Confirm
Smart Manufacturing

Ensemble Methods for Collective Intelligence: Combining Ubiquitous ML Models in IoT

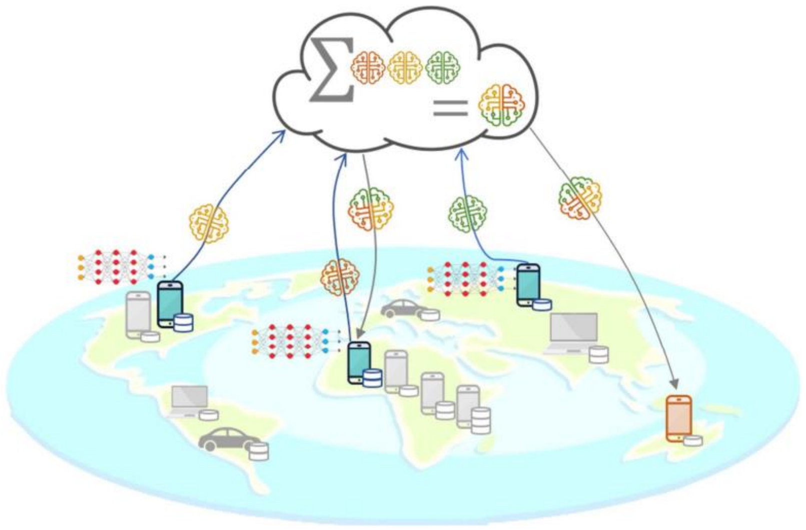
Bharath Sudharsan

A World Leading SFI Research Centre

Science Foundation Ireland For what's next

Introduction

Confirm
Smart Manufacturing



- Concept of **ML model aggregation rather than data aggregation** has gained much attention as it boosts prediction performance while maintaining stability and preserving privacy

- In a non-ideal scenario, there are chances for a base model trained on a single device to make **independent but complementary errors**

- To handle such cases, we implement 8 robust ML model combining methods that achieves reliable prediction results by combining numerous base models (trained on many devices) to form a central model that effectively **limits errors, built-in randomness and uncertainties**

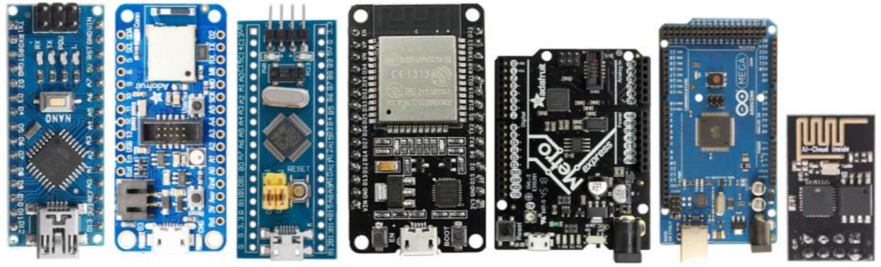
Contribution to TinyML

Confirm
Smart Manufacturing

Powerful CPU + basic GPU based SBCs (Single Board Computers)

