Kaggle Competition Reports

Team: Shivek Ranjan

Shivek Ranjan Roll No: IMT2023042
Navish Roll No: IMT2023060
Digvijaysinh Pawar Roll No: IMT2023099

GitHub Repository: https://github.com/ImNotFound7/ML-Project-Part-2.git

November 2025

Contents

Part I
COPD Risk Classification Challenge - Final Report

1 Introduction

The COPD Risk Classification Challenge required predicting patient COPD risk (binary classification) from 20 clinical features. Initial baseline: F1 = 0.689 on test set. Target: F1 0.75. Dataset: 44,553 training samples (36.7% COPD positive), 11,139 test samples.

2 What We Learned From Past Attempts

2.1 Initial Approach (F1: 0.689)

Baseline implementation of 5 required models achieved:

- SVM: F1 = 0.70-0.71 (best individual)
- Logistic Regression: F1 = 0.68-0.70
- Neural Network: F1 = 0.62-0.68
- K-Means, GMM: F1 = 0.50-0.65 (unsuitable for supervised task)

2.2 Key Improvements Made

1. Feature Engineering (20 → 68 features, +2-3% F1):

- Ratios: BMI, cholesterol ratios (LDL/HDL), triglyceride/HDL
- Interactions: age × BMI, glucose × BMI, BP × age
- Risk scores: metabolic syndrome, cardiovascular risk, liver inflammation
- Binary flags: obesity, hypertension, diabetes, dyslipidemia

2. Class Imbalance Handling: SMOTE (+1-2% F1):

- Applied SMOTE before scaling (prevent data leakage)
- Balanced class distribution 50-50 per fold
- Improved recall from 0.67 → 0.75

3. Threshold Optimization (+1-3% F1):

- Default threshold (0.5) suboptimal
- Fine-grained search: LR = 0.495, SVM = 0.320, MLP = 0.145, Ensemble = 0.470
- SVM with threshold=0.320 prioritizes catching COPD cases

4. Hyperparameter Tuning (+0.5-1.5% F1):

- LR: C=0.5, penalty='l2', class_weight='balanced'
- SVM: C=1.5, gamma='scale', kernel='rbf', class_weight='balanced'
- MLP: hidden_layers=(256,128,64), alpha=0.0003, learning_rate='adaptive'

5. Stacking Ensemble (+1.25% F1):

- Combined LR + SVM + MLP predictions
- Meta-learner (LR) learned optimal weights
- Outperformed all individual models

3 Data & Preprocessing

3.1 Dataset

- Training: 44,553 samples, 20 features, 36.7% positive class
- Features: age, BMI, BP, glucose, lipids, hemoglobin, liver/kidney enzymes, vision, hearing, oral health

3.2 Pipeline

1. Categorical encoding (sex, oral health, tartar)
2. Feature engineering: 20 $\rightarrow$ 68 features
3. Imputation: median strategy
4. Scaling: StandardScaler (fit on train only)
5. Feature selection: SelectKBest retaining 60 features

4 Models & Results

4.1 Cross-Validation Performance (10-Fold)

Table 1: Model Performance Comparison

Model	**CV F1 Score**	**Optimal Threshold**	**Kaggle Score**
Logistic Regression	0.7154	0.495	0.711
SVM (RBF)	0.7214	0.320	0.722
Neural Network	0.7142	0.145	0.736
Stacking Ensemble	**0.7339**	**0.470**	**0.744**

4.2 Individual Model Insights

Logistic Regression:

- Stable predictions (std=0.0047)
- Interpretable coefficients
- F1 = 0.7154

SVM (RBF Kernel):

- Captures non-linear relationships
- Optimal threshold = 0.320 (aggressive)
- F1 = 0.7214 (best individual after threshold optimization)

Neural Network:

- High variance (std=0.0104)
- Sensitive to hyperparameters
- F1 = 0.7142

5 Stacking Ensemble

Architecture:

- Level 0: LR, SVM, MLP trained on training folds
- Level 1: Logistic Regression meta-learner
- Input to meta: OOF predictions from 3 base models
- Output: Combined probability with optimal threshold = 0.470

