PART 1

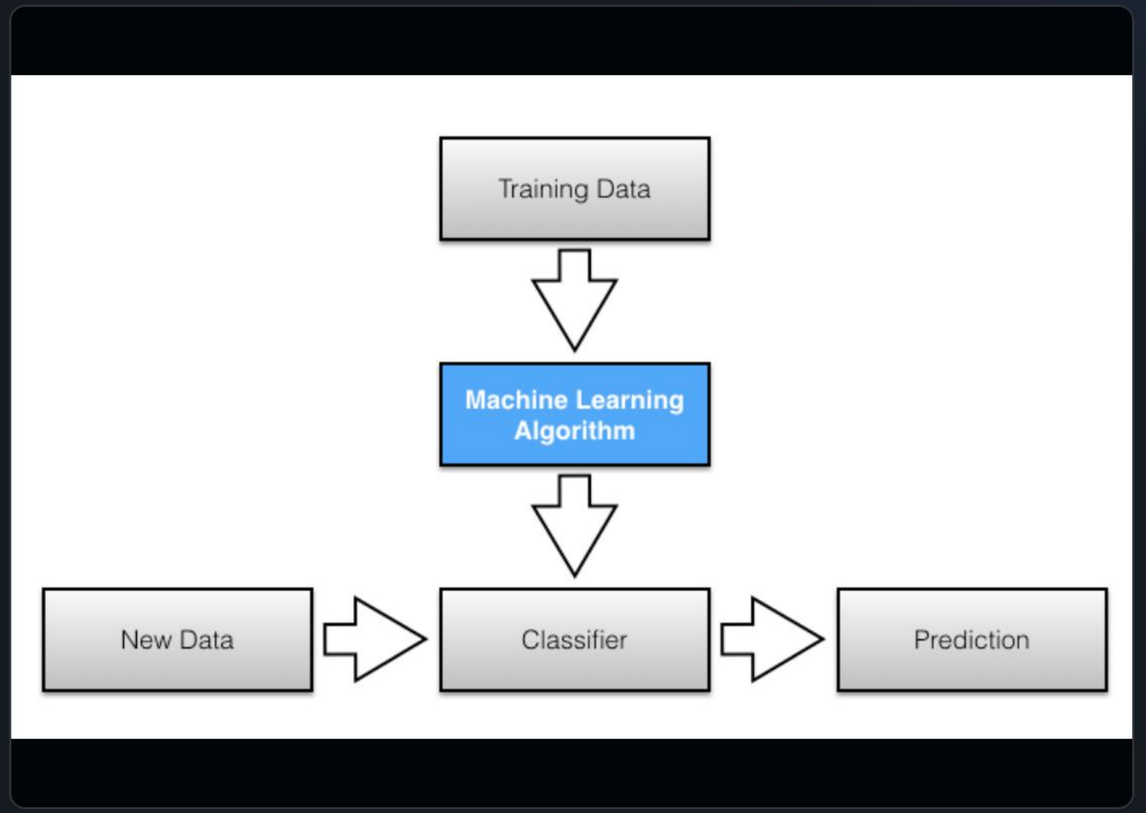
Naive Bayes Classifiers

Probabilistic Machine Learning based on Bayes' Theorem

What is Naive Bayes?

Naive Bayes is a family of probabilistic algorithms that takes advantage of probability theory and Bayes' Theorem to predict the class of a sample.