Set 1

Billions of IoT devices are designed using such MCUs and small CPUs



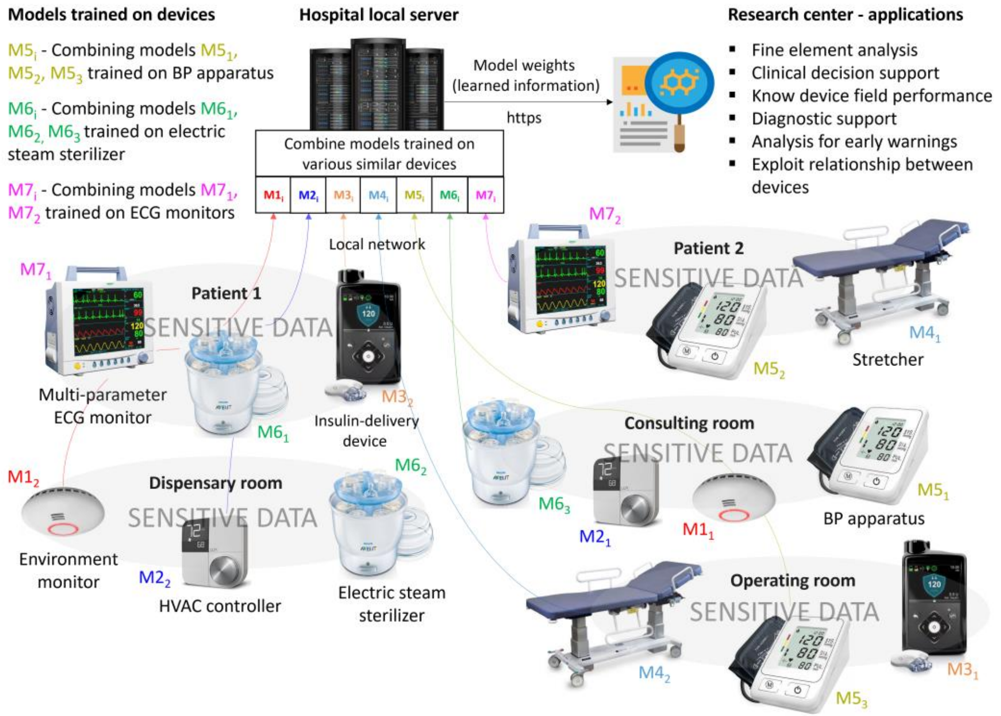
Set 2

- The studies from centralized learning, split learning, distributed ensemble learning have extensively investigated combining models trained on devices like smartphones, Raspberry Pis, Jetson Nanos (Set 1). Such devices have **sufficient resources** to train base models (or ensembles) using standard training algorithms from Python Scikit-learn or light version ML frameworks like TensorFlow Lite

- In contrast, we aim to achieve collective intelligence using MCUs, small CPUs (Set 2) since billions of deployed IoT devices like HVAC controllers, smart meters, video doorbells have **resource-constrained** hardware with only a few MB of memory

