

SMS Spam Classification - Final Report

Project Overview

This project implements and compares two machine learning approaches for SMS spam detection:

1. **Artificial Neural Network (ANN)** using TensorFlow/Keras
2. **Naive Bayes classifier** as a baseline

Summary of Approach

Data Pipeline

- **Dataset:** SMS Spam Collection with 5,574 messages (4,827 ham, 747 spam - 13.4% spam rate)
- **Preprocessing:**
 - Text cleaning (lowercase, special character removal)
 - TF-IDF vectorization (5,000 features, unigrams + bigrams)
 - Train/test split: 80/20 with stratification (random_state=42)
 - Final split: 4,454 training samples, 1,114 test samples

Model Architectures

1. Artificial Neural Network (ANN)

- **Architecture:**
 - Input layer: 5,000 features (TF-IDF dimensions)
 - Hidden layer 1: 128 neurons with ReLU activation + Dropout (0.5)
 - Hidden layer 2: 64 neurons with ReLU activation + Dropout (0.5)
 - Output layer: 1 neuron with sigmoid activation
- **Training Configuration:**
 - Optimizer: Adam
 - Loss: Binary crossentropy
 - Batch size: 32, Epochs: 20
 - Validation split: 20% of training data

- Total parameters: 648,449 (2.47 MB)

2. Naive Bayes Classifier

- **Algorithm:** Multinomial Naive Bayes
- **Hyperparameters:** Alpha (Laplace smoothing) = 1.0
- **Training:** Single-pass probabilistic approach (< 1 second training time)

Key Results

Performance Comparison Table

Metric	ANN	Naive Bayes	Difference	Winner
Accuracy	97.49%	97.31%	+0.18%	ANN
Precision	92.86%	100.00%	-7.14%	Naive Bayes
Recall	87.25%	79.87%	+7.38%	ANN
F1-Score	89.97%	88.81%	+1.16%	ANN
Error Rate	2.60%	2.69%	-0.09%	ANN

Confusion Matrix Analysis

ANN Confusion Matrix

	Predicted Ham	Predicted Spam
Actual Ham (965)	955 (TN)	10 (FP)
Actual Spam (149)	19 (FN)	130 (TP)

- **Error Rate:** 2.60% (29 misclassifications / 1,114 samples)
- **False Positive Rate:** 1.04% (10 ham marked as spam)
- **False Negative Rate:** 12.75% (19 spam marked as ham)

Naive Bayes Confusion Matrix

	Predicted Ham	Predicted Spam
Actual Ham (965)	965 (TN)	0 (FP)
Actual Spam (149)	30 (FN)	119 (TP)

- **Error Rate:** 2.69% (30 misclassifications / 1,114 samples)
- **False Positive Rate:** 0.00% (perfect precision - zero ham marked as spam)
- **False Negative Rate:** 20.13% (30 spam marked as ham)

Training Dynamics (ANN Only)

- **Initial Performance** (Epoch 1): Train acc: 85.85%, Val acc: 89.67%
- **Final Performance** (Epoch 20): Train acc: 99.97%, Val acc: 96.63%
- **Loss Progression:** Train loss: 0.0013, Val loss: 0.1936
- **Observation:** Model shows slight overfitting but generalizes well to unseen test data

Comparative Analysis: Why Metrics Change

1. Precision: Naive Bayes Dominates (100% vs 92.86%)

Why Naive Bayes Wins:

- **Zero false positives** - never misclassifies legitimate messages as spam
- Probabilistic thresholds favor conservative predictions (higher confidence required for spam label)
- Conditional independence assumption creates natural decision boundaries that avoid false alarms
- **Trade-off:** Sacrifices recall to maintain perfect precision

Why ANN Loses:

- Neural networks optimize for overall loss, not specifically precision
- 10 false positives indicate aggressive spam detection (lower confidence threshold)
- Complex non-linear boundaries can overfit to training patterns

Business Impact: Naive Bayes is ideal when false positives are costly (e.g., missing important messages).

2. Recall: ANN Wins (87.25% vs 79.87%)

Why ANN Wins:

- **Catches 7.38% more spam** (11 additional spam messages detected)
- Deep architecture with 2 hidden layers learns complex non-linear feature interactions
- Dropout regularization prevents memorization, enabling better generalization to novel spam patterns
- Backpropagation optimizes feature representations across layers

Why Naive Bayes Loses:

- Strong independence assumption: treats each word independently, ignoring word order and context
- Cannot capture interactions like "not good" vs "good" (negation patterns)
- Misses 30 spam messages due to lack of contextual understanding

Business Impact: ANN is better at catching sophisticated, evolving spam tactics.

3. F1-Score: ANN Wins (89.97% vs 88.81%)

Why ANN Wins:

- F1-score is the harmonic mean of precision and recall ($2 \times P \times R / (P + R)$)
- ANN achieves better **balance** between precision (92.86%) and recall (87.25%)
- The 10 false positives are offset by catching 11 more spam messages (net gain of 1 error)
- **Better overall trade-off** for real-world deployment

Why Naive Bayes Loses:

- Perfect precision (100%) cannot compensate for significantly lower recall (79.87%)
- Harmonic mean penalizes imbalanced performance more than arithmetic mean
- Missing 30 spam messages outweighs the benefit of zero false positives

4. Accuracy: Near Tie (97.49% vs 97.31%)

Why Similar:

- Both models handle class imbalance well (86.6% ham dominates dataset)

- High accuracy is driven by strong true negative rate (correctly identifying ham)
- Small difference (0.18%) is **not statistically significant**
- Accuracy is less informative for imbalanced datasets (can achieve 86.6% by always predicting ham)

Key Insight: Accuracy alone is misleading - precision, recall, and F1-score provide better model comparison.

