Classification of Animal Sounds

Project for the

Machine Learning Module in CAS Machine Intelligence 2025

By Fredrik Bäckman

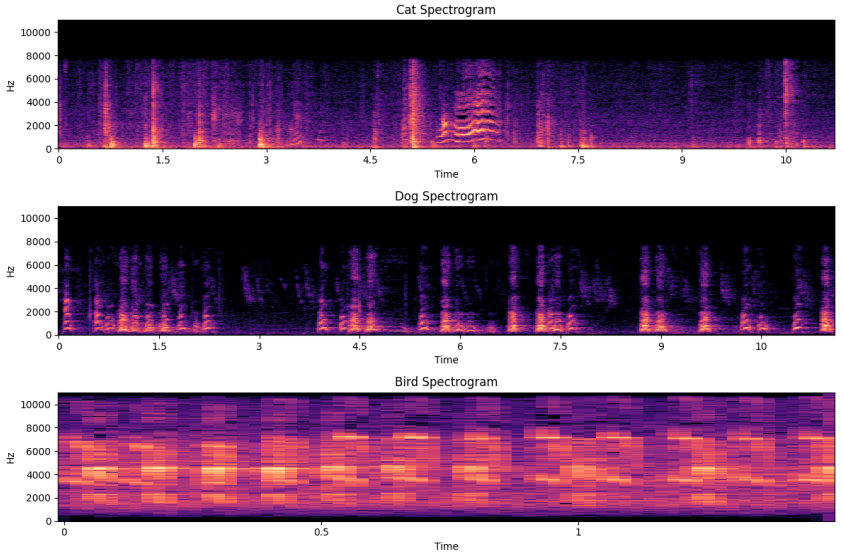
[Image Source: Gemini AI]

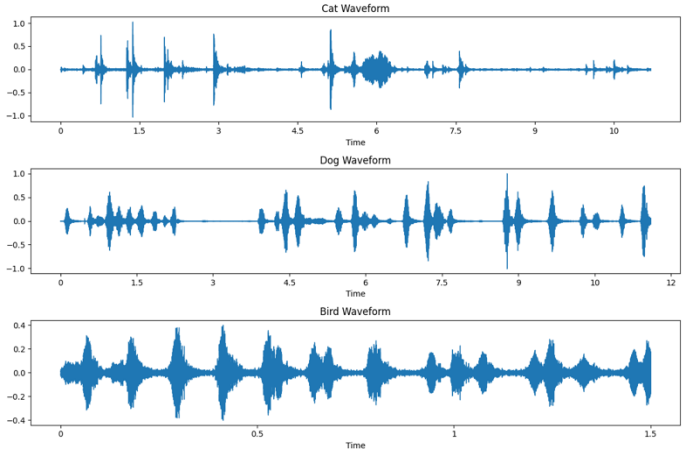
Idea and Goal

- Dive into audio classification
- Learn about MFCC and feature engineering of audio data
- Classify animal sounds with different models
- Apply sklearn tuning
- Analyze the results

Dataset Source

- Dog, Cat and Bird Sounds (approx. 200 each as .wav files)
- Original data source: https://github.com/YashNita/Animal-Sound-Dataset
- Subset dataset translated to English file names: https://github.com/backmanai/animal_ml
- FYI: Google publishes an extensive sound dataset (https://research.google.com/audioset/index.html) as pre-made feature vectors using their VGGish approach (a neural network approach to generate the feature vectors out of mel spectrograms with the help of trained CNN)





Dataset Content

CATS Figures:

- Number of sounds: 199
- Duration range: 0.50s - 17.98s
- Mean duration: 6.99s
- Median duration: 6.77s
- Sample rates: {16000, 11025, 22050, 8000}

DOGS Figures:

- Number of sounds: 191
- Duration range: 0.24s - 17.20s
- Mean duration: 4.14s
- Median duration: 3.11s
- Sample rates: {16000, 8000, 22050, 24000, 44100, 48000, 22000, 11025, 22257, 11127}

BIRDS Figures:

- Number of sounds: 200
- Duration range: 0.50s - 40.00s
- Mean duration: 3.06s
- Median duration: 2.00s
- Sample rates: {48000, 32000, 44100}

