Naïve Bayes Classifier

Group 1:

Thale Eliassen Fink, Nicole Quattrini, Cameron Louis Penne, Aria Alinejad, Anette Fagerheim Bjerke, M. Tsaqif Wismadi, Bjørnar Kaarevik

Outline

- Bayes' Theorem
- Naïve Bayes Classifiers
- Different types
- Examples
- Common cases
- Pros & Cons
- What not to do

Bayes' Theorem

- A type of conditional probability of an event:

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

  - $P(A|B)$: probability of event A occurring <u>given</u> that B is true;
  - $P(B|A)$: probability of event B occurring <u>given</u> A is true;
  - $P(A)$ & $P(B)$: probabilities of A and B respectively.

Bayes' Theorem – Example

- The probability of having Really Terrible Disease A = 0.01 (😱)
- The probability of *testing positive* <u>given</u> you have the disease = 0.99

Q: What is the probability of having the disease given the positive test result?

Bayes' Theorem – Example

- $P(D)$ = Prob. having disease = 0.01
- $P(D')$ = Prob. not having disease = 0.99
- $P(T|D)$ = Prob. +ve test *given* having disease = 0.99
- $P(T|D')$ = Prob. +ve test *given* not having disease = 0.01
- $P(D|T)$ = ??

- The probability of having Really Terrible Disease A = 0.01 (😱)
- The probability of *testing positive* given you have the disease = 0.99

Q: What is the probability of having the disease given the positive test result?

Bayes' Theorem – Example

- $P(D)$ = Prob. having disease = 0.01
- $P(D')$ = Prob. not having disease = 0.99
- $P(T|D)$ = Prob. +ve test *given* having disease = 0.99
- $P(T|D')$ = Prob. +ve test *given* not having disease = 0.01
- $P(D|T)$ = ??

$$P(D|T) = \frac{P(T|D) \cdot P(D)}{P(T)}$$

$P(T)$ = Total probability of testing positive

= **0.0198**

$$P(D|T) = \frac{0.99 \cdot 0.01}{0.0198}$$

$$P(D|T) \approx 0.5$$

Even with a positive test result, there is a 50% chance you have the disease.

Naïve Bayes Classifiers

- Family of classifiers based on Bayes' Theorem
- ***Naïve***: assume feature independence
- Used for classification

Naïve Bayes Classifiers

Assumptions:

- Feature independence
- Normal distribution (continuous variables)
- Multinomial distribution (categorical variables)
- Equal importance across features
- No missing data

Naïve Bayes Classifiers

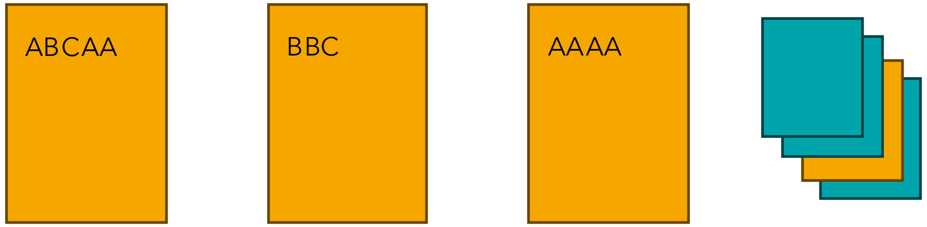
$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

Y - class

X - feature

P(Y =y_1) = Number of y_1 in data set/Total elements in Y

Multinomial & Bernoulli Naive Bayes



Bernoulli

$$\mathrm{P}(\mathrm{A}|\mathrm{yellow\ doc.}) = \frac{\#yellow\ docs.with\ feature}{\#yellow\ docs.} = \frac{2}{3}$$

Multinomial

$$\mathrm{P}(\mathrm{A}|\mathrm{yellow\ doc.}) = \prod \frac{\#\mathrm{A}\ in\ this\ yellow\ doc.}{\#features\ in\ this\ yellow\ doc.} = \frac{3}{5} * \frac{0}{3} * \frac{4}{4} = \frac{0}{60}$$