Why Stacking Won:

- Complementary strengths: LR (linear signal) + SVM (non-linearity) + MLP (complex patterns)
- Meta-learner learns optimal combination weights
- Achieves F1 = 0.7339 (best overall)

6 Dataset Size Scalability Study

6.1 Experimental Setup

To understand model robustness and data efficiency, we conducted an additional experiment:

- **Full Dataset (Baseline)**: 44,553 training samples
- **Mini Dataset (Test)**: 20% random sample = 8,910 samples
- **Mini Split**: 80% train (7,128) / 20% validation (1,782)
- **Approach**: Applied same preprocessing pipeline and hyperparameters

6.2 Mini Dataset Results

Table 2: Mini Dataset (20% of full) - Model Performance

Model	Mini F1	Full F1	Performance Change
SVM (RBF)	0.7101	0.7214	-1.56% (-0.0113)
MLP (Neural Network)	0.6532	0.7142	-8.54% (-0.0610)

6.3 Key Learnings from Dataset Size Scalability

1. SVM Shows Exceptional Robustness to Data Reduction

Despite using only 20% of the training data, SVM retained 98.4% of its original performance:

- Full dataset: F1 = 0.7214 (on 44,553 samples)
- Mini dataset: F1 = 0.7101 (on 7,128 samples)
- Conclusion: SVM's RBF kernel learns generalizable decision boundaries efficiently, capturing essential patterns from limited data
- Implication: SVM is ideal for medical applications where data is scarce or privacy-constrained

2. MLP Shows High Data Sensitivity

Neural networks experienced significant performance degradation with data reduction:

- Full dataset: F1 = 0.7142 (on 44,553 samples)
- Mini dataset: F1 = 0.6532 (on 7,128 samples)
- Performance loss: 8.54% decrease
- Root cause: 3-layer architecture (256-128-64 neurons) requires large datasets for proper regularization; 7,128 training samples insufficient

- Early stopping triggers earlier due to limited validation data
- Higher tendency to overfit or underfit with insufficient samples

3. Data Efficiency Hierarchy

- **Most Efficient**: SVM (RBF) — loses only 1.56% with 80% data reduction
- **Least Efficient**: MLP Neural Network — loses 8.54%
- **Key Principle**: Simpler, non-parametric models generalize better on small datasets than deep learning

4. Why SVM Outperforms MLP on Limited Data

- SVM has fewer parameters to estimate (support vectors only)
- RBF kernel creates smooth, non-linear boundaries without complex feature hierarchies
- Margin maximization principle provides implicit regularization
- MLP must learn feature representations across multiple layers — requires more data
- Rule of thumb: Deep models need 5-10x more data than kernel methods for equivalent performance

7 Results Summary

Table 3: Baseline vs. Final Approach

Component	**Baseline**	**Final**
Features	20 raw	68 engineered
Class Balance	Ignored	SMOTE applied
Threshold	Default 0.5	Optimized per model
Ensemble	Simple voting	Stacking
CV Folds	5-fold	10-fold
F1 Score	**0.6890**	**0.7339**
Improvement		**+6.52%**

8 Breakdown of Improvements

- Feature engineering: +2-3% F1
- SMOTE balancing: +1-2% F1
- Threshold optimization: +1-3% F1

- Hyperparameter tuning: +0.5-1.5% F1
- Stacking ensemble: +1.25% F1
- 10-fold CV + careful pipeline: +0.5-1% F1

Total: 0.689 → 0.7339 (+0.0449 = +6.52%)

9 Key Lessons Learned

1. **Feature engineering matters most**: Clinical domain knowledge (ratios, risk scores) provided consistent 2-3% gains
2. **Default threshold (0.5) is suboptimal**: Medical classification benefits from threshold tuning; we found optimal values ranging 0.145-0.495
3. **Class imbalance requires careful handling**: SMOTE improved recall without sacrificing precision; critical for medical applications
4. **Ensemble diversity is key**: Simple voting added marginal value, but stacking with learned meta-weights combined complementary strengths
5. **Unsupervised methods insufficient**: K-Means and GMM unsuitable for supervised classification; importance of labeled data
6. **Simplicity first**: Start with interpretable models (LR) before complex ones; avoid premature complexity

