Name: Sai Ram Gunturu
ID: C00313478

Naive Bayes Classifier - SMS Spam Detection

Data Analytics & Algorithms Log

- **Algorithm Used:** Naive Bayes (MultinomialNB)
- **Dataset Used:** SMS Spam Collection
- **Framework:** CRISP-DM

---

1. Business Understanding

Objective

- To build a **Naive Bayes Classifier** to classify SMS messages as spam or ham.
- To improve efficiency, I implemented **data balancing techniques (SMOTE & ADASYN)** to handle class imbalance.
- Spam messages are underrepresented, so I had to find a way for the **model to learn spam patterns better**.

Challenges I Faced

- **Class Imbalance:**
  - The dataset I chose had **87% ham messages**, leading to **biased predictions**.
- **Processing Text Data:**
  - I used **TF-IDF & CountVectorizer**, to convert text into numbers, removed **stopwords**, applied **stemming**, and tested **n-grams**.
- **Overfitting Risks due to Resampling:**
  - Since SMOTE & ADASYN create synthetic data, I had to **make sure the model generalizes well**.

2. Data Understanding

Dataset Overview

- **Total Messages:** 5,572
- **Class Distribution:**
  - **Ham (Authentic Messages):** 4,825
  - **Spam (Unsolicited Messages):** 747
- No missing values
- Converted categorical labels: **(Spam → 1, Ham → 0) s**

Initial Class Distribution

Class	Count	Percentage
Ham (0)	4,825	87%

Spam (1)	747	13%

Key Insight: I needed **oversampling techniques** to balance the dataset, because spam messages were much fewer

3. Data Preparation

- **Text Preprocessing:** Utilized **CountVectorizer & TF-IDF** to convert text into numerical format
- Removed **stopwords**, applied **stemming and n-grams**.
- **Train-Test Split: 80% training, 20% testing**.

4. Baseline Naive Bayes Model

Algorithm: Multinomial Naive Bayes

Since Naive Bayes performs well on text classification, I started with **MultinomialNB**.

Baseline Model Performance:

Metric	Value
Accuracy	96.68%
Precision (Spam)	100%
Recall (Spam)	75%
F1-score (Spam)	86%

Confusion Matrix:

True Label	Predicted Ham	Predicted Spam
Ham (0)	966	0
Spam (1)	37	112

What I Noticed:

- **High accuracy (96.68%)**, but **spam recall was low (75%)**.
- **Spam messages were often misclassified as ham** → needed a **better way to handle imbalance**.

5. Using SMOTE for Class Balancing

To enable the model to learn spam patterns more effectively, I decided to **oversample the spam messages using SMOTE**

Performance After SMOTE:

True Label	Predicted Ham	Predicted Spam
Ham (0)	953	13
Spam (1)	6	143

Key Impact of SMOTE

- **Spam recall went up from 75% → 96%**.
- **False negatives decreased from 37 → 6** (meaning less spam messages got incorrectly classified as ham).
- **False positives increased a little bit (0 → 13)**, but this was fine since recall improved a lot.

6. Using ADASYN for Class Balancing

Another technique I tried was **ADASYN**, which **generates synthetic spam samples only where needed**.

Performance After ADASYN

True Label	Predicted Ham	Predicted Spam
Ham (0)	967	19
Spam (1)	242	687

What I Found:

- **Spam recall fell to 74%** (worse than SMOTE).
- **False negatives increased highly (6 → 242)**, which means the model **missed a lot of spam messages**.

7. **Final Model Comparison**

Metric	Baseline Model	SMOTE Model	ADASYN Model
Accuracy	96.7%	95.8%	86.37%
Recall (Spam Detection)	75%	96%	74%
False Negatives (Missed Spam)	37	6	242
False Positives (Misclassified Ham)	0	13	19

Final Decision: SMOTE performed best.

- ### Dataset Change: Diabetes Classification

This time I switched to the **Diabetes Classification dataset** as I wanted to see how **Naive Bayes performs on structured numerical data**.

Dataset Overview:

Feature	**Type**	**Description**
glucose	Integer	Glucose level
blood pressure	Integer	Blood pressure
diabetes	Integer	Target (0 = No, 1 = Yes)

- No missing values
- **Well balanced dataset** (50% diabetic, 50% non-diabetic)

4. Baseline Naive Bayes Model (Diabetes)

Algorithm: Gaussian Naive Bayes

Since my data was numerical, I opted **Gaussian Naive Bayes**, which presumes a **normal distribution**.

Performance Before Tuning:

Metric	Value
Accuracy	90.95%
Precision (Diabetes)	91%
Recall (Diabetes)	91%
F1-Score (Diabetes)	91%

5. Hyperparameter Tuning

I used **GridSearchCV**. for tuning *var_smoothing,* to **optimize performance**,

Best Parameter: *var_smoothing = 0.0001.*

Performance After Tuning

Metric	Value
Accuracy	90.95% (Same)
Precision (Diabetes)	91%
Recall (Diabetes)	91%

Tuning had no impact → Default model was already efficient (optimal).

6. Conclusion

- **SMS Spam Detection: SMOTE worked best**, making spam recall rise from **75% → 96%**.
- **Diabetes Prediction:** Naive Bayes worked **exceptionally well (91% accuracy) with no tuning or resampling**.
- **What I Learned:**
    - **Text vs. Numerical Data:** The ML models behave differently based on data type.
    - **Class Imbalance Matters:** Resampling worked for **spam detection** but wasn't needed for **diabetes classification**.