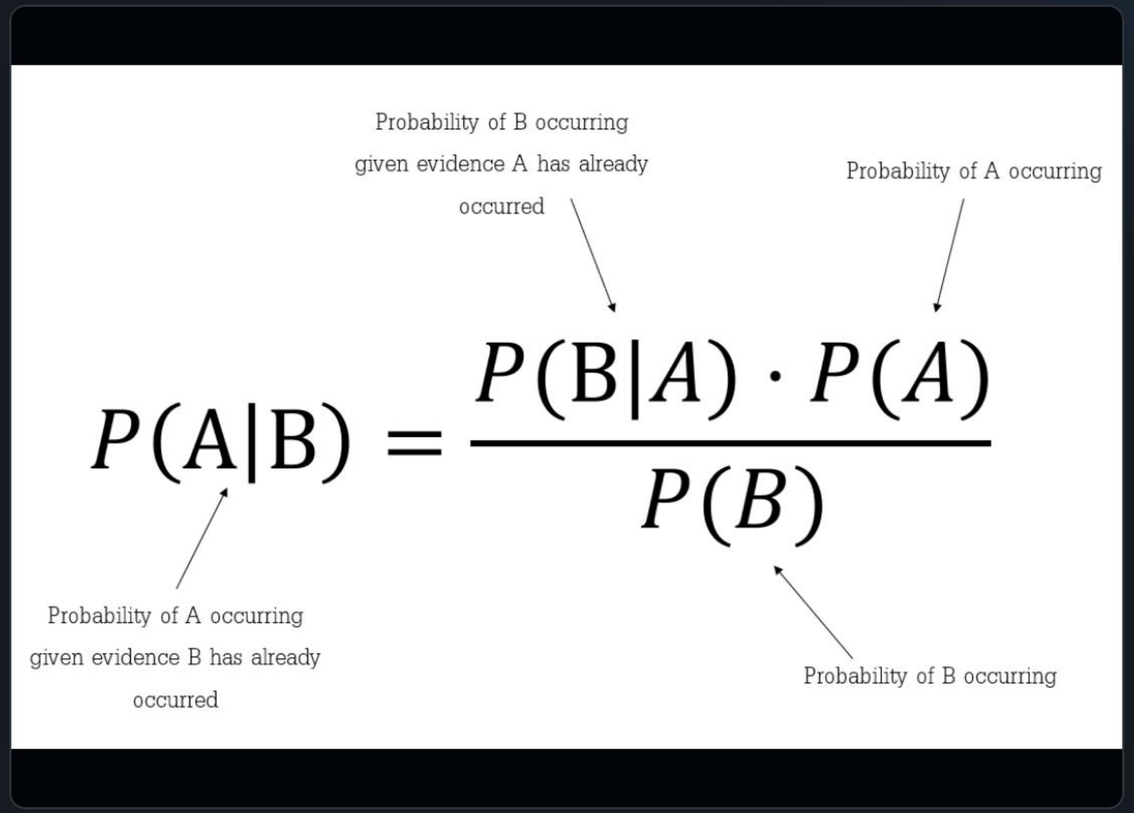
- **Probabilistic:** Calculates the probability of each class.
- **Fast:** Extremely efficient for large datasets.
- **"Naive":** Assumes features are independent.



Bayes' Theorem

The foundation of the algorithm. It describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

```
P(A|B) = [P(B|A) * P(A)] / P(B)
```



The "Naive" Assumption

Feature Independence

It assumes that the presence of a particular feature in a class is **unrelated** to the presence of any other feature.

Example: A fruit may be considered an apple if it is red, round, and about 3 inches in diameter. Naive Bayes considers each of these features to contribute independently to the probability.

In reality, features are often correlated, but this simplification makes the computation incredibly fast.

Types of Naive Bayes

Gaussian

Used when features are continuous and follow a normal distribution (e.g., Iris dataset).

Multinomial

Used for discrete counts. Very popular in text classification (e.g., word counts in emails).

