ACONA: Active Online Model Adaptation for Predicting Continuous Integration Build Failures

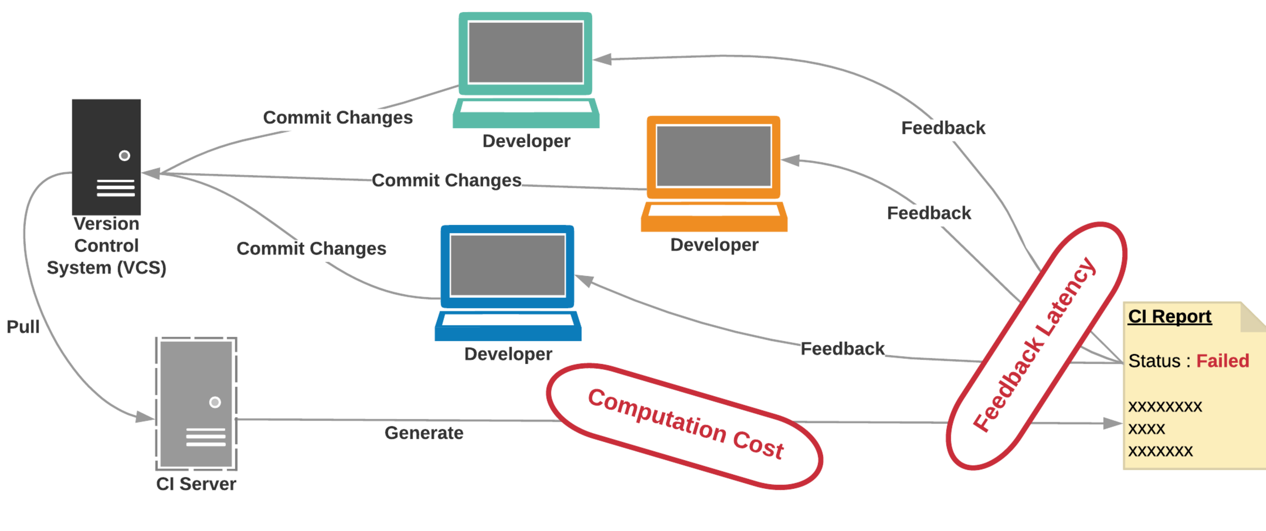
Ansong Ni, Ming Li

National Key Laboratory for Novel Software Technology, Nanjing University, China

niansong1996@gmail.com, lim@nju.edu.cn

Motivation

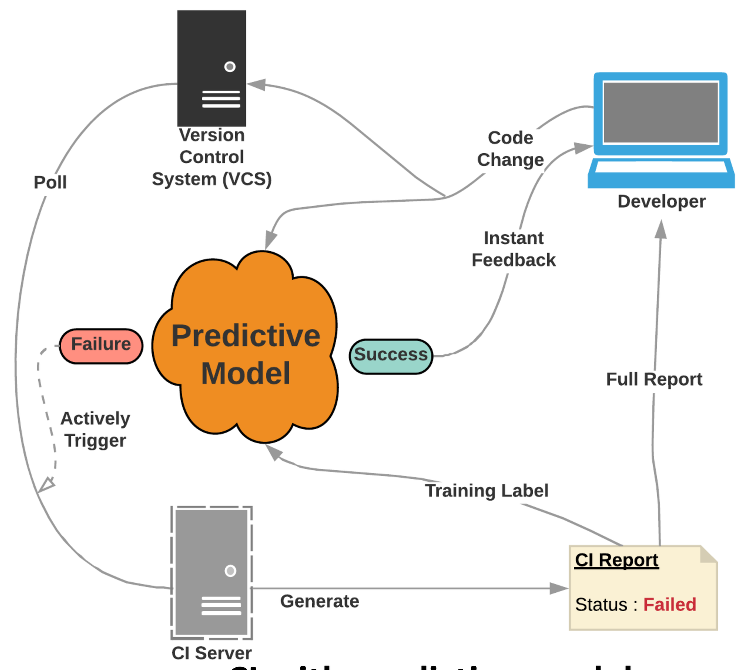
Reducing the cost of CI



A typical workflow of CI

Observation1: Despite the great benefit of CI, it consumes great computation resources and causes great latency in team collaboration.

Observation2: A majority of the builds turned out to be clean, which provides no information on potential defects.



CI with predictive model

If predicted as SUCCESS, developers will directly receive this predictive feedback thus their work is not interrupted. We can either delay this build and priotize more buggy ones, or simply mark it as clean and skip it.

