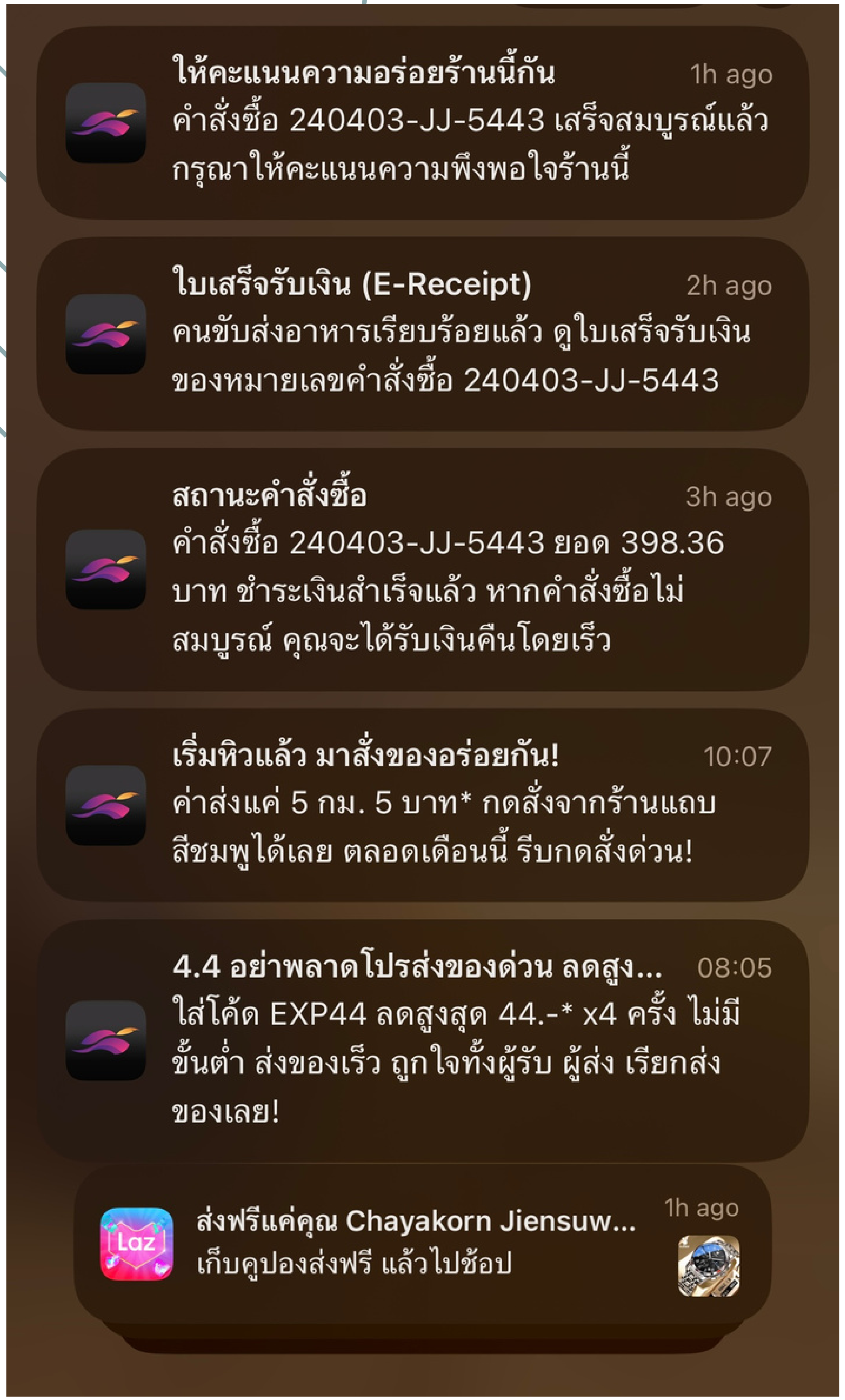
THAI MOBILE APP NOTIFICATION FILTER

OVERVIEW

- Aiming at classifying mobile application notifications into two categories:
  - "ham" (bill reminding, delivery noticing, non-advertising related)
  - "spam" (marketing, advertising related, etc.)
- Solving the issue of receiving mixed notifications like
  - delivery status updates and marketing offers, from mobile applications