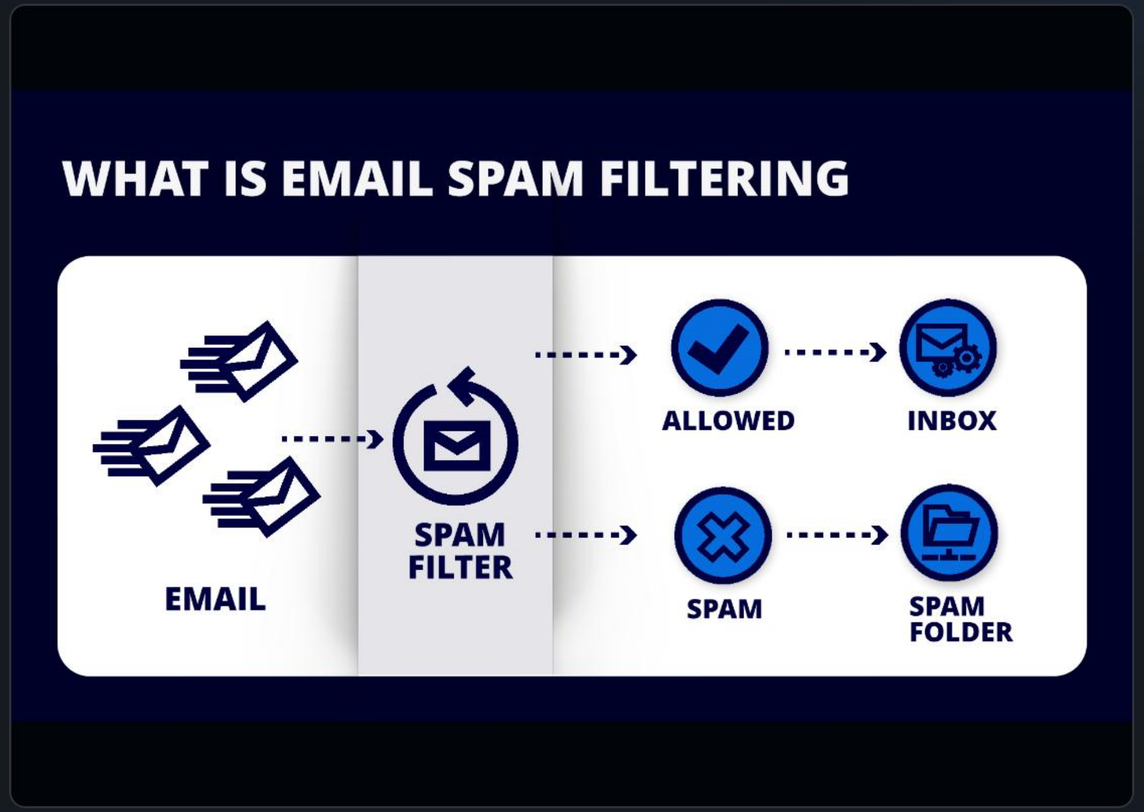
Bernoulli

Used for binary/boolean features. Checks if a word appears or not (True/False).

Application: Spam Filtering

One of the most famous use cases. The classifier calculates the probability a message is spam given the words it contains.

- **Features:** Words like "Free", "Winner", "Credit".
- **Training:** Learn probability of words appearing in Spam vs Ham.
- **Prediction:** Multiply probabilities of all words in the new email.



Pros & Cons

👍 Advantages

- Extremely fast and scalable.
- Performs well with high-dimensional data (text).
- Needs less training data than other models.

👎 Disadvantages

- Assumption of independence is often violated.
- "Zero Frequency" problem: If a categorical variable has a category in test data not seen in training, probability becomes 0.

Python Implementation: Setup

We use `sklearn.naive_bayes`. In this example, we generate a synthetic dataset and split it into training and testing sets.

GaussianNB is chosen for continuous features.

```
from sklearn.datasets import make_classification
from sklearn.model_selection import
train_test_split from sklearn.naive_bayes import
GaussianNB # 1. Generate Data X, y =
make_classification( n_samples=1000, n_features=20,
random_state=42 ) # 2. Split Data X_train, X_test,
y_train, y_test = train_test_split( X, y,
test_size=0.3, random_state=42 )
```

