## **Naive Bayes Classifier - SMS Spam Detection and Diabetes Classification**

### **Data Analytics & Algorithms Log**

* **Algorithm Used:** Naive Bayes (MultinomialNB and GaussianNB)
* **Datasets Used:**
  + **SMS Spam Detection (Predicting spam or ham)**
  + **Diabetes Dataset (Person is having diabetes or not)**
* **Framework:** CRISP-DM
* **Original Notebooks Link :** [**Notebook 1**](https://github.com/r21gh/SAKI-2021/blob/main/naive-bayes-tennis-1.ipynb) **and** [**Notebook 2**](https://github.com/r21gh/SAKI-2021/blob/main/naive-bayes-tennis-2.ipynb)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phases** | **Changes Made** | **Reason for the change** | **Duration** | **Difficulty level**  **(1-10)** |
| **Datasets selection** | Changed the dataset from Tennis Dataset to SMS Spam Detection and Diabetes Classification | Since the dataset in the original notebook (tennis dataset) worked out well, so I took challenging datasets namely “SMS Spam Detection” and “Diabetes Classification”. | 50 min | 5 |
| **Algorithm Version** | Changed from GaussianNB to Multinomial NB | Multinomial NB is best from classifying text data which is relevant to my dataset SMS Spam detection | 30 min | 6 |
| **Data Preprocessing** | Preprocessed the data according to the type  Methods used: CountVectorizer, TF-IDF Transformer | Converted text data to numerical form for the model to understand the data in an efficient way | 30 min | 6 |
| **Training and baseline model prediction** | No changes made | In the results, accuracy was impressive but found a huge data imbalance. | 10 min | 4 |
| **Further Enhancements (SMS Spam Detection)** | SMOTE (Synthetic Minority Oversampling Technique) | Due to highly imbalanced data, I choose SMOTE oversampling technique for its synthetic samples making. | 2 hours | 8 |
| ADASYN (Adaptive Synthetic Sampling) | I found significant improvement in spam recall and slight improvement in overall accuracy from the above method, I thought to make it more accurate in synthetic sample making process, so chosen ADASYN. |
| **Further Enhancements (Diabetes classification)** | Gaussian NB | I switched from Multinomial NB to Gaussian NB since I’m dealing with numerical data. | 2 hours | 8 |
| Hyper parameter tuning (GridSearchCV) | Although baseline model performed exceptionally well, just tried to improve the overall accuracy and recall by applying  GridSearchCV |
| **Conclusions** |  | Both GaussianNB and Multinomial NB worked really well in two datasets because of the type of the datasets (text and numerical respectively) I have used. Further enhancements worked out really well in the SMS dataset due to imbalance of data. But in the Diabetes dataset the base model itself performed exceptionally well due to balance in the dataset.  Initially I thought ADASYN was better than SMOTE and started implementing both on the SMS dataset. In the end SMOTE worked really well than ADASYN. Through this I learned that it is better to try different techniques to balance the data before coming to a conclusion.  Finally, through this algorithm's implementation (from Decision Trees to Naive Bayes) I understood that data balance is crucial, if dataset is imbalanced, trying out different and effective techniques to balance the dataset will play a crucial role in the improvement of the model. | 40 min | 6 |

### **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**.

### **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

### **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**.

### **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**.

### **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.

### **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**.

### **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 for 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**.