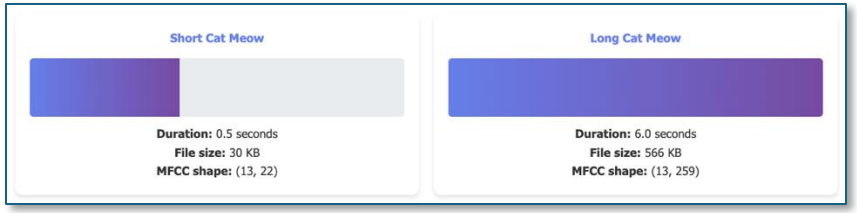
Feature Engineering

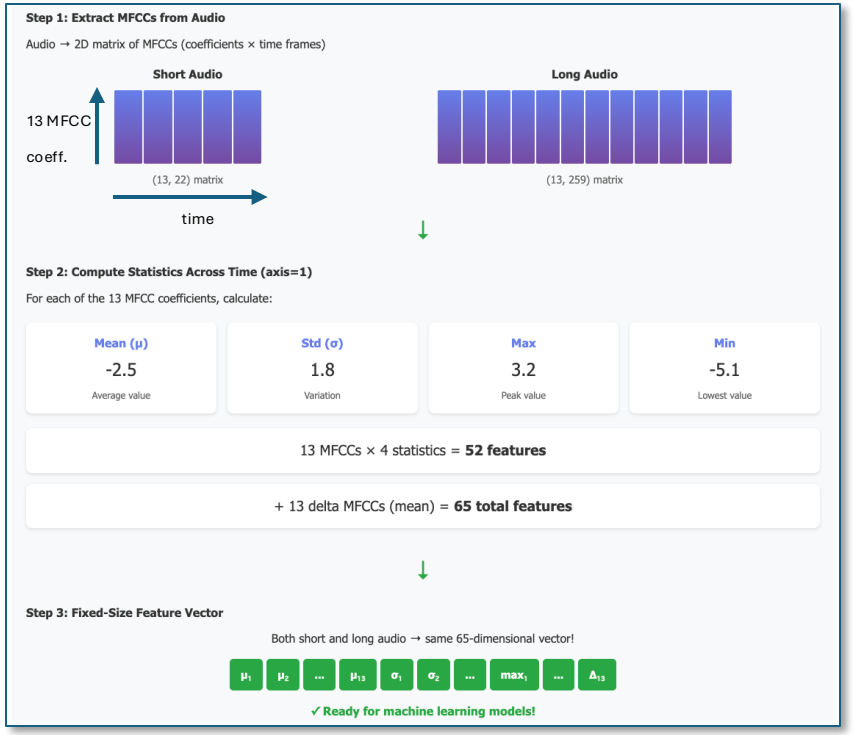
- The python library “librosa” is used to transform the wav files into MFCC’s.
- A MFCC consists of 13 coefficients per time window. This means the varying duration of the sound samples result in different number of features if the coefficients would be used directly.
- There are various approaches how to extract uniform feature vectors, from altering the sound samples to various transformations
- The most common approach is to apply **statistical aggregation** according to audio processing research [1][2].