Python Implementation: Training

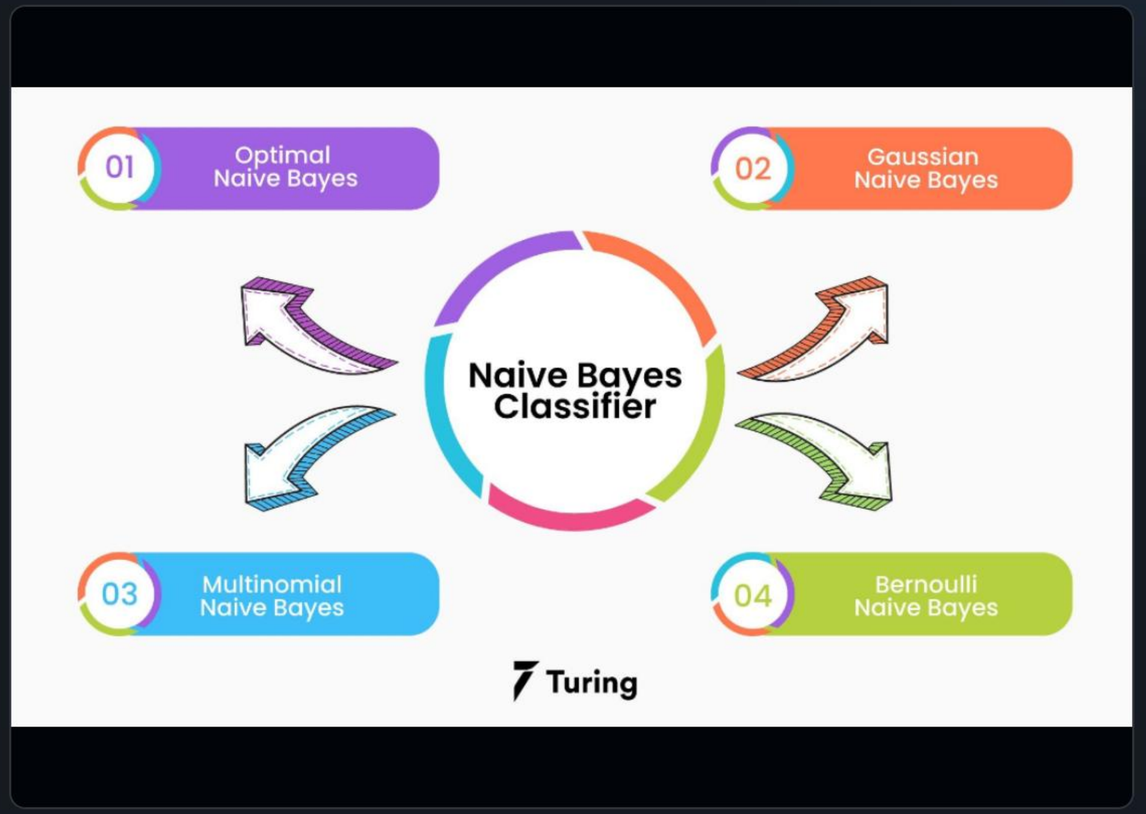
```
# 3. Initialize Model model = GaussianNB() # 4. Train Model model.fit(X_train, y_train) # 5. Make Predictions y_pred = model.predict(X_test) # 6. Evaluate from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc:.2f}")
```

Key Steps

1. **Initialize:** Create an instance of the classifier.
2. **Fit:** Train the model using the training vectors and labels.
3. **Predict:** Perform classification on an array of test vectors.

Summary: Naive Bayes

- A powerful baseline for text classification.
- Relies on the independence assumption.
- Variants exist for different data types (Gaussian, Multinomial, Bernoulli).



