UnCodemy

# KNN, Naive Bayes and SVM PDF

K-nearest neighbours classifies by the labels of the closest training points, Naive Bayes applies Bayes' theorem with an independence assumption, and a support vector machine finds the boundary with the widest margin. KNN and SVM need scaled features, Naive Bayes is fast for text, and each has clear strengths and limits.

## Key takeaways

- KNN stores the data and predicts by voting among the k nearest points.
- Naive Bayes is fast and works well on text despite its simple independence assumption.
- SVM seeks the widest margin and can use kernels for curved boundaries.
- KNN and SVM are sensitive to feature scale, so standardise inputs.
- Compare all candidates on the same split with the same metric.

## What is KNN, Naive Bayes and SVM classifiers?

KNN, Naive Bayes and support vector machines are three supervised classification algorithms that decide a class using neighbour voting, probability estimates and a maximum-margin boundary respectively.

## How does k-nearest neighbours work?

KNN keeps all training examples. To classify a new point, it measures distance to every stored point, picks the **k** closest and returns the most common class among them. A small k gives a jagged boundary that can overfit; a large k smooths it but can blur classes. Prediction is slow on big data because distances are computed at query time.

## When is KNN a good choice?

Use it as a simple baseline on small or medium datasets with a few meaningful numeric features. **Scale features**, since a column measured in thousands will dominate one measured in fractions. Avoid it when there are many irrelevant features or very large data.

## How does Naive Bayes work?

It uses Bayes' theorem to compute the probability of each class given the features, and assumes features are independent given the class. The assumption is rarely true, yet the method often works well, especially for text such as spam filtering, because it needs little data and trains in a single pass. Variants include Gaussian for continuous values and Multinomial for counts. A technique called smoothing prevents zero probabilities for unseen words.

## How does a support vector machine work?

An SVM looks for the boundary that separates classes with the largest margin. The points nearest the boundary are the **support vectors**. A **soft margin**, controlled by the parameter C, allows some mistakes so the model can generalise. With **kernels** such as the radial basis function, an SVM can separate classes that are not linearly separable by working in a transformed space.

## How do the three compare?

| Aspect | KNN | Naive Bayes | SVM |
|---|---|---|---|
| Training speed | Almost none | Very fast | Slower on large data |
| Prediction speed | Slow on big data | Fast | Fast |
| Needs scaling | Yes | Not for counts | Yes |
| Handles text | Poorly | Well | Well with linear kernels |
| Main parameter | k | Smoothing | C and kernel |
| Explainability | Easy to describe | Probabilities | Harder with kernels |

## Which parameters matter most?

For KNN, tune k and the distance type. For Naive Bayes, tune the smoothing value. For SVM, tune C and, for kernels, the gamma value. Use cross-validation and a grid or random search, and fit scalers inside a pipeline to avoid leakage.

## What does a minimal comparison look like?

```
models = {"knn": KNeighborsClassifier(5), "nb": GaussianNB(), "svm": SVC(C=1.0)}
for name, m in models.items():
    pipe = make_pipeline(StandardScaler(), m)
    print(name, cross_val_score(pipe, X, y, cv=5).mean())
```

Print mean scores side by side, then inspect precision and recall for the best candidate.

## What mistakes are common?

Forgetting to scale, tuning on the test set, using accuracy on imbalanced classes, choosing a kernel without trying a linear baseline and applying Gaussian Naive Bayes to count data. Keep a record of settings and scores.

## How do I choose among them for a new problem?

Begin with the data. For short texts or word counts, try Naive Bayes first because it is fast and strong. For a small table with a few numeric features, try KNN as a baseline and an SVM for a stronger boundary. For many features with limited samples, a linear SVM often performs well.

Whatever you pick, compare against a simple baseline such as logistic regression and measure on the same cross-validation folds so that the comparison is fair.

## How do these methods handle imbalanced classes?

When one class is rare, plain accuracy can look excellent even if the model never predicts that class. Use class weights in the SVM, adjust prior probabilities in Naive Bayes or change the decision threshold. For KNN, weighting neighbours by distance can help a little. Always report precision, recall and the confusion matrix, and consider resampling carefully inside cross-validation so that no duplicated rows leak between the training and validation folds.

## How do I explain these models to a manager?

For KNN say that the model looks at the most similar past cases and follows the majority. For Naive Bayes say that it adds up evidence from each feature, like a checklist of clues. For SVM say that it draws the cleanest possible dividing line between two groups. Follow each explanation with the main limit, such as slow lookups for KNN, simple assumptions for Naive Bayes and difficulty explaining kernels for SVM, so that decisions about which model to use are made with honest expectations.

## Frequently asked questions

### How does KNN decide the class of a new point?

It finds the k closest training points by distance and assigns the class that appears most often among those neighbours.

### Why is Naive Bayes called naive?

It assumes features are independent given the class, which is rarely true in practice but often still gives useful results.

### What is the margin in an SVM?

The margin is the gap between the decision boundary and the nearest points of each class, and an SVM tries to make it as wide as possible.

### Do these algorithms need feature scaling?

KNN and SVM do, because they depend on distances, while Naive Bayes on counts or categories usually does not.

### Which classifier is best for text?

Naive Bayes and linear SVMs are common strong baselines for text classification, but you should compare them with cross-validation.

## Sources and further reading

- scikit-learn user guide
- scikit-learn supervised learning

## About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550