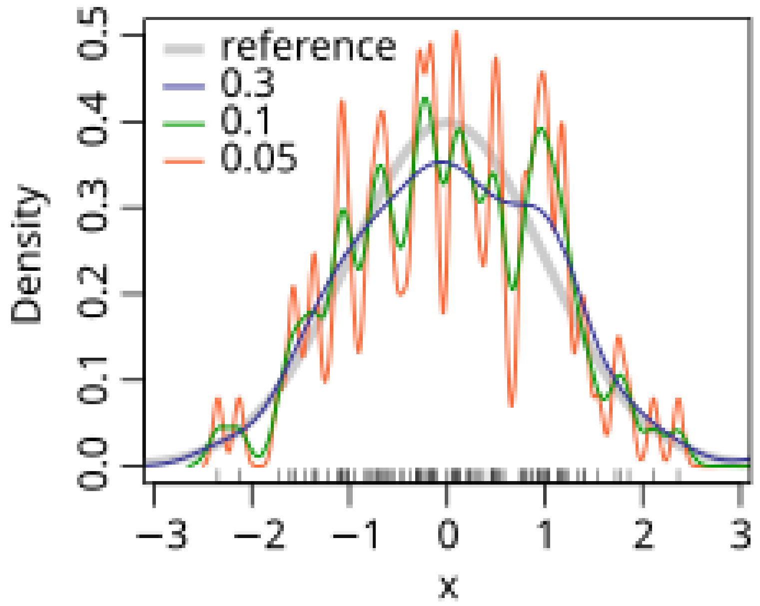
Gaussian Naïve Bayes

- Assumes normal distributed features
  - Often far from true
- Handles continous data
  - Binning or Kernel density estimation can also be used

$$P(x_i|y_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp(-\frac{(x_i - \mu_j)^2}{2\sigma_j^2})$$

Kernel density estimation (KDE)

- Non-parametric method to estimate the probability density function of a random variable based on kernels as weights
- Kernel must be chosen
  - Uniform, normal, Epanechnikov, …
- Smoothing parameter must be chosen
  - Bias-variance trade off