PART 2

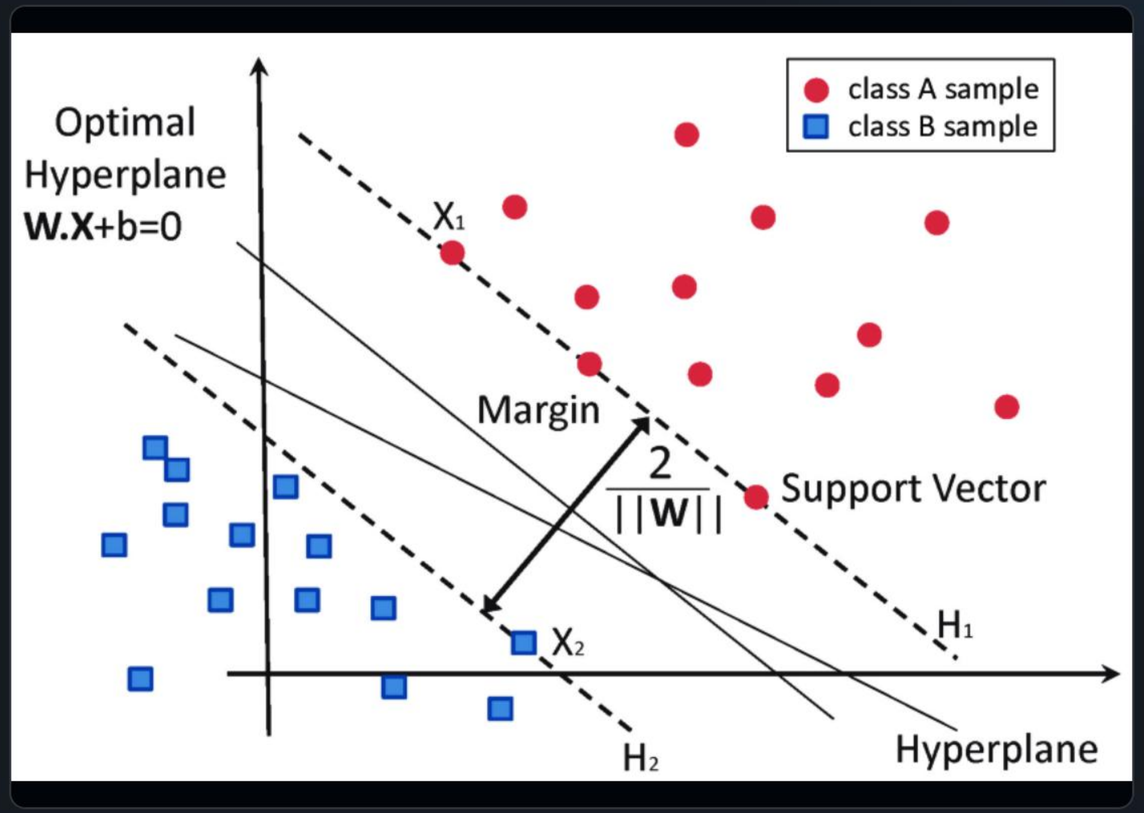
Support Vector Machines

Finding the Optimal Hyperplane for Classification

What is SVM?

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression.

Goal: Find a hyperplane in N-dimensional space that distinctly classifies the data points.



Key Terminology

Hyperplane

The decision boundary that separates different classes. In 2D it's a line, in 3D it's a plane.

Margin

The distance between the hyperplane and the nearest data point from either class. We want to **maximize** this.

Support Vectors

The data points closest to the hyperplane. These points influence the position of the hyperplane.

Hard vs Soft Margin

Hard Margin

Strictly forbids any misclassification. Only works if data is linearly separable and noise-free.

Soft Margin

Allows some misclassification to find a better general boundary.

C Parameter: Controls the trade-off.