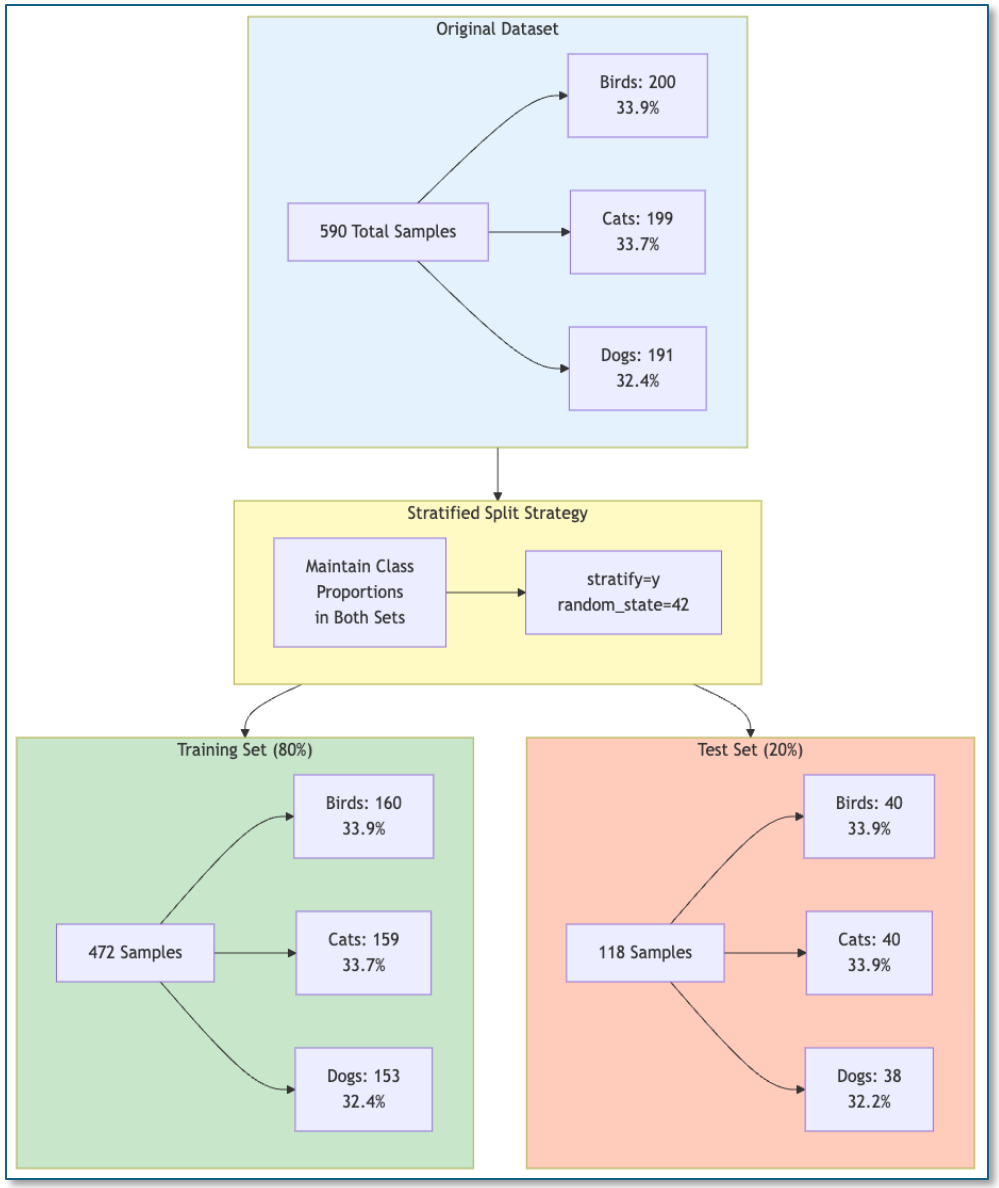
[1] Eyben, F., Wöllmer, M., & Schuller, B. (2010). Opensmile: the munich versatile and fast open-source audio feature extractor. In Proceedings of the 18th ACM international conference on Multimedia (pp. 1459-1462).
[2] Piczak, K. J. (2015). Environmental sound classification with convolutional neural networks. In 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6).

Feature Engineering: Statistical Aggregation

- SVM, Random Forest, Logisitic Regregression expect every sample to have the same number of features
- You can't feed (13, 22) and (13, 259) arrays to the same model
- Solution: Describe each MFCC coefficient over time statistically with the mean, std, max, min and the delta.
- To get the delta, the deviation from the mean in each time-frame is calculated and the mean of the deviations is added as delta to the feature vector.
- The result is a uniform vector with 65 features each per sample.