Medical Data Use case

Confirm
Smart Manufacturing

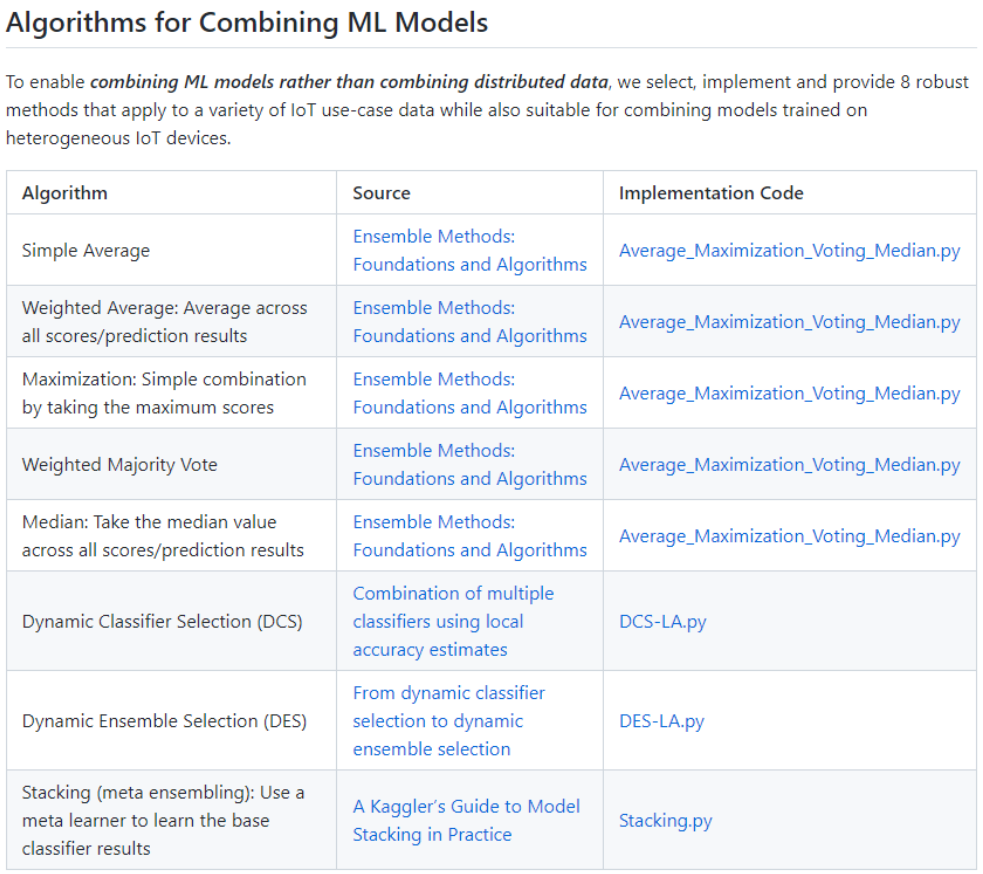


Models trained on devices
$M5_i$ - Combining models $M5_1$, $M5_2$, $M5_3$ trained on BP apparatus
$M6_i$ - Combining models $M6_1$, $M6_2$, $M6_3$ trained on electric steam sterilizer
$M7_i$ - Combining models $M7_1$, $M7_2$ trained on ECG monitors
Hospital local server
Combine models trained on various similar devices
$M1_i$ $M2_i$ $M3_i$ $M4_i$ $M5_i$ $M6_i$ $M7_i$
Model weights (learned information)
https
Research center - applications
Fine element analysis
Clinical decision support
Know device field performance
Diagnostic support
Analysis for early warnings
Exploit relationship between devices
Local network
$M7_1$
Patient 1
SENSITIVE DATA
Multi-parameter ECG monitor
$M6_1$
Insulin-delivery device
$M3_2$
$M7_2$
Patient 2
SENSITIVE DATA
$M4_1$
Stretcher
$M5_2$
$M1_2$
Dispensary room
SENSITIVE DATA
Environment monitor
$M2_2$
HVAC controller
$M6_2$
Electric steam sterilizer
Consulting room
SENSITIVE DATA
$M6_3$
$M2_1$
$M1_1$
$M5_1$
BP apparatus
Operating room
SENSITIVE DATA
$M4_2$
$M5_3$
$M3_1$

Providing sensitive medical data for research can be a potential application where combining ML models can be utilized

- Data required for most research are sensitive in nature, as it revolves around a private individual
- GDPR restricts sending sensitive yet valuable medical data (from hospitals, imaging centers) to research institutes
- When medical devices are equipped with hardware-friendly training algorithms they can perform onboard training of base models
- After training, base models from similar devices can be extracted, combined, and sent to research labs with improved data privacy preservation
- **Example.** The 2 base models $M7_1$, $M7_2$ trained on ECG monitors using vital data of patients can be combined centrally, then shared for research

Model Combining Methods



Algorithms for Combining ML Models

To enable ***combining ML models rather than combining distributed data***, we select, implement and provide 8 robust methods that apply to a variety of IoT use-case data while also suitable for combining models trained on heterogeneous IoT devices.

Algorithm	Source	Implementation Code
Simple Average	Ensemble Methods: Foundations and Algorithms	Average_Maximization_Voting_Median.py
Weighted Average: Average across all scores/prediction results	Ensemble Methods: Foundations and Algorithms	Average_Maximization_Voting_Median.py
Maximization: Simple combination by taking the maximum scores	Ensemble Methods: Foundations and Algorithms	Average_Maximization_Voting_Median.py
Weighted Majority Vote	Ensemble Methods: Foundations and Algorithms	Average_Maximization_Voting_Median.py
Median: Take the median value across all scores/prediction results	Ensemble Methods: Foundations and Algorithms	Average_Maximization_Voting_Median.py
Dynamic Classifier Selection (DCS)	Combination of multiple classifiers using local accuracy estimates	DCS-LA.py
Dynamic Ensemble Selection (DES)	From dynamic classifier selection to dynamic ensemble selection	DES-LA.py
Stacking (meta ensembling): Use a meta learner to learn the base classifier results	A Kaggler's Guide to Model Stacking in Practice	Stacking.py

- From the available multitudinous number of studies, we choose, implement, and provide 8 robust ML model combining methods
- Compatible with a wide range of datasets (varying feature dimensions and classes) and IoT devices (heterogeneous hardware specifications)
- Open-source implementation, utilizing which researchers and engineers can start practicing distributed ensemble learning by combining ML base models trained on ubiquitous IoT devices