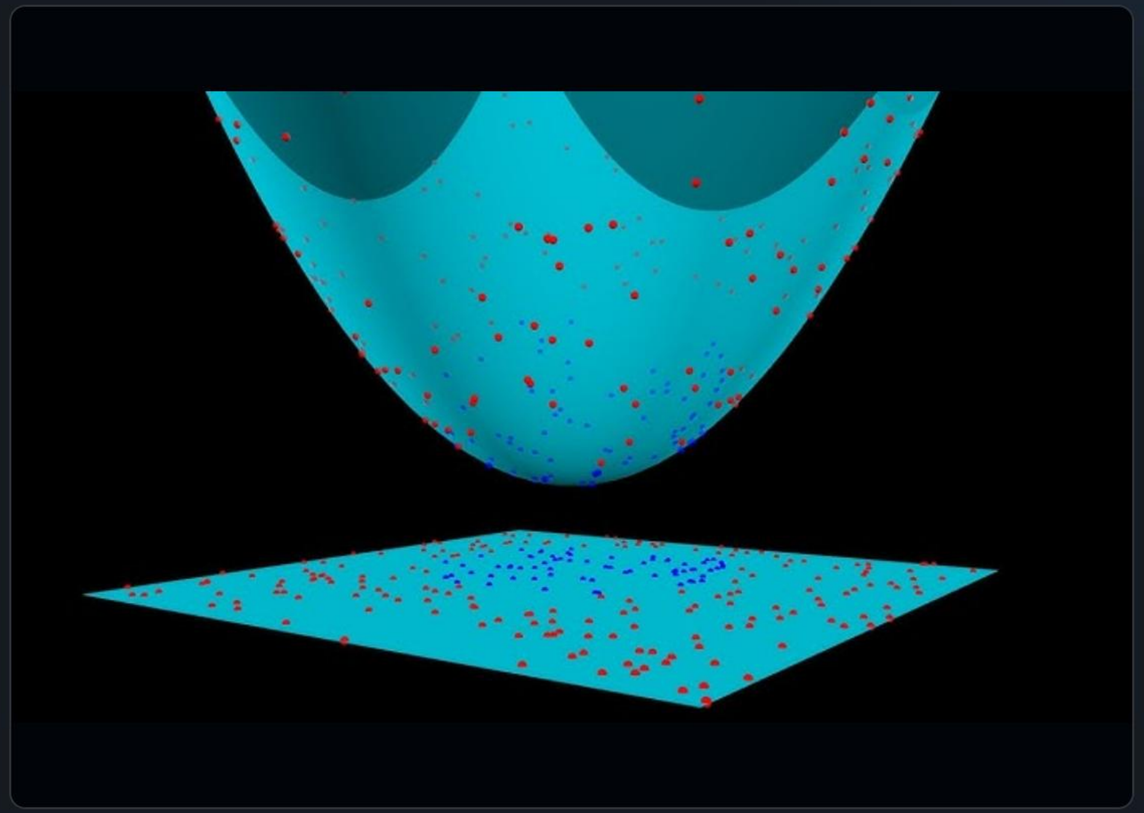
- **High C:** Strict (Hard Margin).
- **Low C:** Loose (Soft Margin).

The Kernel Trick

What if data is not linearly separable?

SVM projects the data into a **higher-dimensional space** where it becomes separable.

This is computationally cheap because of the "Kernel Trick"—we don't actually calculate the coordinates in the high-dimensional space.



Types of Kernels

Linear

Useful for linearly separable data. Very fast.

Polynomial

Represents similarity of vectors in a degree *d* polynomials.

RBF (Radial Basis)

The default kernel. Maps to infinite dimensions. Great for complex boundaries.

Pros & Cons of SVM

👍 Advantages

- Effective in high dimensional spaces.
- Memory efficient (uses subset of training points).
- Versatile (different Kernel functions).

👎 Disadvantages

- Not suitable for large datasets (slow training).
- Sensitive to noise and outliers (if C is high).
- Does not provide probability estimates directly.

Python Implementation: Setup

We use `sklearn.svm.SVC` (Support Vector Classification).

We can specify the kernel type (linear, rbf, poly) during initialization.

```
from sklearn import datasets from
sklearn.model_selection import train_test_split
from sklearn.svm import SVC # 1. Load Data (Iris
Dataset) iris = datasets.load_iris() X = iris.data
y = iris.target # 2. Split Data X_train, X_test,
y_train, y_test = train_test_split( X, y,
test_size=0.3, random_state=42 )
```

Python Implementation: Training

```
# 3. Initialize Model (Linear Kernel) model = SVC(kernel='linear', C=1.0) # 4. Train Model model.fit(X_train, y_train) # 5. Predict y_pred = model.predict(X_test) # 6. Evaluate from sklearn.metrics import accuracy_score acc = accuracy_score(y_test, y_pred) print(f"Accuracy: {acc:.2f}")
```

Parameters

- **kernel:** 'linear', 'poly', 'rbf', 'sigmoid'.
- **C:** Regularization parameter. Strictness of the margin.
- **gamma:** Kernel coefficient for 'rbf', 'poly', 'sigmoid'.

Visualizing Non-Linearity

When using the **RBF Kernel**, SVM can capture complex, non-linear boundaries like circles or curves, effectively separating classes that a straight line cannot.