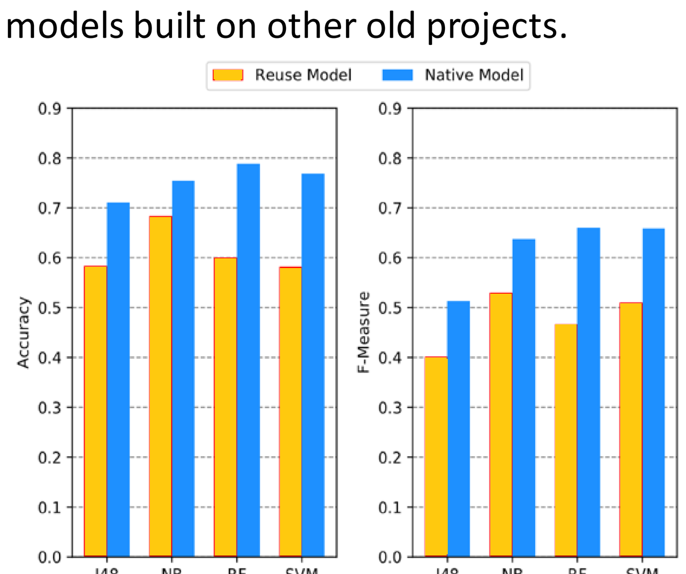
If predicted as FAILURE, a real CI build is requested to generate a full report for locating potential defects. The real CI result will also be fed to the model as training labels.

Developers receive instant feedback, improving the efficiency of the group. Computation resources are saved due to reduced amount of builds and more reasonable dispatching policy.

The challenges in CI outcome prediction

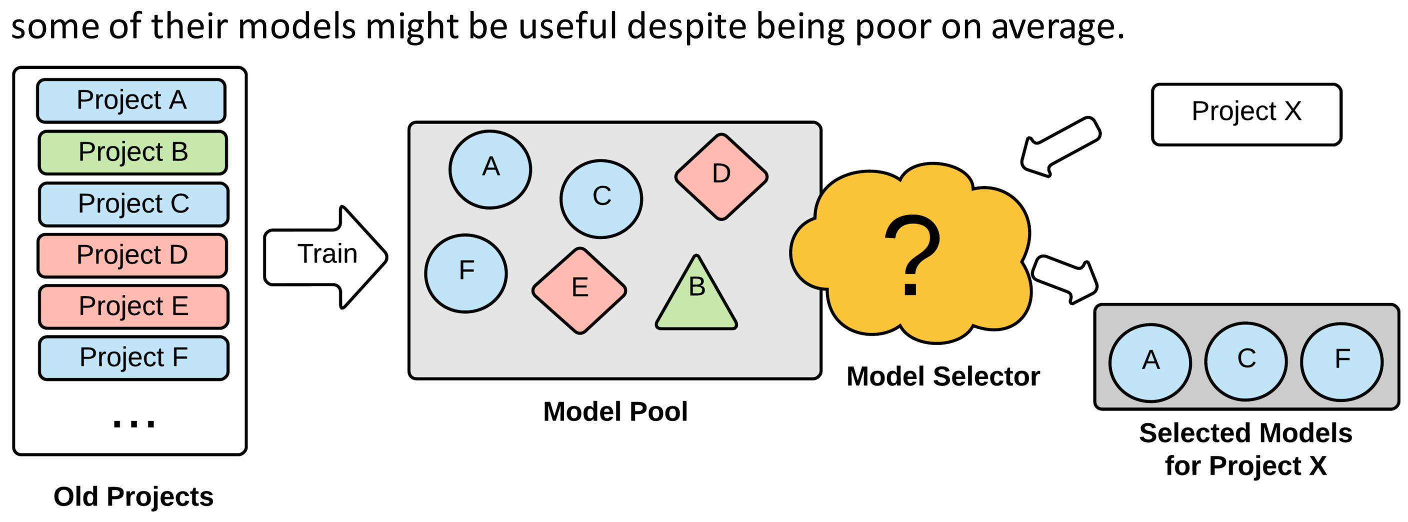
Challenge1: The build data from a project is almost always not enough for training an effective classifier. Especially for newly started projects, which suffer from a severe "cold start" problem.

Possible Solution: Directly apply existing models built on other old projects.



Result: ✗ (Poor Performance)

Observation: A new project may share some characteristics with old projects thus some of their models might be useful despite being poor on average.