Implementation

Confirm
Smart Manufacturing

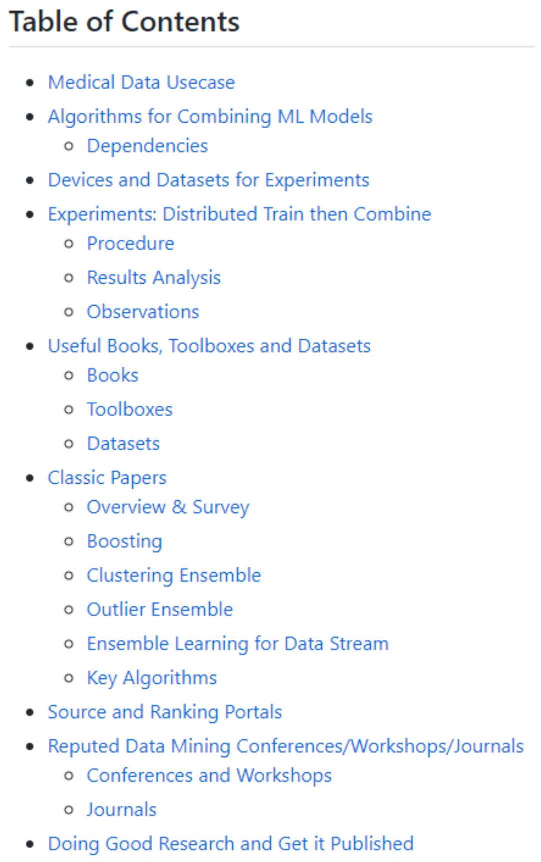
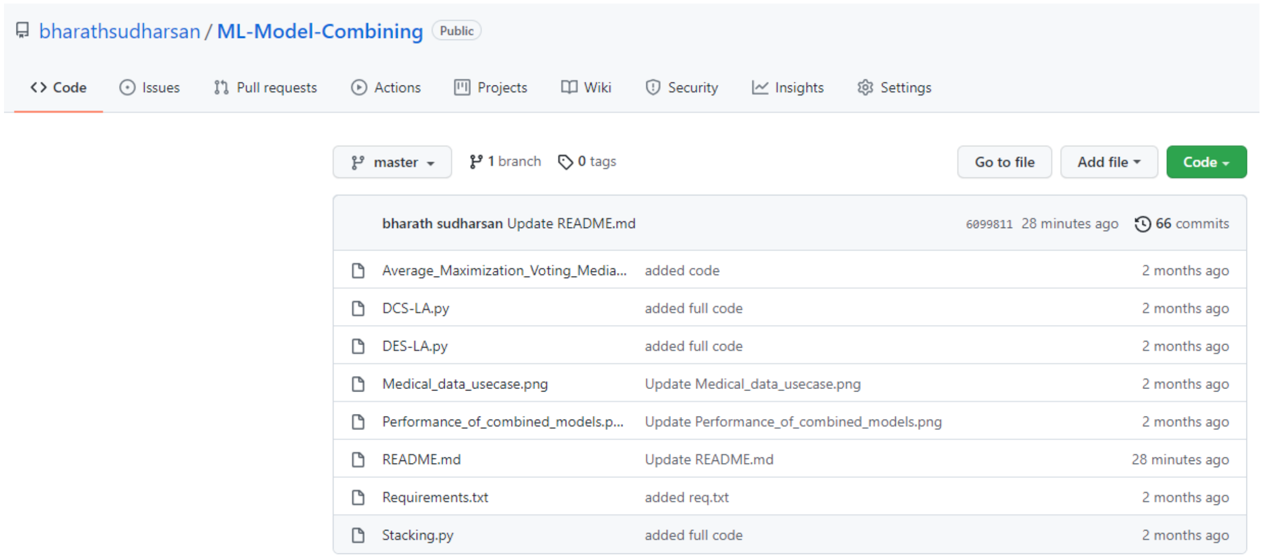


Table of Contents

- Medical Data Usecase
- Algorithms for Combining ML Models
  - Dependencies
- Devices and Datasets for Experiments
- Experiments: Distributed Train then Combine
  - Procedure
  - Results Analysis
  - Observations
- Useful Books, Toolboxes and Datasets
  - Books
  - Toolboxes
  - Datasets
- Classic Papers
  - Overview & Survey
  - Boosting
  - Clustering Ensemble
  - Outlier Ensemble
  - Ensemble Learning for Data Stream
  - Key Algorithms
- Source and Ranking Portals
- Reputed Data Mining Conferences/Workshops/Journals
  - Conferences and Workshops
  - Journals
- Doing Good Research and Get it Published