ให้คะแนนความอร่อยร้านนี้กัน
1h ago
คำสั่งซื้อ 240403-JJ-5443 เสร็จสมบูรณ์แล้ว กรุณาให้คะแนนความพึงพอใจร้านนี้
ใบเสร็จรับเงิน (E-Receipt)
2h ago
คนขับส่งอาหารเรียบร้อยแล้ว ดูใบเสร็จรับเงินของหมายเลขคำสั่งซื้อ 240403-JJ-5443
สถานะคำสั่งซื้อ
3h ago
คำสั่งซื้อ 240403-JJ-5443 ยอด 398.36 บาท ชำระเงินสำเร็จแล้ว หากคำสั่งซื้อไม่สมบูรณ์ คุณจะได้รับเงินคืนโดยเร็ว
เริ่มหิวแล้ว มาสั่งของอร่อยกัน!
10:07
ค่าส่งแค่ 5 กม. 5 บาท* กดสั่งจากร้านแถบสีชมพูได้เลย ตลอดเดือนนี้ รีบกดสั่งด่วน!
4.4 อย่าพลาดโปรส่งของด่วน ลดสูง...
08:05
ใส่โค้ด EXP44 ลดสูงสุด 44.-* x4 ครั้ง ไม่มีขั้นต่ำ ส่งของเร็ว ถูกใจทั้งผู้รับ ผู้ส่ง เรียกส่งของเลย!
Laz
ส่งฟรีแค่คุณ Chayakorn Jiensuw...
1h ago
เก็บคูปองส่งฟรี แล้วไปช้อป


Laz
ส่งฟรีแค่คุณ Chayakorn Jiensuw...
2h ago
เก็บคูปองส่งฟรี แล้วไปช้อป
ไอเดีย APPLE AirPods Pro (รุ่นที่ 2...
17:29
เรากำลังจัดโปรลดราคาอยู่นะ
💰 Earn up to ฿100 LazRewards!
16:55
Watch and earn here 👉
fire emblem warriors ลดราคาแล้ว
15:11
ส่วนลดพิเศษ fire emblem warriors รีบช้อป
⏰ ลาซาด้าช่วยเปย์ เปย์อีก 1,500.-...
14:00
พรุ่งนี้แล้ว! 4.4 ลดสับรับซัมเมอร์ 🌞
Leacat Store
13:02
[Voucher] ทักครับ! สินค้าใหม่น่าสนใจเพีย...
Apple Acc EarPods with 3.5mm...
12:14
เรากำลังจัดโปรลดราคาอยู่นะ
สุดค...

DATASET

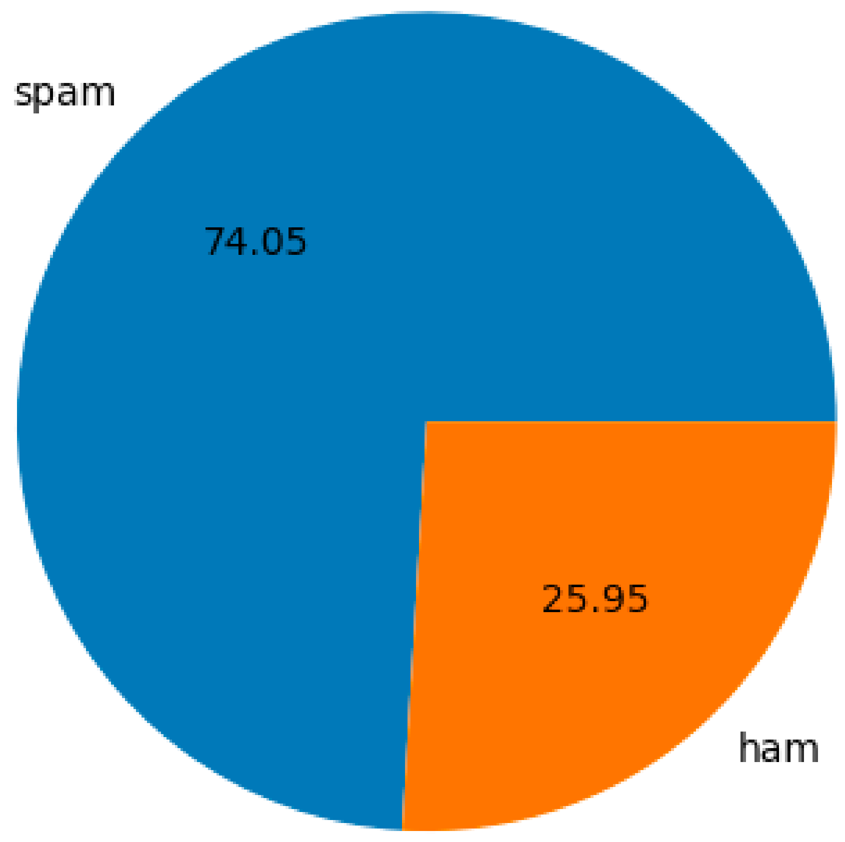
LABEL : "HAM" OR "SPAM"

CONTENT : TEXT OF THE NOTIFICATION

MAINLY IN THAI LANGUAGE FOCUSING ON SHOPPING, FOOD DELIVERY, AND SERVICE-RELATED APPLICATIONS.