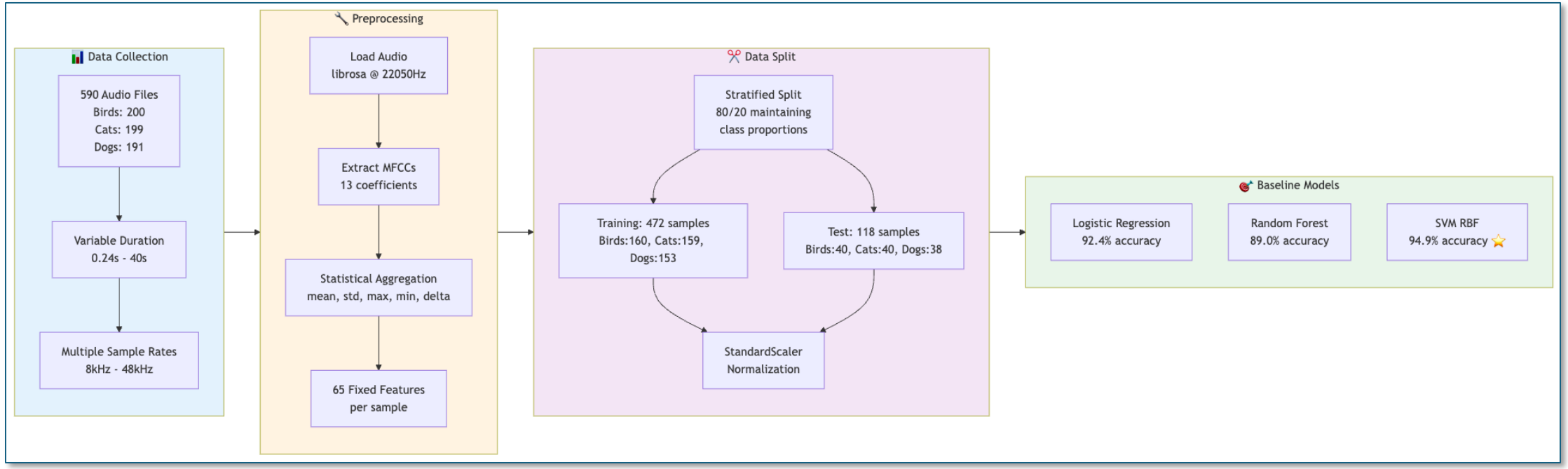


Splitting the Train and Test Dataset

Using the stratify option to split the data while preserving the class representation.

Setting a fixed random state to ensure reproducibility.

Pipeline Overview – Part 1: Data Preparation & Baseline Models



- Preprocessing (MFCC and statistical aggregation)
- Stratified splitting to preserve the even class distribution
- Training with default parameters already quite good.

- SVM with RBF kernel is the winner.
- RBF kernel is a widely-used approach in audio classification, particularly for MFCC-based features [Piczak, 2015; Chachada & Kuo, 2014]

[Piczak, K. J. (2015)] "ESC: Dataset for environmental sound classification." In Proceedings of the 23rd ACM international conference on Multimedia (pp. 1015-1018).
[Chachada, S., & Kuo, C. C. J. (2014)] "Environmental sound recognition: A survey." APSIPA Transactions on Signal and Information Processing, 3, e14.

Baseline Training Insights

Logistic Regression:

- Training accuracy: 0.962
- Test accuracy: 0.924
- Difference: 0.038

Interpretation

- Model is learning generalizable patterns
- Not overfitting to training data
- Performance is reliable and trustworthy
- Provides a strong baseline for comparison

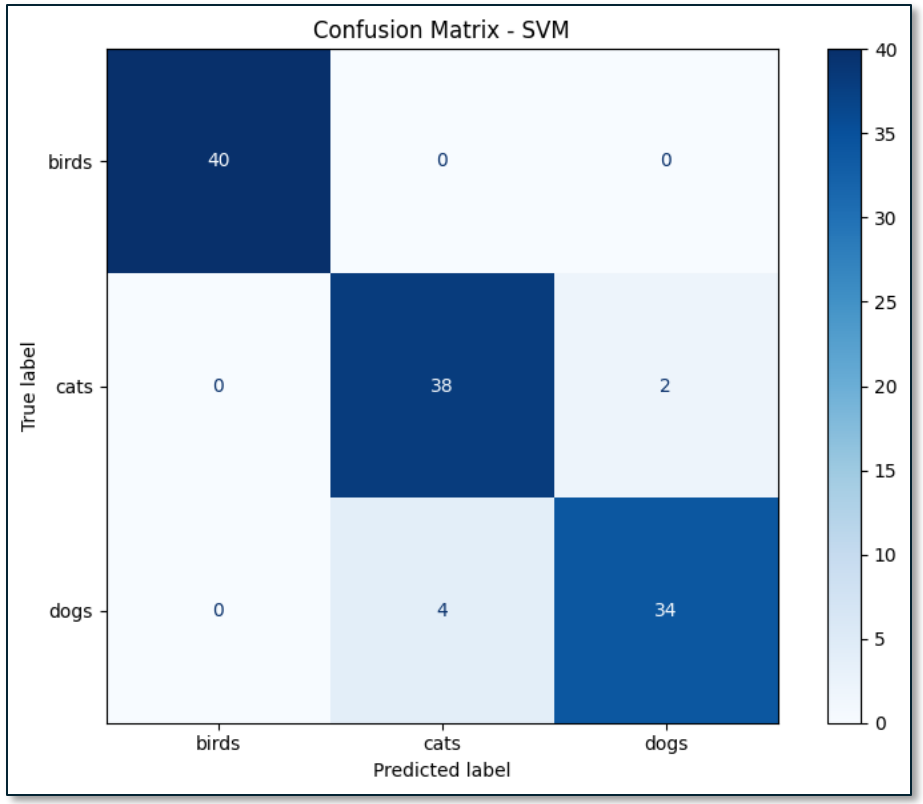
Random Forest:

