Report - Predicting In Bed Probability

Project: Supervised Learning - Binary Classification Models **Date:** 2025-11-06 13:55:52 **Author:** Bruno Silva

Environment:

- pandas: 2.3.1
- NumPy: 2.3.1
- scikit-learn: 1.7.2

1. INTRODUCTION

Project Objective

Develop a binary classification model to predict whether a person is lying in bed based on home automation sensor data.

Business Problem

- **Home Automation**: Adjust lighting, temperature, notifications automatically
- **Sleep Monitoring**: Track sleep patterns and optimize environment
- **Energy Efficiency**: Reduce power consumption during sleep
- **Comfort**: Activate "sleep mode" at the right time

Task Type

- **Supervised Binary Classification**
- **Target**: `in_bed` (0 = not in bed, 1 = in bed)
- **Challenge**: Class imbalance

Data Sources

7 CSV files with sensor data:

1. Bedroom Blinds
2. Hallway Light
3. Bedroom Light
4. WC Light
5. Bedroom TV
6. Hallway Motion Sensor
7. Sleep Tracking (ground truth)

2. EXPLORATORY DATA ANALYSIS

Dataset Summary

- **Total Records**: 6,523
- **Features**: 18

Class Distribution

Class	Label	Count	Percentage
0	Not in Bed	5,490	84.16%
1	In Bed	1,033	15.84%

Imbalance Ratio: 5.31:1

This imbalance is expected - people spend more time awake (~16h) than sleeping (~8h).

Key Insights

- Hour of day highly correlated with target
- Sequential patterns (motion → WC → lights off) indicate bedtime routine
- Engineered 30-minute window features capture routine patterns
- All lights off is strong indicator of sleep

---

3. PREPROCESSING PIPELINE

Steps Executed

1. Data Consolidation

- Merged 7 CSV files by timestamp
- Removed 'unavailable' states
- Forward fill for last valid sensor state

2. Target Variable Creation

- Labeled timestamps within sleep windows as `in_bed=1`
- Used sleep tracking data as ground truth

3. One-Hot Encoding

- Applied to categorical sensor states
- Prevents false ordinal relationships

4. Stratified Train-Test Split (80/20)

- Maintains class proportions
- Critical for imbalanced data

5. StandardScaler

- Fitted ONLY on training data (prevents data leakage)
- Essential for KNN

---

4. MODELS TRAINED

Algorithms

1. **Logistic Regression** - Linear baseline, interpretable
2. **K-Nearest Neighbors** - Non-parametric, local patterns
3. **SVM Linear** - Optimal hyperplane, high dimensions
4. **SVM RBF** - Non-linear decision boundaries
5. **Naïve Bayes** - Fast, probabilistic

Training Times

Model	Training_Time_seconds
KNN	0.0048
NaiveBayes	0.0069
LogisticRegression	0.0337
SVM_Linear	0.2902
SVM_RBF	0.3196

5. RESULTS AND METRICS

Comparative Metrics

Model	Accuracy	Precision	Recall	Specificity	F1-Score	ROC-AUC
KNN	0.9502	0.8698	0.8068	0.9772	0.8371	0.8920
SVM_RBF	0.9295	0.7602	0.8116	0.9517	0.7850	0.8817
SVM_Linear	0.9188	0.7149	0.8116	0.9390	0.7602	0.8753
LogisticRegression	0.9188	0.7488	0.7343	0.9536	0.7415	0.8439
NaiveBayes	0.2828	0.1794	0.9855	0.1503	0.3036	0.5679

Best Model: KNN (F1-Score: 0.8371)