Dimension of dataset: (1056, 2)

	label	content
0	0	คนขับส่งอาหารเรียบร้อยแล้ว ดูใบเสร็จรับเงิน ขอ...
1	0	หมายเลข: 09116240327514498 อยู่ในสถานะ 'เสร็จส...
2	0	หมายเลข: 09116240327514498 พนักงานกำลังน้ำ สิน...
3	0	7-Delivery: เนื่องจากขณะนี้มีออเดอร์เป็นจำนวนม...
4	0	หมายเลข: 09116240327514498 อยู่ในสถานะ 'จัดของ...
...	...	...
1051	1	เที่ยงคืน แจกอีก5,000.- เก็บด่วน! ช้อปดีลเด็ด ...
1052	1	4.4 มาแล้ว! ลดโหด โค้ด 50% วันนี้วันเดียว!
1053	1	โค้ดลด 50% 9 โมง » เก็บด่วน
1054	1	Shopee Live! watch cintage.official's live: co...
1055	1	โค้ดเด็ด! 4.4 ลด 50% ถึงตีสอง



PREPROCESSING

Thai language behavior is handled differently from **English language**, which we handle these following steps to **streamline the feature set for modeling**

	label	content	transformed_content
0	0	คนขับส่งอาหารเรียบร้อยแล้ว ดูใบเสร็จรับเงิน ขอ...	คนขับส่งอาหารเรียบร้อยดูใบเสร็จรับเงินหมายเลขส...
1	0	หมายเลข: 09116240327514498 อยู่ในสถานะ 'เสร็จส...	หมายเลข09116240327514498อยู่สถานะสมบูรณ์date15...
2	0	หมายเลข: 09116240327514498 พนักงานกำลังน้ำ สิน...	หมายเลข09116240327514498พนักงานน้ำสินค้าส่งกรุ...
3	0	7-Delivery: เนื่องจากขณะนี้มีออเดอร์เป็นจำนวนม...	7deliveryออเดอร์จำนวนมากพนักงานดำเนินการจัดส่ง...
4	0	หมายเลข: 09116240327514498 อยู่ในสถานะ 'จัดของ...	หมายเลข09116240327514498อยู่สถานะdate1428นาฬิกา
...	...	...	...
1051	1	เที่ยงคืน แจกอีก5,000.- เก็บด่วน! ช้อปดีลเด็ด ...	เที่ยงคืนแจก5000ด่วนช้อปดีลเด็ด44จำกัดเวลาตี2
1052	1	4.4 มาแล้ว! ลดโหด โค้ด 50% วันนี้วันเดียว!	44ลดโหดโค้ด50
1053	1	โค้ดลด 50% 9 โมง » เก็บด่วน	โค้ดลด509โมงด่วน
1054	1	Shopee Live! watch cintage.official's live: co...	shopeelivewatchcintageofficialslivecode50200บา...
1055	1	โค้ดเด็ด! 4.4 ลด 50% ถึงตีสอง	โค้ดเด็ด44ลด50ตีสอง

01 - REMOVING ACRONYMS

02 - SUBSTITUTING DATES WITH A GENERIC PLACEHOLDER

03 - REMOVE STOP WORDS AND NON-THAI/ALPHANUMERIC CHARACTERS

PREPROCESSING SAMPLE

```python
def transformContent(string: str) -> str:
    cleanedString = removeAcronym(string, 'acronyms.txt') # Remove acronym
    cleanedString = dateSubstitute(cleanedString) # Date Substitution
    cleanedString = re.sub(r'[^\u0E00-\u0E7F\w]', '', cleanedString) # Remove non-Thai / alphanumeric characters using unicode range
    cleanedString = cleanedString.lower()
    cleanedString = removeStopWord(cleanedString) # Remove stop words
    return cleanedString

sample = "โค้ดพิเศษ สำหรับคุณ! ลดเพิ่ม 250.- สำหรับลูกค้าใหม่ Ray-Ban TH Official Store © เมื่อช้อปในวันที่ 3 เม.ย. 67 นี้ วันเดียวเท่านั้น! ใช้โค้ดเลย"

print("Original sample: ", sample)
print("Transformed sample: ", transformContent(sample))
```