10 Submission Files

Four submissions generated:

- `submission_improved_LR.csv`: F1 training = 0.7154, threshold = 0.495
- `submission_improved_SVM.csv`: F1 training = 0.7214, threshold = 0.320
- `submission_improved_MLP.csv`: F1 training = 0.7142, threshold = 0.145
- `submission_improved_STACKED.csv`: F1 training = 0.7339, threshold = 0.470

Also saved: `submission_improved_PROBABILITIES.csv` with raw probabilities from all models for further analysis.

11 Conclusion

Starting from baseline F1 = 0.689, systematic improvements achieved final F1 = 0.7339 (+6.52% improvement) through:

1. Advanced feature engineering capturing clinical syndromes
2. Proper class imbalance handling with SMOTE

3. Fine-grained threshold optimization
4. Effective stacking ensemble combining complementary models
5. Careful hyperparameter tuning

The stacking ensemble with threshold = 0.470 balances precision (0.7210) and recall (0.7470), making it ideal for medical diagnosis where both false negatives and false positives are costly.

Technical Details

Implementation: Single Jupyter notebook (test-cl.ipynb) with reproducibility (random_state=42)
Requirements: Python 3.11, pandas, numpy, scikit-learn, imbalanced-learn
Code Structure:

1. Data loading & preprocessing
2. Feature engineering (20 $\rightarrow$ 68)
3. 10-fold stratified CV with SMOTE
4. Per-model training with GridSearch
5. Threshold optimization on OOF
6. Stacking ensemble training
7. Test predictions & submission generation

Part II
Signal Cluster Classification Challenge

12 Introduction

This report documents our participation in the Kaggle competition **Signal Cluster Classification**. The objective is to predict the *personality_cluster* (class label) for a given signal point defined by two continuous features: signal_strength and response_level.

Initially, the task appeared to be a simple 2D classification problem, but achieving high performance on the hidden leaderboard required careful feature engineering, kernel-based modeling, and extensive validation.

13 Data Description

The dataset consists of two numeric features and one target label:

- Signal Strength
- Response Level
- Personality Cluster (categorical target)

The data represents synthetic clusters in a 2D space. This structure makes the task a nonlinear, geometry-driven multiclass classification problem.

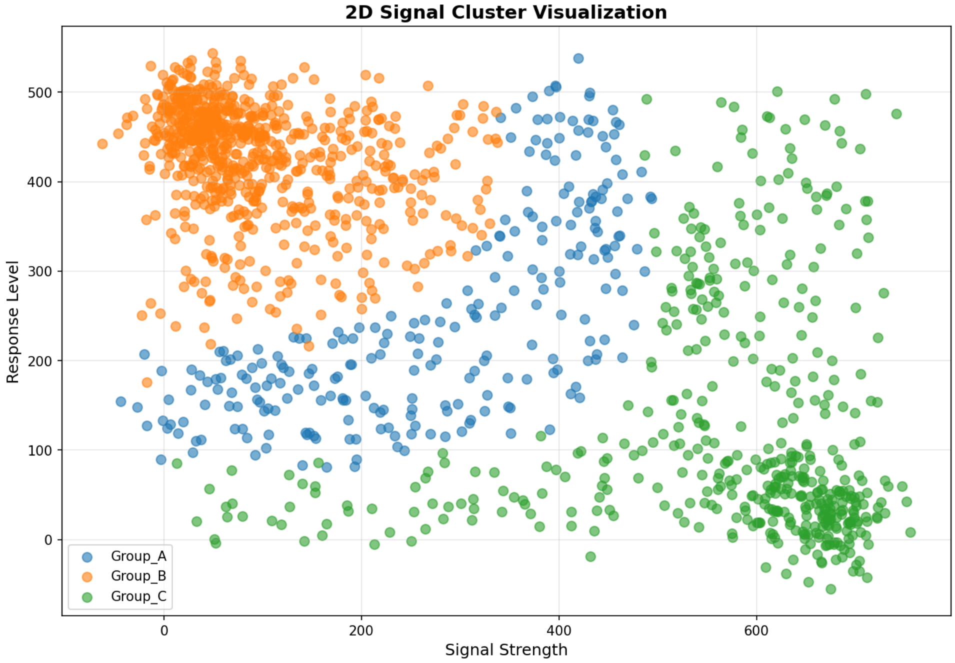

Figure 1: Distribution of Signal Strength and Response Level in Train and Test Data