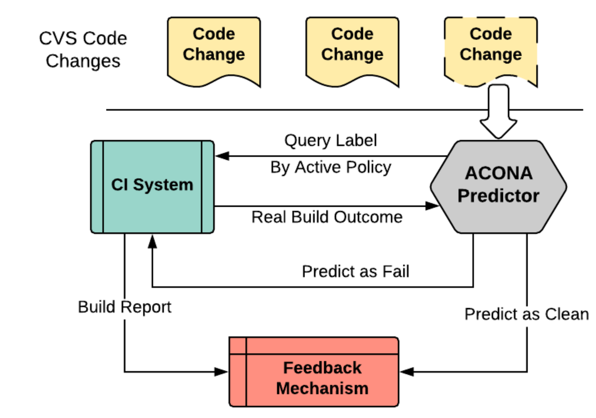
Solution1: We utilize a pool of models built on other old projects and we try to select some models for a new project. We do this by learning a combination weight, a simple but very effective approach.

Challenge2: The code changes are streaming data, which requires re-training after every single change is staged.

Solution2: We solve this with **online learning** of the combination weight. With the project evolving, the combination weight is repeatedly while the models in the pool stay unchanged.

Challenge3: Ture label for updating and training the classifier can only be obtained by executing real CI builds. To reduce the total cost, the amount of labels should be minimized.

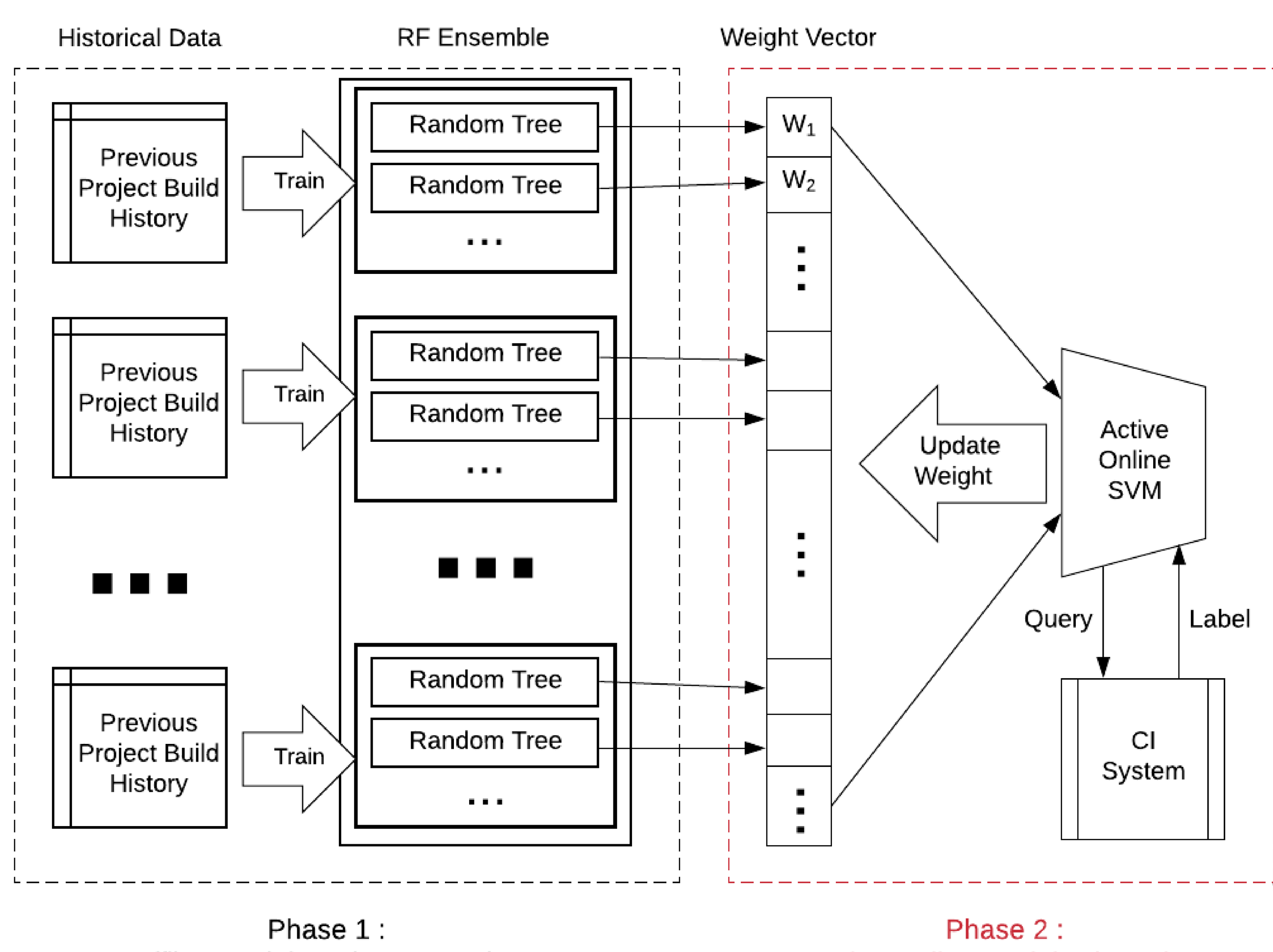
Solution3: We leverage **active learning** to address this problem. With active learning policy, ACONA can minimize the demand for labeled data by selecting the most valuable build tasks to query the real build outcome for update.