✓ 0.0s

```
Original sample:  โค้ดพิเศษ สำหรับคุณ! ลดเพิ่ม 250.- สำหรับลูกค้าใหม่ Ray-Ban TH Official Store © เมื่อช้อปในวันที่ 3 เม.ย. 67 นี้ วันเดียวเท่านั้น! ใช้โค้ดเลย
Transformed sample:  โค้ดพิเศษสำหรับลด250สำหรับลูกค้าraybanthofficialstoreช้อปวันที่dateโค้ด
```

PREPROCESSING

```python
tfidf = TfidfVectorizer(max_features=2000)
X = tfidf.fit_transform(noti_dataset['transformed_content']).toarray()
y = noti_dataset['label'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((844, 2000), (212, 2000), (844,), (212,))
```

TfidfVectorizer
A method for converting a collection of raw documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features

train_test_split : Train 80% / Test 20%

MODEL SELECTION
MULTINOMIAL NAIVE BAYES

1

Text Classification Task

categorizing text documents or messages into predefined classes based on their content.

2

Multinomial Naive Bayes Classifier

based on Bayes' theorem, the assuming that the features (words) are conditionally independent given the class label.

3

Robust to

- irrelevant features
- noisy data

4

In practice

Real-world applications show that MNB perform well on text classification tasks, like spam detection.

MODEL TRAINING

FIT

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}.$$

- Compute the P(y) (Prior) and P(x|y) (Likelihood)
- P(y) = count(y) / count(Y) [Y = all samples, y = specific class]
- P(x|y) = count(x, y) + 1 / count(y) + |X|
  [X = all features, x = specific feature]

```python
def fit(self, X, y):
    """
    Prior : P(y) = count(y) / count(Y) [ Y = all samples, y = specific class ]
    Likelihood : P(x|y) = count(x, y) + 1 / count(y) + |X| [ X = all features, x = specific feature ]
    """
    self.classes = np.unique(y)
    self.class_prior = np.zeros(len(self.classes))
    self.feature_prob = np.zeros((len(self.classes), X.shape[1])) #

    for i, c in enumerate(self.classes): #
        X_c = X[y == c] # Get samples with class c
        self.class_prior[i] = X_c.shape[0] / X.shape[0] # P(y) = count(y) / count(Y)
        self.feature_prob[i] = (np.sum(X_c, axis=0) + 1) / (np.sum(X_c) + X.shape[1]) # P(x|y) = count(x, y) + 1 / count(y) + |X|
```

PREDICT PROBABILITY

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

- log_likelihood = log($P(x \mid y)$) × XT
- *Compute log of the likelihood of each feature given each class*
- *Use feature probabilities learned during model training phase*

```
def predict_proba(self, X):
    """
    Calculate the posterior probability for each class given the input features X.

    Parameters:
    - X: Input features, shape (n_samples, n_features)

    Returns:
    - posterior probabilities: Probability of each class for each sample in X, shape (n_classes, n_samples)
    """
    # Compute the logarithm of the likelihood of each feature given each class, then transpose the result
    log_likelihood = np.log(self.feature_prob + 1e-9) @ X.T  # log(P(x|y)) * X (dot product [ @ : matrix multiplication ])

    # Add the logarithm of the prior probability of each class to the likelihood
    log_likelihood += np.log(self.class_prior.reshape(-1, 1))

    # Normalize the log-likelihoods by subtracting the maximum log-likelihood value to avoid numerical instability
    log_likelihood -= np.max(log_likelihood, axis=0)

    # Exponentiate the normalized log-likelihoods to get the likelihoods
    likelihood = np.exp(log_likelihood)
    return likelihood / np.sum(likelihood, axis=0) # Normalize the likelihoods to get the posterior probabilities
```

PREDICT PROBABILITY

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

- *log_likelihood += log(P(y))*
- *Adds log of the prior probability of each class to log-likelihood*
- *Adjusts likelihoods based on the prior probability of each class*

```python
def predict_proba(self, X):
    """
    Calculate the posterior probability for each class given the input features X.

    Parameters:
    - X: Input features, shape (n_samples, n_features)

    Returns:
    - posterior probabilities: Probability of each class for each sample in X, shape (n_classes, n_samples)
    """
    # Compute the logarithm of the likelihood of each feature given each class, then transpose the result
    log_likelihood = np.log(self.feature_prob + 1e-9) @ X.T  # log(P(x|y)) * X (dot product [ @ : matrix multiplication ])

    # Add the logarithm of the prior probability of each class to the likelihood
    log_likelihood += np.log(self.class_prior.reshape(-1, 1))

    # Normalize the log-likelihoods by subtracting the maximum log-likelihood value to avoid numerical instability
    log_likelihood -= np.max(log_likelihood, axis=0)

    # Exponentiate the normalized log-likelihoods to get the likelihoods
    likelihood = np.exp(log_likelihood)
    return likelihood / np.sum(likelihood, axis=0) # Normalize the likelihoods to get the posterior probabilities
```

