Naive Bayes & SVM – Final Cheat Sheet

# 📘 Naive Bayes (NB) – Summary

Naive Bayes is a probabilistic classifier based on Bayes’ Theorem. It assumes feature independence and calculates the posterior probability for each class.

## Key Concepts:

- Bayes' Theorem: P(Class|Data) ∝ P(Data|Class) \* P(Class)  
- Assumes features are conditionally independent  
- Fast and effective, especially for text data

## Types of Naive Bayes:

1. GaussianNB – For continuous features (e.g., Breast Cancer dataset)  
2. MultinomialNB – For count/frequency data (e.g., SMS Spam with CountVectorizer)  
3. BernoulliNB – For binary features (e.g., word presence/absence)

## Tips & Tricks:

- Use Laplace Smoothing to handle zero probabilities  
- CountVectorizer works best with MultinomialNB  
- TF-IDF may hurt recall in some NB models  
- Works well as a fast baseline classifier

# ⚔️ Support Vector Machines (SVM) – Summary

SVMs are supervised classifiers that find the hyperplane that maximizes the margin between classes. They work well with both linear and non-linear data using kernel functions.

## Key Concepts:

- Support Vectors: Points closest to the decision boundary  
- Margin: Distance between support vectors and hyperplane  
- Kernel Trick: Transforms data into higher dimensions implicitly

## Common Kernels:

1. Linear – For linearly separable data  
2. Polynomial – Adds non-linearity through polynomial features  
3. RBF – Radial Basis Function (default); handles complex patterns well  
4. Sigmoid – Mimics neural networks; rarely used

## Hyperparameters:

- C: Regularization (low = soft margin, high = hard margin)  
- gamma: Influences how far a single training example affects the decision boundary  
- degree: Only for polynomial kernel

## Tuning with GridSearchCV:

- Use GridSearchCV for parameter tuning  
- Evaluate using cross-validation accuracy and classification reports  
- Normalize or scale features before fitting

# 💡 Final Tips

- Naive Bayes is excellent for text and fast predictions.  
- SVM is more flexible, especially with kernels, but slower on large datasets.  
- Always visualize results using confusion matrices and evaluate class-wise precision/recall.  
- Document your kernel choice and tuning parameters.