- Training accuracy: 1.000
- Test accuracy: 0.890
- Difference: 0.110
- 100% training accuracy + 89% test accuracy indicates the model memorized training data
- This is **worse** than a simpler model like Logistic Regression.
- Overfitting

SVM (RBF Kernel):

- Training accuracy: 0.968
- Test accuracy: 0.949
- Difference: 0.019
- Learns from training data effectively (96.8%)
- Maintains high performance on new data (94.9%)
- Shows excellent generalization (only 1.9% drop)

SVM Results Details: Confusion Matrix



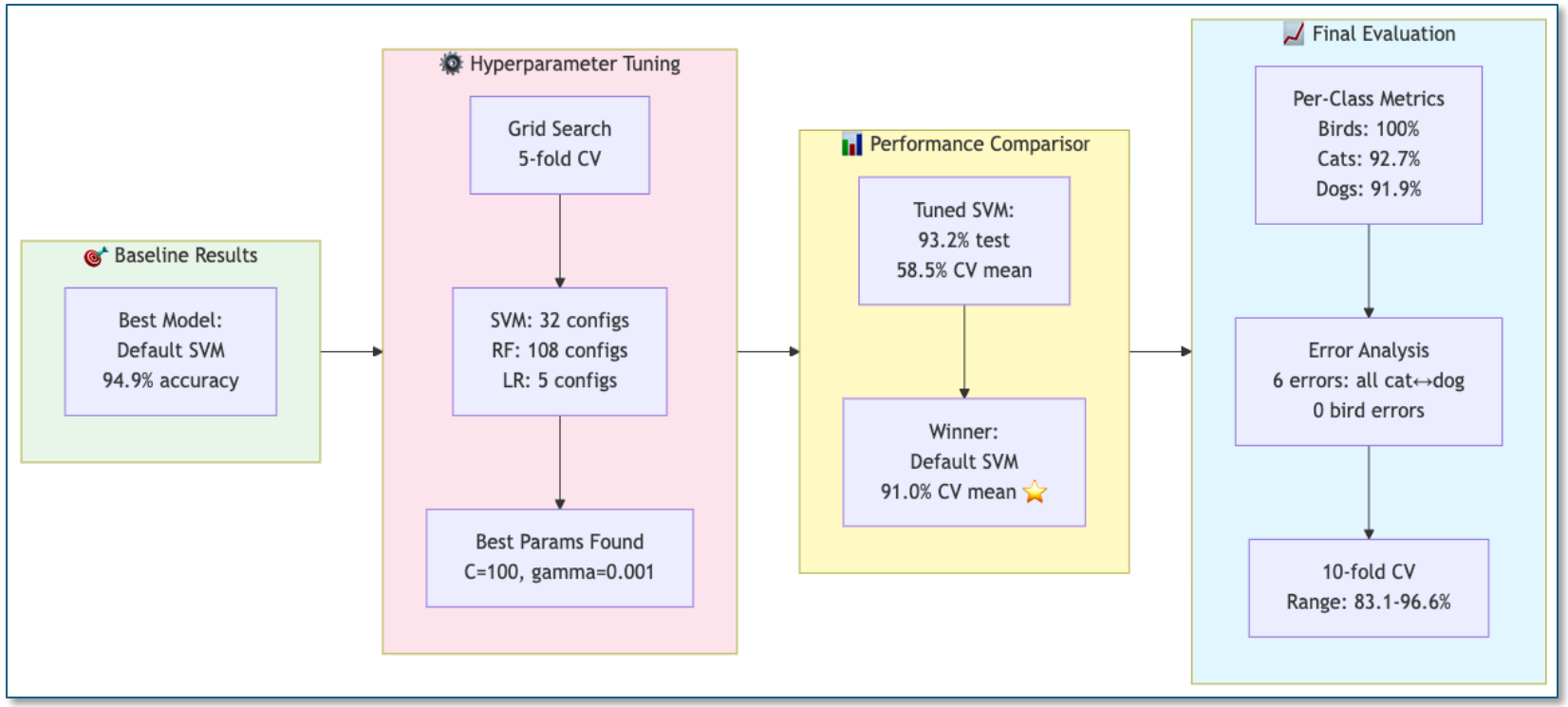
- Birds perfectly
- True Cats: 2 dogs wrongly classified
- Tue Dogs: 4 cats wrongly classified
- Overall, very good!