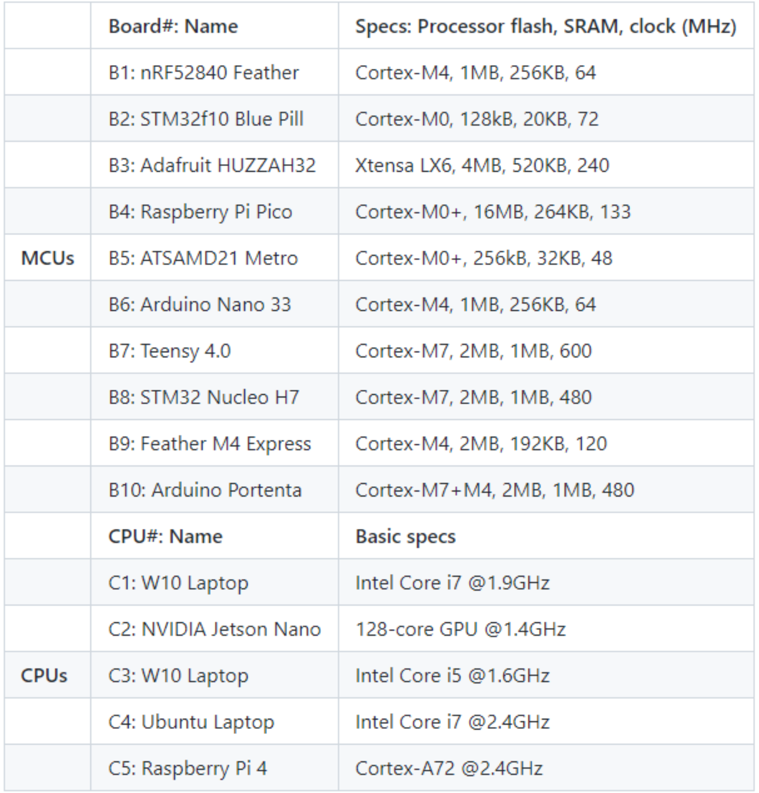
bharathsudharsan / ML-Model-Combining Public

bharath sudharsan Update README.md 6099811 28 minutes ago 66 commits

Average_Maximization_Voting_Media...	added code	2 months ago
DCS-LA.py	added full code	2 months ago
DES-LA.py	added full code	2 months ago
Medical_data_usecase.png	Update Medical_data_usecase.png	2 months ago
Performance_of_combined_models.p...	Update Performance_of_combined_models.png	2 months ago
README.md	Update README.md	28 minutes ago
Requirements.txt	added req.txt	2 months ago
Stacking.py	added full code	2 months ago

Repo: https://github.com/bharathsudharsan/ML-Model-Combining

Experiments Setup



	Board#: Name	Specs: Processor flash, SRAM, clock (MHz)
	B1: nRF52840 Feather	Cortex-M4, 1MB, 256KB, 64
	B2: STM32f10 Blue Pill	Cortex-M0, 128kB, 20KB, 72
	B3: Adafruit HUZZAH32	Xtensa LX6, 4MB, 520KB, 240
	B4: Raspberry Pi Pico	Cortex-M0+, 16MB, 264KB, 133
MCUs	B5: ATSAMD21 Metro	Cortex-M0+, 256kB, 32KB, 48
	B6: Arduino Nano 33	Cortex-M4, 1MB, 256KB, 64
	B7: Teensy 4.0	Cortex-M7, 2MB, 1MB, 600
	B8: STM32 Nucleo H7	Cortex-M7, 2MB, 1MB, 480
	B9: Feather M4 Express	Cortex-M4, 2MB, 192KB, 120
	B10: Arduino Portenta	Cortex-M7+M4, 2MB, 1MB, 480
	CPU#: Name	**Basic specs**
	C1: W10 Laptop	Intel Core i7 @1.9GHz
	C2: NVIDIA Jetson Nano	128-core GPU @1.4GHz
CPUs	C3: W10 Laptop	Intel Core i5 @1.6GHz
	C4: Ubuntu Laptop	Intel Core i7 @2.4GHz
	C5: Raspberry Pi 4	Cortex-A72 @2.4GHz

- **Devices.** Distributed, ubiquitous IoT Devices in the real world have heterogeneous hardware specifications. To replicate this, the devices chosen to carry out the distributed training contains 10 resource-constrained MCU boards (B1-B10) and 5 CPU devices (C1-C5)

- **Datasets.** Below datasets are used for training on the selected MCUs and CPUs

  - Banknote Authentication (5 features, 2 classes, 1372 samples)
  - Haberman's Survival (3 features, 2 classes, 306 samples)
  - Titanic (11 features, 2 classes, 1300 samples)