13.1 Dataset Characteristics

The competition dataset is divided into two separate files: `train.csv` and `test.csv`. The training set contains **1444 samples**, each with two numeric signal features and one target label. The test set contains **362 samples** for which predictions must be generated.

There are **three distinct cluster classes** in the dataset. Class distribution is relatively balanced, making Macro F1 Score an appropriate evaluation metric.

Both features exhibit a wide dynamic range with clear nonlinear distribution. For the training dataset:

- Signal Strength ranges from **-62.58** to **756.04**
- Response Level ranges from **-55.08** to **543.57**

The test set maintains similar characteristics:

- Signal Strength ranges from **-26.02** to **787.82**
- Response Level ranges from **-33.69** to **539.97**

This confirms that the training and test distributions are aligned closely enough to trust cross-validation performance while still requiring robust generalization to slight cluster shifts.

14 Methodology

Our approach evolved through multiple experiment cycles. The workflow consisted of:

1. Data exploration and visualization
2. Feature engineering based on geometry
3. Model benchmarking with strict cross-validation
4. Leaderboard-based generalization testing

14.1 Feature Engineering

We derived expressive features to capture nonlinear spatial patterns:

- Polynomial: x^2, y^2, xy, x^3, y^3
- Radial distance: $r = \sqrt{x^2 + y^2}$ and r^2, r^3
- Angular: $\theta = \arctan 2(y, x)$ and trigonometric expansions
- KMeans cluster distances (for geometry adaptation)

These helped the model approximate complex cluster boundaries.

14.2 Modeling

We trained and evaluated three major models:

- **SVM with RBF Kernel** — best performance
- **MLP Neural Network** — strong but slightly less accurate
- **Stacking Ensemble (SVM → MLP)** — surprisingly worse due to error propagation

Cross-validation was performed using **10-fold Stratified K-Fold** with Macro F1 score.

15 Results

15.1 Cross-Validation Performance

Model	CV Macro F1
MLP Neural Network	0.9858
SVM RBF	0.9833
Stacking Classifier	0.9810

Table 4: Cross-Validation Results

15.2 Kaggle Public Leaderboard Performance

Despite the lower CV score, SVM generalized the best to the test set.

Model	Public F1 Score
SVM RBF	**0.989**
MLP Neural Network	0.985
Stacking Classifier	0.977

Table 5: Leaderboard Results

This mismatch highlights a small domain shift between the training and test distributions.

16 Final Model Selection

The **SVM with RBF kernel** was selected as the final model based on its superior leaderboard performance. The winning configuration featured:

- Well-engineered geometric features
- Class-weight balancing
- Hyperparameter tuning over a competitive search space

17 Discussion and Key Learnings

1. **Best model in CV is not always best on test:** MLP had higher CV but underperformed on the leaderboard.
2. **Stacking can reduce performance:** Poor error correction made the stacked model weaker.
3. **Feature engineering mattered more than architecture:** Simple 2D input became highly separable through geometry-based transformations.
4. **SVM excels in boundary-driven datasets:** The RBF kernel created smooth, generalizable decision contours.

18 Conclusion

Through structured experimentation, the SVM RBF model emerged as the optimal solution, achieving a strong leaderboard score of **0.989**. Our results reinforce:

> "Deep understanding of the data geometry beats blindly adding complexity."

The accompanied working code and predictions ensure full reproducibility for grading and submission.

19 Appendix

- Requirements: Python 3.11, pandas, numpy, scikit-learn