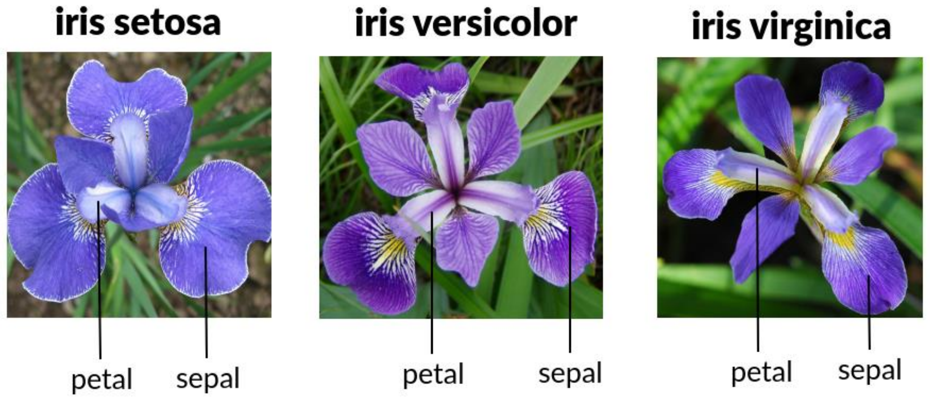


Example – Iris Flower Dataset

```python
# Load iris dataset
iris = datasets.load_iris()

# Select petal length (index 2) and petal width (index 3)
X = iris.data[:, [2, 3]]
y = iris.target

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```



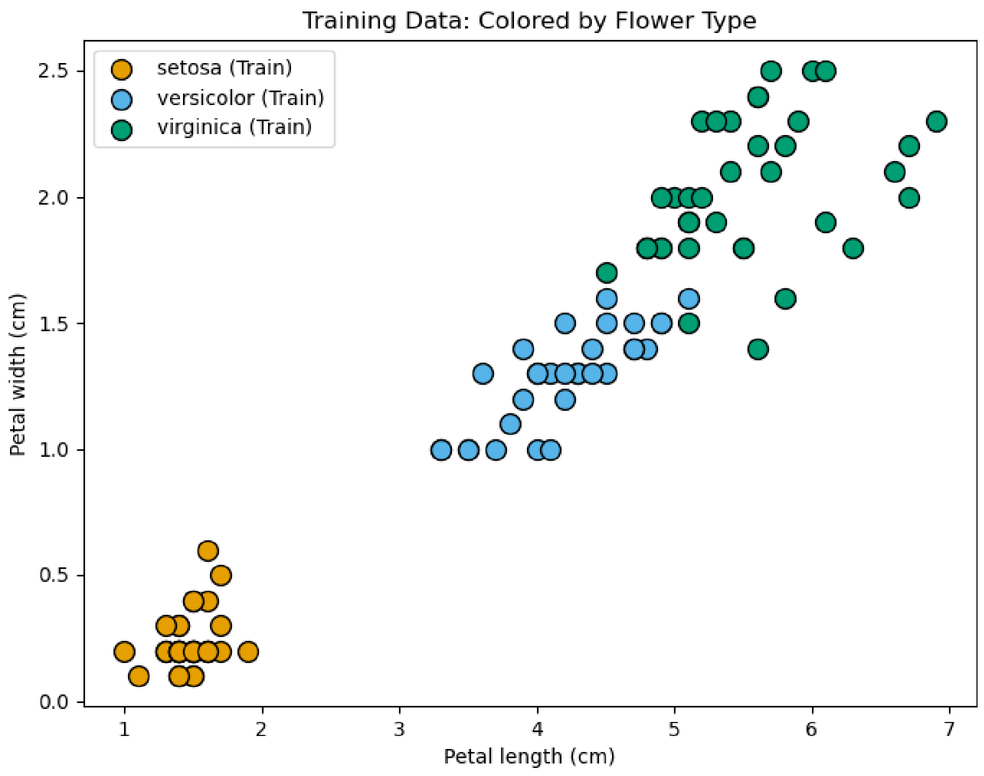
Example – Iris Flower Dataset

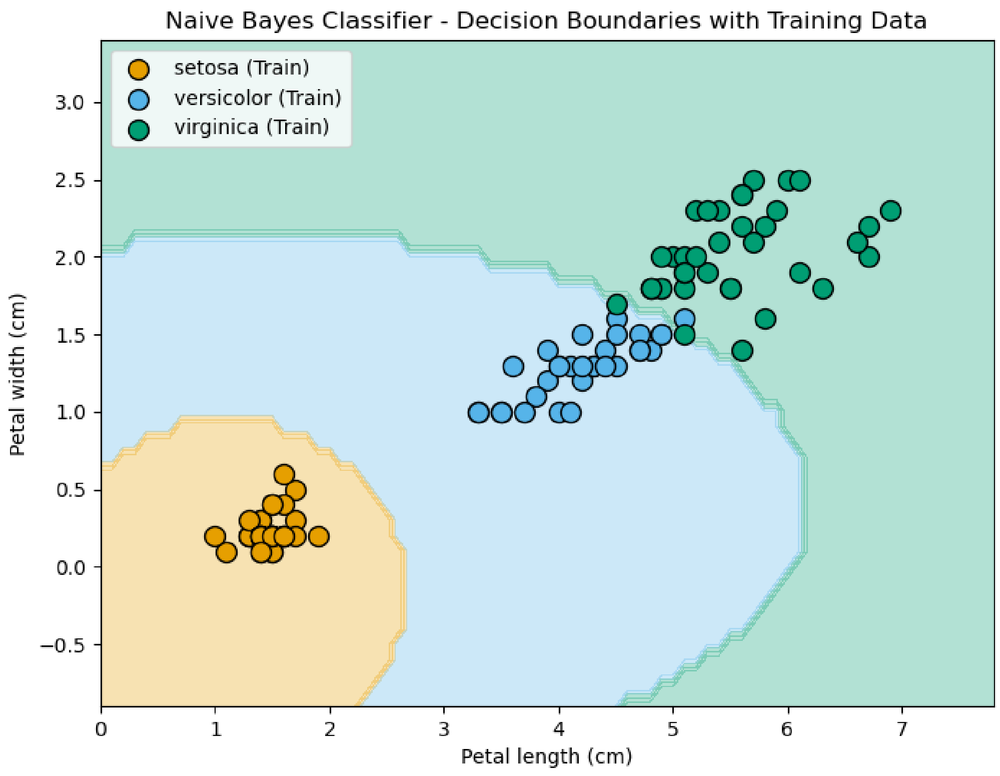
```
# Training the model
gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred = gnb.predict(X_test)  # Predicting test data labels