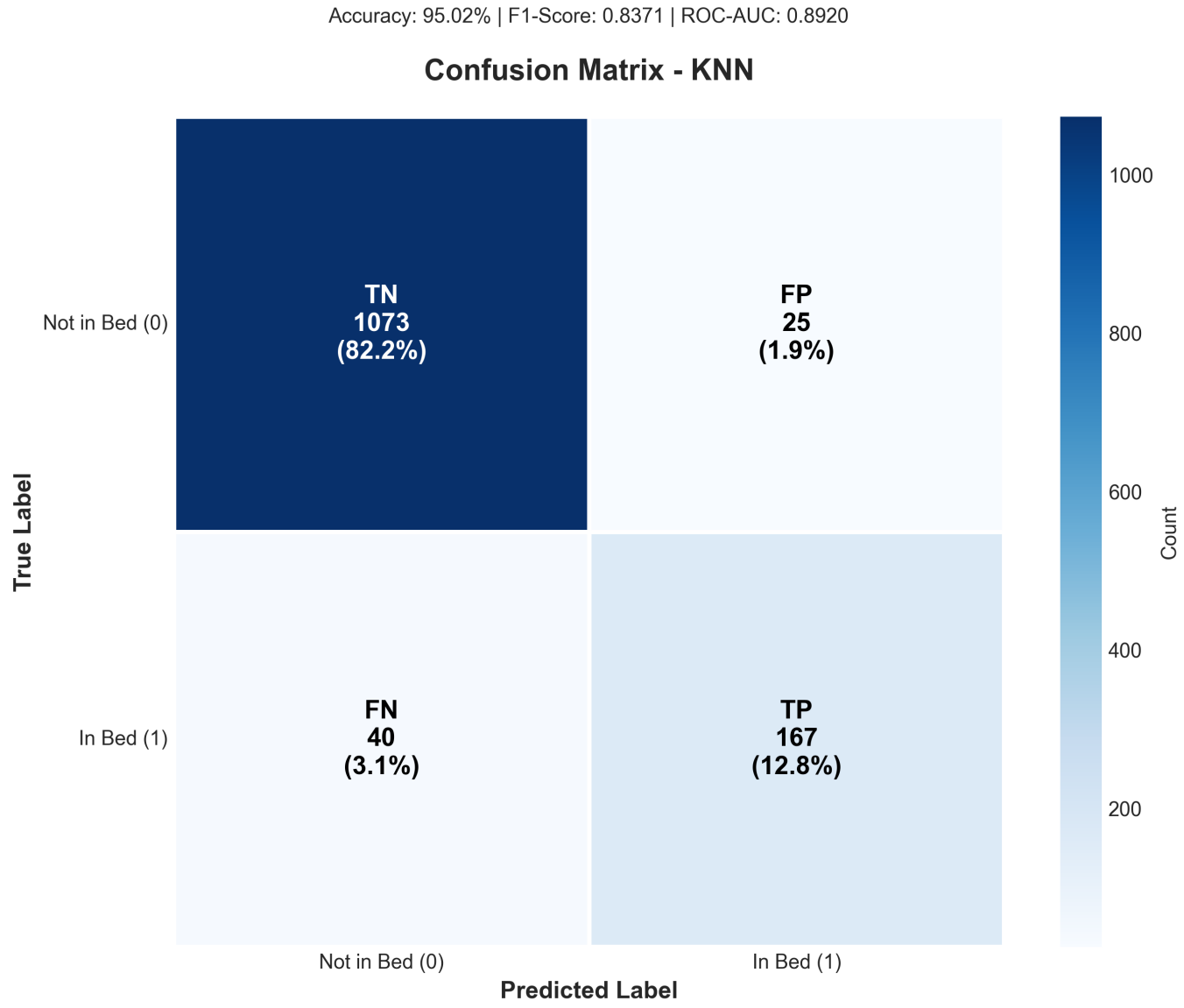
Why Accuracy Is Not Enough

In imbalanced data, a model that always predicts the majority class can have high accuracy but be useless.

Key Metrics:

- **Precision**: Of predicted "in bed", how many correct?
- **Recall**: Of actual "in bed", how many detected?
- **F1-Score**: Balance between Precision and Recall
- **ROC-AUC**: Discrimination ability across thresholds

6. CONFUSION MATRIX



Error Analysis

False Positives (FP): Predicted "in bed" when awake

- Impact: Sleep mode activates prematurely
- Severity: MODERATE (annoying but correctable)