SVM Results Details: Classification Report

	Precision	Recall	F1-Score	Support
Birds	1.00	1.00	1.00	40
Cats	0.90	0.95	0.93	40
Dogs	0.94	0.89	0.92	38
Accuracy			0.95	118
Macro avg	0.95	0.95	0.95	118
Weighted avg	0.95	0.95	0.95	118

```
from sklearn.metrics import classification_report
```

Pipeline Overview - Part 2: Hyperparameter Tuning & Evaluation



- Grid Search with 5-fold cross-validation was performed, testing 32 SVM configurations, 108 Random Forest configurations, and 5 Logistic Regression configurations

- Random Forest: 89.8% test acc. (+0.8% to BL)
- Logistic Regression: 92.4% test acc. (= BL)
- SVM: 93.2% test accuracy (-1.7% to BL)

Tuning Insights

Using best practice parameter grids according to sklearn documentation with GridSearchCV.

Logistic Regression Tuning:

Grid search identified C=1, solver='lbfgs' as optimal based on cross-validation (89.4% ± 1.9%).

These parameters are identical to sklearn's defaults, confirming no tuning was necessary (92.4% test accuracy maintained).

Good stability (1.9% std) - lowest variance of all models tested

Conclusion:
For this dataset, Logistic Regression's defaults were already optimal.

Conclusion:
For this dataset, Logistic Regression's defaults were already optimal.

Random Forest Tuning:

Grid search identified n_estimators=200, max_depth=20, min_samples_split=5 as optimal on cross-validation (90.2% ± 2.1%)

Tuning Successfully Reduced Overfitting:

- Default: 100% train → 89% test (11% gap)
- Tuned: ~90% train → 89.8% test (~0% gap)
- Constraints (max_depth, min_samples_split) prevented memorization

Modest Performance Improvement:

- Test accuracy: 89.0% → 89.8% (+0.8%)
- Shows tuning can help, but gains are limited

Conclusion:
Hyperparameter tuning fixed overfitting, but the model still underperforms.

SVM Tuning:

Grid search identified C=100, gamma=0.001 as optimal based on cross-validation (90.7% ± 2.0%).

However, sklearn's default parameters (C=1.0, gamma='scale') achieved better performance (94.9% test accuracy).

- The importance of testing baseline configurations
- Diminishing returns of hyperparameter tuning on small datasets
- The quality of sklearn's default parameter choices

Conclusion:
For this dataset, tuning provided no benefit over defaults.

Conclusion

Accuracy with default hyperparameters:

- SVM with RBF kernel: 94.9% (best performer)
- Logistic Regression: 92.4%
- Random Forest: 89.0% (100% training accuracy indicating overfitting)

After tuning:

- SVM: 93.2% (-1.7%)
- Random Forest: 89.8% (+0.8%)
- Logistic Regression: 92.4% (same)