар:    0.752415
   - Sklearn загвар:  0.752415

2. Таамаглалын тохирол:
   - Ижил таамаглал:  1,967 / 1,967 (100.00%)

3. Магадлалын ялгаа:
   - Дундаж ялгаа:    0.00000000
   - Хамгийн их ялгаа: 0.00000000
```

```
✓ Манай загвар sklearn-тэй бараг 100% ижил таамаглал өгч байна!
```

Дүгнэлт

1. Загварын нарийвчлал 75.2
2. AUC-ROC үзүүлэлт 0.7222 байна.
3. 5-фолд кросс-валидацийн дундаж нарийвчлал 74.2
4. Манай бүтээсэн загвар sklearn-ийн BernoulliNB-тэй бараг 100

Энэхүү судалгааны ажил нь Бернуллийн Наив Байесын алгоритмыг математик үндэслэл, онолын суурь зарчмуудын дагуу Python хэл дээр эхнээс нь хэрэгжүүлж, Groceries өгөгдлийн сантай уялдуулан турших замаар загварын оновчтой ажиллагаа, үнэн зөв таамаглал хийх чадварыг бодит өгөгдөл дээр нотлон харууллаа.

Хязгаарлалт ба цаашдын судалгаа:

- Наив Байесын нөхцөлт үл хамаарлын таамаглал нь бодит байдалд үргэлж биелдэггүй
- Бусад машин сургалтын алгоритмуудтай (Random Forest, SVM) харьцуулалт хийх
- Илүү олон зорилтот бүтээгдэхүүнийг таамаглах олон ангиллын загвар бүтээх

Багийн гишүүдийн үүрэг оролцоо

	Гишүүн	Оюутны код	Үүрэг ба оролцоо
0	Б.Билгүүнтөгөлдөр	20B1NUM1087	Онолын үндэслэл, математик томьёолол (20%)
1	О.Энхбаяр	21B1NUM0940	Өгөгдлийн цуглуулалт, боловсруулалт (20%)
2	Э.Алтаншагай	22B1NUM0331	Загварыг эхнээс бүтээх, кодлох (25%)
3	Б.Намуундарь	23B1NUM0543	Загварын үнэлгээ, визуализаци (20%)
4	Ч.Саранцацралт	24B1NUM0019	Тайлан бичих, дүгнэлт гаргах (15%)

Ашигласан материал

Hahsler, M., Grün, B., & Hornik, K. (2006). Arules—a computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, *14*(15), 1–25. https://doi.org/10.18637/jss.v014.i15

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in R* (2nd ed.). Springer. https://doi.org/10.1007/978-1-0716-1418-1

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets* (2nd ed.). Cambridge University Press.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.