accuracy = metrics.accuracy_score(y_test, y_pred) * 100  # Calculate accuracy
num_train_samples = len(y_train)
num_test_samples = len(y_test)
num_correct_predictions = (y_pred == y_test).sum()
```

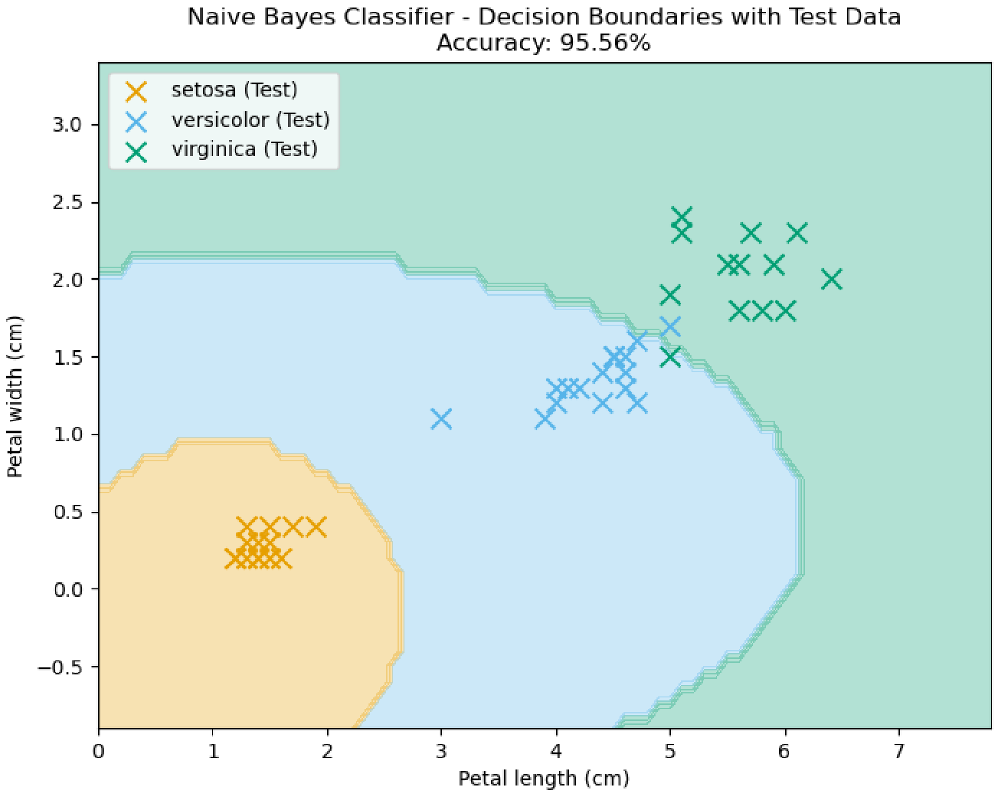
- Number of training samples: 105
- Number of test samples: 45
- Number of correct predictions: 43
- Accuracy: 95.56%

Example – Iris Flower Dataset





Example – Iris Flower Dataset



Example – Spam Email Detection