PREDICT PROBABILITY

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

- *log_likelihood -= max(log_likelihood)*
- *log-likelihood normalization, performed to address numerical instability issues which occur when dealing with very small or very large values (logarithms)*

```python
def predict_proba(self, X):
    """
    Calculate the posterior probability for each class given the input features X.

    Parameters:
    - X: Input features, shape (n_samples, n_features)

    Returns:
    - posterior probabilities: Probability of each class for each sample in X, shape (n_classes, n_samples)
    """
    # Compute the logarithm of the likelihood of each feature given each class, then transpose the result
    log_likelihood = np.log(self.feature_prob + 1e-9) @ X.T  # log(P(x|y)) * X (dot product [ @ : matrix multiplication ])

    # Add the logarithm of the prior probability of each class to the likelihood
    log_likelihood += np.log(self.class_prior.reshape(-1, 1))

    # Normalize the log-likelihoods by subtracting the maximum log-likelihood value to avoid numerical instability
    log_likelihood -= np.max(log_likelihood, axis=0)

    # Exponentiate the normalized log-likelihoods to get the likelihoods
    likelihood = np.exp(log_likelihood)
    return likelihood / np.sum(likelihood, axis=0) # Normalize the likelihoods to get the posterior probabilities
```

PREDICT PROBABILITY

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

- *likelihood = exp(log_likelihood)*
- *posterior_prob = likelihood / Σ likelihood*

```python
def predict_proba(self, X):
    """
    Calculate the posterior probability for each class given the input features X.

    Parameters:
    - X: Input features, shape (n_samples, n_features)

    Returns:
    - posterior probabilities: Probability of each class for each sample in X, shape (n_classes, n_samples)
    """
    # Compute the logarithm of the likelihood of each feature given each class, then transpose the result
    log_likelihood = np.log(self.feature_prob + 1e-9) @ X.T  # log(P(x|y)) * X (dot product [ @ : matrix multiplication ])

    # Add the logarithm of the prior probability of each class to the likelihood
    log_likelihood += np.log(self.class_prior.reshape(-1, 1))

    # Normalize the log-likelihoods by subtracting the maximum log-likelihood value to avoid numerical instability
    log_likelihood -= np.max(log_likelihood, axis=0)

    # Exponentiate the normalized log-likelihoods to get the likelihoods
    likelihood = np.exp(log_likelihood)
    return likelihood / np.sum(likelihood, axis=0) # Normalize the likelihoods to get the posterior probabilities
```

CLASS PREDICTION

```python
def predict(self, X):
    """
    Predict the class with the highest probability by finding the argmax of the posterior probability.
    """
    return np.argmax(self.predict_proba(X), axis=0)
```

$$\operatorname*{argmax}_{c} P(y = c \mid \mathbf{x}) \propto \operatorname*{argmax}_{c} \hat{\pi}_c \prod_{\alpha=1}^{d} \hat{\theta}_{\alpha c}^{x_\alpha}$$

Selects the class with the highest probability (argmax)
as the predicted class for each sample

PERFORMANCE METRICS

```python
def accuracy_score(self, y_true, y_pred):
    correct_predictions = np.sum(y_true == y_pred)
    total_samples = len(y_true)
    accuracy = correct_predictions / total_samples
    return accuracy
```

- measures the proportion of correctly classified samples out of the total number of samples
- Accuracy = Number of Correct Predictions / Total Number of Predictions

PERFORMANCE METRICS

```
def precision_score(self, y_true, y_pred, pos_label=0):
    true_positives = np.sum((y_true == pos_label) & (y_pred == pos_label))
    predicted_positives = np.sum(y_pred == pos_label)
    precision = true_positives / predicted_positives if predicted_positives > 0 else 0
    return precision
```

- Measure of the accuracy of positive predictions made by the model
- Precision = True Positives / True Positives + False Positives
- Indicates the model's ability to avoid false positive predictions.

PERFORMANCE METRICS

```python
# Calculate accuracy score
accuracy = accuracy_score(y_test, predictions)
print("Accuracy score:", accuracy)

# Calculate precision score
precision_score = precision_score(y_test, predictions)
print("Precision score:", precision_score)
```

✓ 0.0s

```
Accuracy score: 0.9339622641509434
Precision score: 0.9176470588235294
```

REFERENCES

- **Multinomial Naive Bayes**
  - Cornell University - CS4780 Lecture Notes
- **Slides**
  - Colorful Modern Business Infographic Presentation