Spam detection with Logistic Regression, Random Forest, and Naive Bayes

What I implemented

I built a text-classification pipeline to detect spam vs. ham. The steps were:

1. Load mail.csv and map labels: spam → 0, ham → 1.
2. Split messages into train/test sets (20% test).
3. Convert text to numeric features using TfidfVectorizer.
4. Train three classifiers on the TF-IDF features: Logistic Regression (LR), Random Forest (RF), and Multinomial Naive Bayes (NB).
5. Evaluate each model with Accuracy, Precision, Recall, F1 (treating **spam = 0** as the "positive" class) and show a confusion matrix.

How each model was used (brief)

- **Logistic Regression**: linear model trained on TF-IDF features; outputs class probabilities used with the default 0.5 threshold.
- **Random Forest**: ensemble of decision trees (200 estimators) trained on the same TF-IDF features; votes produce the class prediction.
- **Multinomial Naive Bayes**: probabilistic model well-suited to word-count/TF-IDF features; uses Bayes rule with Laplace smoothing (default) and the conditional independence assumption.

All models were trained on x_train_features and evaluated on x_test_features. The positive class for metric functions was set to 0 (spam), so precision/recall/F1 reported are for detecting spam.

Results (summary table)

Model	Accuracy	Precision (spam)	Recall (spam)	F1 (spam)	TP_spam	FN_spam
Logistic Regression	0.968	1.000	0.758	0.863	113	36
Random Forest	0.983	1.000	0.872	0.932	130	19
Naive Bayes	0.977	1.000	0.826	0.904	123	26

Notes:

- Test set totals: Actual ham = 966, Actual spam = 149, Total = 1,115.
- Precision = 1.000 for all models because **false positives = 0** (no ham was labeled as spam).

- False negatives (spam missed) are 36 (LR), 19 (RF), and 26 (NB).

Comparison of the three sanity-check messages

You ran three sample messages (sanity checks). All three models agreed on all of them:

1. A clear spam message → all predicted **spam**.
2. A reflective personal message → all predicted **ham**.
3. A casual message about a ring → all predicted **ham**.

Because the three samples are easy/clear examples, agreement is expected. These sanity-checks confirm that for prototypical spam/ham the models behave consistently. The difference in model performance appears on the harder boundary cases, reflected in the differing false-negative counts above.

Understanding Naive Bayes:

What is Naive Bayes?
Naive Bayes is a family of probabilistic classifiers based on Bayes' theorem. Multinomial Naive Bayes models the probability of words given a class and assumes features (words) are conditionally independent given the class (the "naive" assumption).

Why often used for spam detection?

- Text classification naturally fits the probabilistic word-count model of Multinomial NB.
- It is fast to train and predict, scales well to high-dimensional sparse vectors (TF-IDF), and often performs surprisingly well even when the independence assumption is not strictly true.

Advantages

- Very fast to train and apply (low compute/memory).
- Works well with high-dimensional sparse data (text).
- Requires few hyperparameters and is robust on small datasets.

Limitations

- The conditional independence assumption is unrealistic for language (words interact/correlate).
- Probabilities can be poorly calibrated — the output probabilities are not always reliable as confidence scores.
- Performance can lag behind ensemble or discriminative models when there are strong feature interactions (which tree models or logistic regression can capture).

Metrics discussion and confusion-matrix interpretation

- **Accuracy**: RF (98.3%) > NB (97.7%) > LR (96.8%).
- **Precision**: all models = 100% because FP = 0 (no ham was classified as spam).
- **Recall (sensitivity for spam)**: RF (87.2%) > NB (82.6%) > LR (75.8%). Higher recall means fewer spam emails slip into the inbox.
- **F1 (harmonic mean)**: RF (0.932) > NB (0.904) > LR (0.863). F1 combines precision and recall; since precision = 1.0, F1 mainly reflects recall differences here.

Confusion matrix meaning (for these results):

- **False positives (FP)** = ham labeled spam = 0 across all models. Consequence: no legitimate email was blocked—good for avoiding lost/blocked messages.
- **False negatives (FN)** = spam labeled ham = 36 (LR), 19 (RF), 26 (NB). Consequence: these spam messages reach the inbox (annoying and potentially harmful).

Given the dataset and current thresholds, the models are conservative: they avoid false alarms (FP = 0) at the cost of missing some spam (FN > 0). That trade-off (precision vs recall) is a design choice.

My Findings and My recommendation

- **Best raw performance**: Random Forest produced the best overall metrics (highest accuracy, recall, and F1) and the fewest missed spam (19/149). If your priority is *catching as much spam as possible* while still avoiding false positives, RF is the best choice among the three.
- **Operational considerations**: If you need very fast inference, low memory and simple deployment, **Naive Bayes** is attractive (good recall and F1, and very fast). **Logistic Regression** is simple and interpretable but here had the lowest recall.
- **Caveats to check**: FP = 0 for all models is suspiciously perfect; verify there is no data leakage, label leakage, or non-random split. Also evaluate models using cross-validation, and with additional held-out data from a different time period to check generalization.

Final recommendation: Use **Random Forest** if model size and inference cost are acceptable — it gives the best detection rate in your tests. If you require a lightweight model, prefer **Multinomial Naive Bayes** or **Logistic Regression**, but consider tuning thresholds or combining models (ensemble/voting) to balance recall vs precision for your operational needs.