```
[ ] # Load the dataset (replace 'email.csv' with the correct path if necessary)
    emails_df = pd.read_csv('email.csv')
```

```
[ ] # Preprocessing the data
    # Convert 'Category' column to binary labels (ham = 0, spam = 1)
    emails_df['Category'] = emails_df['Category'].map({'ham': 0, 'spam': 1})
```

```
[ ] # Drop rows with missing values
    emails_df_clean = emails_df.dropna()
```

```
[ ] emails_df_clean.head(5)
```

	Category	Message
0	0.0	Go until jurong point, crazy.. Available only ...
1	0.0	Ok lar... Joking wif u oni...
2	1.0	Free entry in 2 a wkly comp to win FA Cup fina...
3	0.0	U dun say so early hor... U c already then say...
4	0.0	Nah I don't think he goes to usf, he lives aro...

Example – Spam Email Detection

```
[9] # Feature extraction (convert the text into numerical data using Bag-of-Words)
    vectorizer = CountVectorizer()
    X = vectorizer.fit_transform(emails_df_clean['Message'])  # Feature matrix
    y = emails_df_clean['Category']  # Labels
```

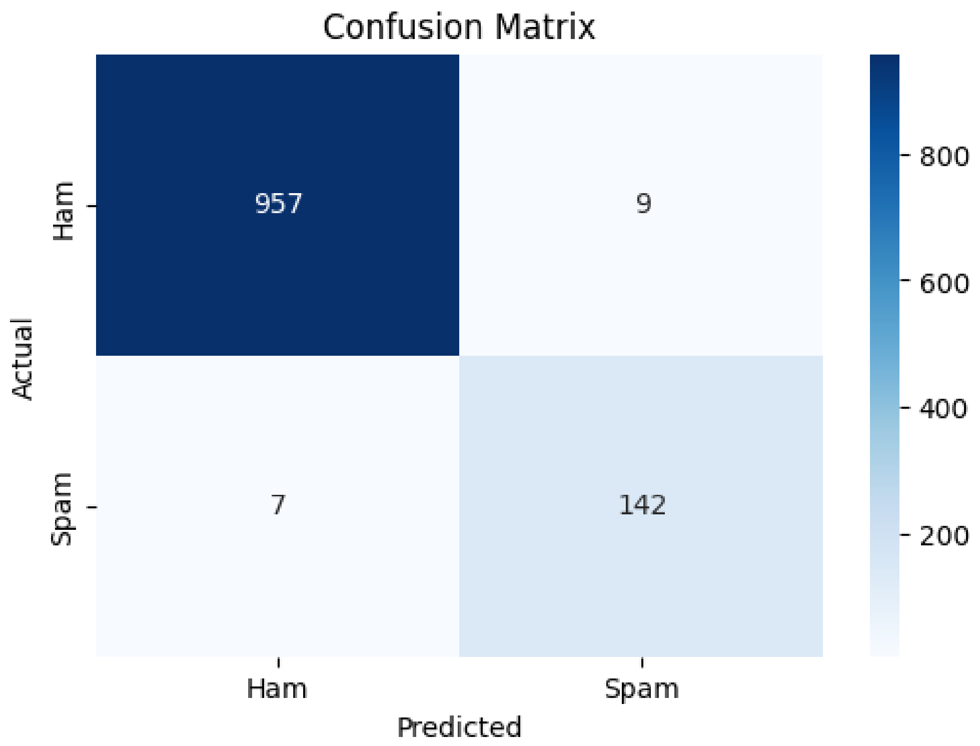
```
[10] # Split the data into training and testing sets (80% train, 20% test)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[11] # Train the Naive Bayes classifier
     nb_classifier = MultinomialNB()
     nb_classifier.fit(X_train, y_train)
```

```
MultinomialNB
MultinomialNB()
```

- Number of training samples: 4458
- Number of test samples: 1114

Example – Spam Email Detection




```
Accuracy: 0.9857

Classification Report:
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       966
         1.0       0.94      0.95      0.95       149

    accuracy                           0.99      1115
   macro avg       0.97      0.97      0.97      1115
weighted avg       0.99      0.99      0.99      1115
```

Example – Spam Email Detection

```
[17] # Test with a new email
     new_email = ["We would like to congratulate you for your new position as our PhD Candidate at NTNU."]
     new_email_transformed = vectorizer.transform(new_email)
     prediction = nb_classifier.predict(new_email_transformed)

     print(f"\nNew email prediction: {'Spam' if prediction[0] == 1 else 'Not Spam'}")
```

```
New email prediction: Not Spam
```

```
[18] # Test with a new email
     new_email = ["Congratulation! You just won a million US dollar, click here for claim."]
     new_email_transformed = vectorizer.transform(new_email)
     prediction = nb_classifier.predict(new_email_transformed)

     print(f"\nNew email prediction: {'Spam' if prediction[0] == 1 else 'Not Spam'}")
```

```
New email prediction: Spam
```

Common cases:

- Sentiment analysis
- Credit scoring risk prediction

Pros and Cons

Pros:

- Fast on high dim. data
- Good performance on limited training data

Cons:

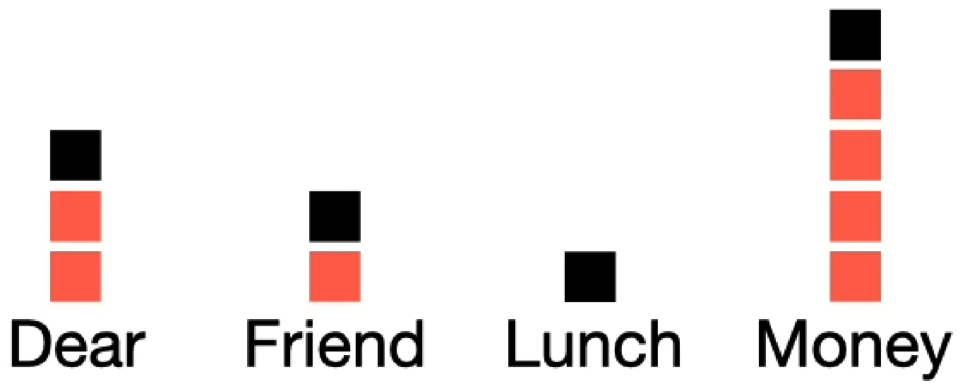
- Feature independence
- Diff. Problems with diff. versions

What Not To Do

- Not eliminating correlated features
  - Ignoring independence assumption
  - Result in overestimation/inaccurate classification
  - Solution: DR and feature selection

What Not To Do

- Zero frequency problem
  - Class that does not exist in training data
  - Solution: additive smoothing



What Not To Do

- Imbalanced classes
  - Solution: resampling or adjusting class weights

Sources

- https://www.ibm.com/topics/naive-bayes