Misclassification Insights (ANN)

False Positives (Ham → Spam) - 10 cases

Examples:

- "glad see reply" (confidence: 94.96%)
- "pic please resend" (confidence: 87.95%)
- "u receive msg" (confidence: 98.47%)

Root Causes:

- **Short imperative commands** resemble spam CTAs ("reply now", "text back")
- **Informal abbreviations** ("u", "ur") common in spam messages
- **Action-oriented language** triggers spam patterns learned during training

Potential Fixes:

- Add message length as feature (spam typically longer)
- Incorporate sender trust scores
- Use contextual embeddings (BERT) to distinguish legitimate short messages

False Negatives (Spam → Ham) - 19 cases

Examples:

- "sale arsenal dartboard good condition double treble" (confidence: 0.11%)
- "romcapspam everyone around responding well presence since warm outgoing bringing real breath sunshine" (confidence: 0.06%)

Root Causes:

- **Legitimate-sounding product descriptions** lack typical spam markers (FREE, WIN, URGENT, !!!)
- **Conversational tone** mimics natural language patterns
- **Sophisticated spam** uses subtle promotional language to evade detection

Potential Fixes:

- Add character-level n-grams to catch obfuscation ("fr33" for "free")
- Train on more recent spam datasets with modern evasion tactics
- Ensemble with rule-based filters for known spam patterns

Reflections

Limitations

1. Dataset Constraints

- Single domain (SMS) - poor generalization to email, social media
- Dated dataset - lacks modern spam tactics (emojis, unicode spoofing, zero-width characters)
- Class imbalance (13.4% spam) - metrics may be inflated

2. Feature Engineering

- TF-IDF loses semantic meaning ("not good" treated same as "good")
- Bag-of-words ignores word order and context
- No meta-features (message length, capitalization ratio, URL presence)

3. Model Limitations

- **ANN:** Overfitting evident (99.97% train vs 96.63% validation accuracy)
- **Naive Bayes:** Independence assumption unrealistic for natural language
- Both lack explainability (black-box predictions)

4. Evaluation Gaps

- No inference time comparison (ANN likely slower)
- Missing cost-sensitive analysis (false positive vs false negative business costs)
- No adversarial testing (spam designed to fool the model)

Potential Improvements

1. Advanced Architectures

- **LSTM/GRU**: Capture sequential dependencies and word order
- **Transformers (BERT, RoBERTa)**: Pre-trained embeddings for semantic understanding
- **Ensemble methods**: Combine ANN + Naive Bayes predictions (voting or stacking)

2. Feature Enhancement

- **Meta-features**: Message length, capitalization ratio, punctuation density, URL count
- **Word embeddings**: Word2Vec, GloVe, FastText for semantic similarity
- **Character n-grams**: Detect obfuscation techniques ("v1agra" for "viagra")

3. Training Strategies

- **Early stopping**: Monitor validation loss to prevent overfitting
- **Class weighting**: Penalize false negatives more (spam is costlier than false alarms)
- **Data augmentation**: Synonym replacement, back-translation for synthetic training samples
- **Cross-validation**: K-fold CV for more robust performance estimates

4. Production Readiness

- **A/B testing**: Compare model performance in live deployment
- **Feedback loop**: Continuously retrain with user-reported spam/ham labels
- **Explainability**: Integrate LIME/SHAP to explain predictions to users
- **Monitoring**: Track concept drift (changing spam patterns over time)

Conclusion

Both models achieve strong performance (>97% accuracy), but serve **different use cases**:

- **Choose Naive Bayes** if:
 - Zero false positives are critical (e.g., email filtering where blocking important messages is unacceptable)
 - Fast inference and low computational cost are required
 - Explainability is important (probability-based decisions)
- **Choose ANN** if:
 - Catching more spam is the priority (higher recall)
 - Balanced precision/recall trade-off is acceptable (higher F1-score)
 - Resources are available for training and inference

Recommended Approach: Deploy **ANN with confidence threshold tuning** to adjust precision/recall based on business requirements. Consider an **ensemble approach** combining both models (e.g., Naive Bayes for high-confidence ham, ANN for borderline cases) for optimal performance.

Visualizations

Confusion Matrices

- **ANN Confusion Matrix:** artifacts/confusion_matrix.png
- **Naive Bayes Confusion Matrix:** artifacts/naive_bayes_confusion_matrix.png

Training Dynamics

- **Training History Plot:** artifacts/training_history.png
 - Shows loss and accuracy curves across 20 epochs
 - Demonstrates convergence and slight overfitting pattern

Error Analysis

- **Misclassification Report:** artifacts/misclassification_analysis.txt
 - Detailed breakdown of 10 false positives and 19 false negatives
 - Includes original messages and confidence scores

Report Generated: November 24, 2025

Project: SMS Spam Classification using ANN and Naive Bayes

Total Test Samples: 1,114 | Training Samples: 4,454