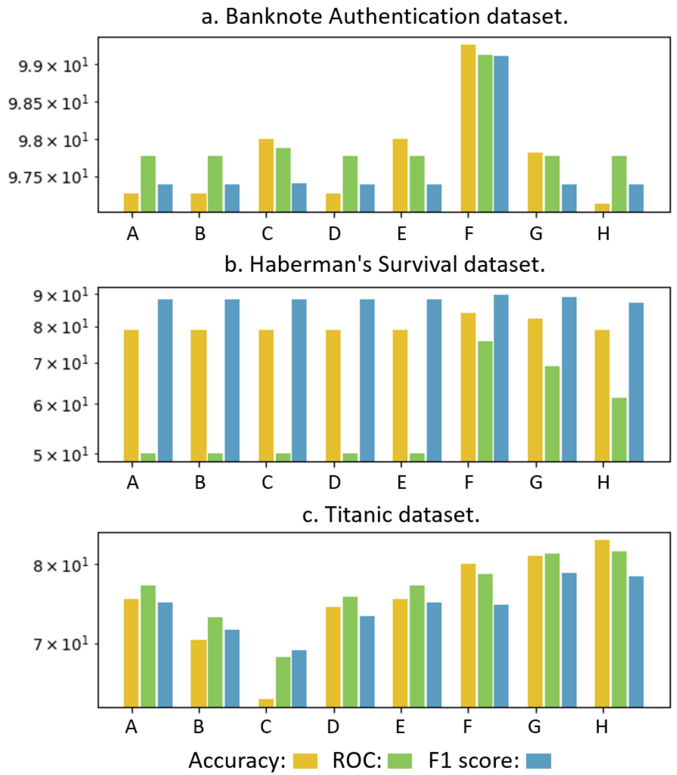
Experiments Procedure

Confirm Smart Manufacturing

- The training process on all 15 devices is carried out using the resource-friendly classifier training algorithm from Machine Learning - Microcontroller Unit (ML-MCU)
  - IoT Journal Paper: https://ieeexplore.ieee.org/document/9490288
  - Code: https://github.com/bharathsudharsan/ML-MCU
- Initially, for the Banknote dataset, upon all devices completing the training, 15 base models are obtained (first set). Then, each of the 8 ML model combining methods are one by one applied on this first set of models, producing 8 central models (one central model as an output of each combining method)
- A similar procedure was followed for the remaining datasets, producing the second and third set of models, followed by model combining. At this stage, there are 8 central models for each dataset, whose performance is evaluated in terms of Accuracy, ROC, and F1 score (F1) metrics

Results

Confirm
Smart Manufacturing



A: Simple Avg, B: Weighted Avg, C: Maximization, D: Weighted Majority Vote, E: Combine by Median, F: Dynamic Classifier Selection, G: Dynamic Ensemble Selection, H: Stacking (Meta Ensembling).

- Performance analysis of the combined central models

  - **Banknote dataset.** Highest performance is shown by *DCS-LA (F)*. Followed by *Maximization (C)*, then *Median (E)* method. Combine by *Stacking (H)* is the least performing, followed by *DES (G)*

  - **Haberman's dataset.** Again, *DCS-LA (F)* showed the top performance. The *DES (G)* and *Stacking (H)* methods that produced a low performance for the previous dataset are the second and third best-performing methods

  - **Titanic dataset.** *Stacking (H)* shows the highest accuracy, but *DES (G)* achieved slightly higher ROC and F1 so, *DES (G)* is the overall top-performing method. Unlike in previous datasets, here, the combine by *Maximization (C)* and *Median (E)*, *Averaging (A)*, and *Voting (D)* methods show varying performance

Discussion

Confirm
Smart Manufacturing

- The following observation was made during experimentation:
  - The computational cost for creating an ensemble is not much higher than training a single base model. It is because multiple versions of the base model need to be generated during parameter tuning. Also, the computational cost for combining multiple IoT devices trained base models was small due to the simplicity of the presented combination strategies
  - To construct a good ensemble, it is recommended to create base models **as accurate and as diverse as possible**
  - Creating a learning algorithm that is consistently better than others is a hopeless daydream. i.e., from results *Stacking* shows **top** performance for the Titanic dataset and **least** in the Banknote dataset

Conclusion

Confirm
Smart Manufacturing

- With strict privacy regulations, the historic datasets building process is rapidly transforming into historic intelligence building - achieved by distributed learning on the IoT devices, then central model combining

- In this work, 8 robust ML model combining methods were implemented and extensively tested, whose open-sourced implementation, we believe to provide the basis for a broader spectrum of decentralized and collective learning applications

Confirm
Smart Manufacturing

Confirm
Smart Manufacturing

Contact: Bharath Sudharsan
Email: bharath.sudharsan@insight-centre.org

www.confirm.ie

Science Foundation Ireland For what's next