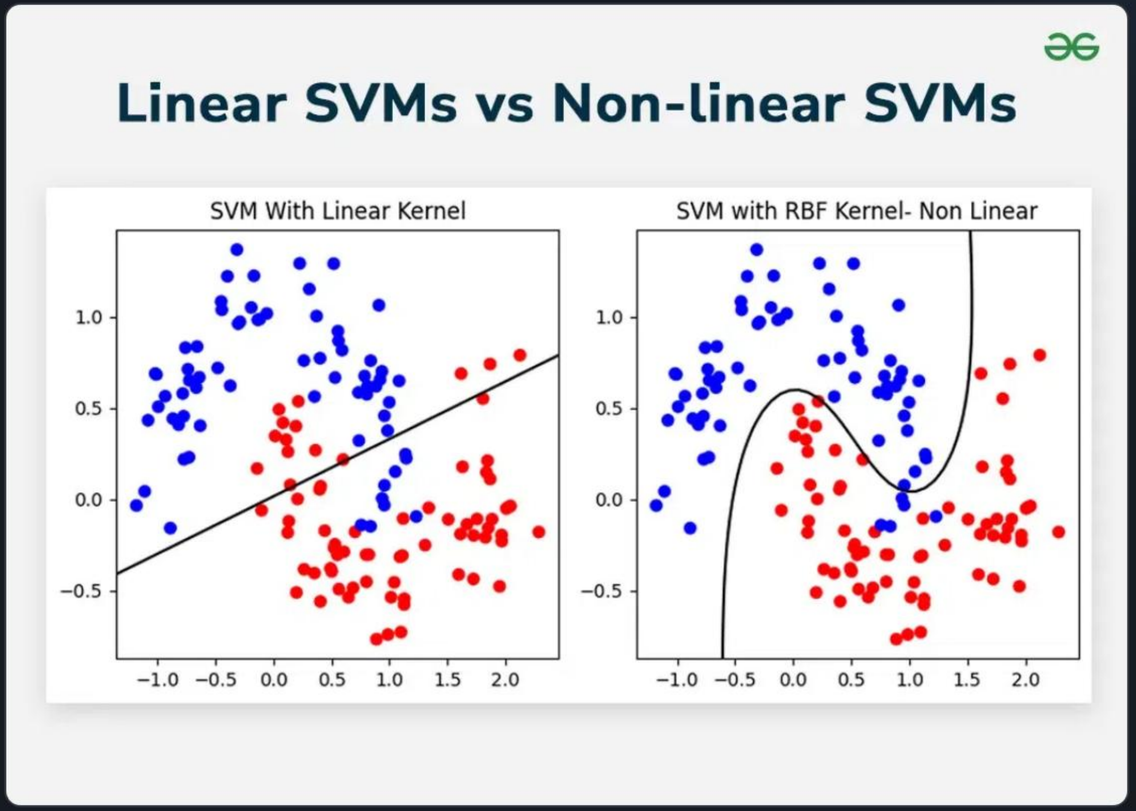
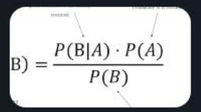


Image Sources

https://sebastianraschka.com/images/blog/2014/naive_bayes_1/learning_algorithm_1.png

Source: sebastianraschka.com



https://thestatsninja.com/wp-content/uploads/2019/03/bayes-1.png

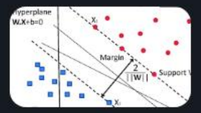
Source: thestatsninja.com

https://assets.mimecast.com/api/public/content/what-is-email-spam-filtering?v=5d0ded05

Source: www.mimecast.com

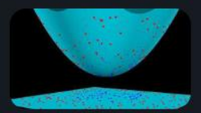
https://miro.medium.com/0*JdB13cyAJ3EESnUu

Source: code.likeagirl.io



https://miro.medium.com/v2/resize:fit:1400/0*5EsKRZqZuZEplh92.png

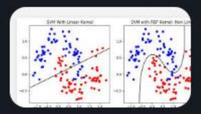
Source: medium.com



https://i.ytimg.com/vi/3liCbRZPrZA/hq720.jpg?sqp=-oaymwEhCK4FEIIDSFryq4qpAxMIARUAAAAAGAElAADIQj0AgKJD&rs=AOn4CLBRtPUmbWTyDqO-s4RJ_yasaHmOFQ

Source: www.youtube.com

Image Sources



https://media.geeksforgeeks.org/wp-content/uploads/20240628112552/Linear-vs-Non-linear-SVMs.webp

Source: www.geeksforgeeks.org