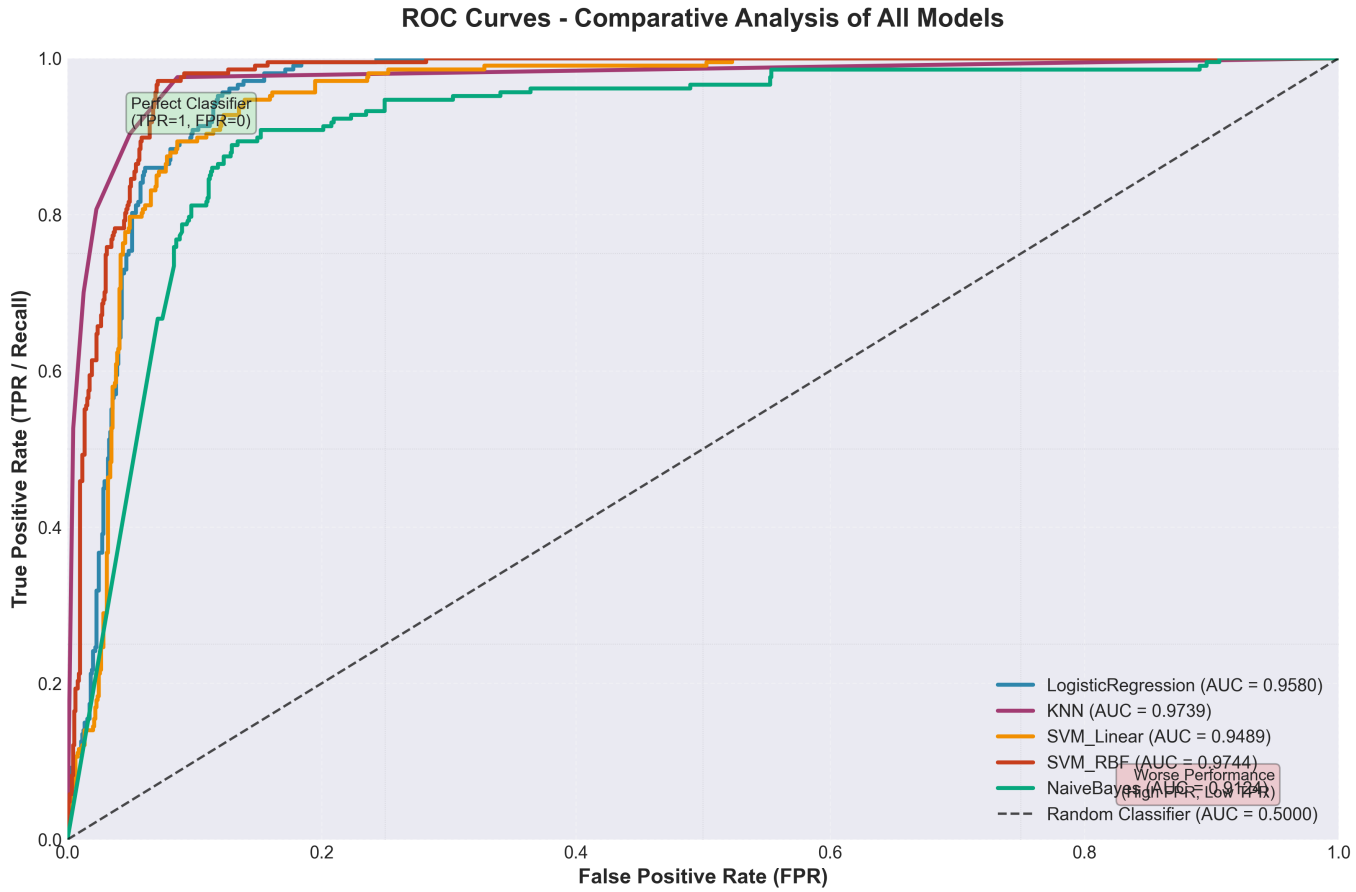
False Negatives (FN): Predicted "not in bed" when sleeping

- Impact: Sleep mode doesn't activate, disrupts sleep
- Severity: HIGH (affects sleep quality directly)

Conclusion: FN is worse in this application - missing sleep detection defeats the purpose.

---

7. ROC CURVES



AUC Scores

Rank	Model	AUC
1	SVM_RBF	0.9744
2	KNN	0.9739
3	LogisticRegression	0.9580
4	SVM_Linear	0.9489
5	NaiveBayes	0.9124

Interpretation

- **AUC = 1.0**: Perfect classifier
- **AUC >= 0.9**: Excellent
- **AUC >= 0.8**: Good
- **AUC = 0.5**: Random guessing

---

8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Recommended Model

Primary Recommendation: KNN

8.2 Handling Imbalance

Strategies:

- Use `class_weight='balanced'` in models
- Try SMOTE (oversampling)
- Consider undersampling majority class

8.3 Threshold Optimization

Default threshold (0.5) may not be optimal:

- Lower threshold → Higher Recall (fewer FN)
- Higher threshold → Higher Precision (fewer FP)
- Optimize based on cost of FN vs FP

8.4 Feature Engineering

Potential improvements:

- Time since last sensor change
- Rolling window statistics
- Cyclical encoding for hour (sin/cos)
- Interaction features

8.5 Hyperparameter Tuning

Use GridSearchCV with StratifiedKFold:

- Logistic Regression: C, penalty
- KNN: n_neighbors, weights
- SVM: C, gamma

8.6 Next Steps

1. Implement threshold tuning
2. Try class weights
3. Collect more "in bed" samples
4. Deploy best model
5. Monitor performance over time