An illustration of active online learning

Method

General framework of our method



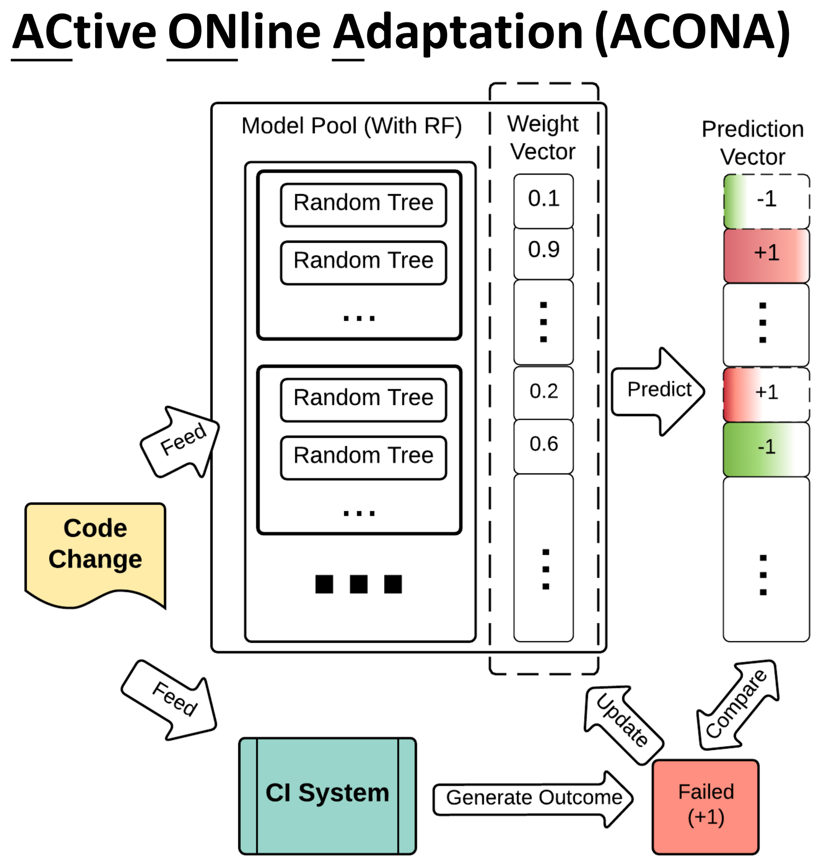
Phase 1 : Offline Model Pool Construction

Phase 2 : Active Online Model Adaptation

For each old project, a random forest, each with several random trees (C4.5 decision trees with random features) are trained with its build history. Then we ensemble all the random forests trained on different projects using bagging to form the model pool.

For each CI build, all RT in the model pool is required to give its prediction. Multiplied by the weight vector, we get the prediction given by the pool. If the label is queried by the active policy, it is used to update the weight vector.

<u>AC</u>tive <u>ON</u>line <u>A</u>daptation (ACONA)



An example of online adaptation

Online Model Pool Adaptation:

In order to learn the optimal combination weight of the model pool, we employ a soft margin SVM. Consider a sequence of data $\{\{x^{(1)}, y^{(1)}\}, \{x^{(1)}, y^{(1)}\}, \ldots, \}$ of length T, the objective is to minimize the average cost of each time step as follows:

$$\min_{\mathbf{w}^{(1)},\ldots,\mathbf{w}^{(T)}} \frac{1}{T}\sum_{t=1}^{T} L(\mathbf{w}^{(t)}, \mathbf{x}^{(t)}, y^{(t)})$$

where

$$L(\mathbf{w}^{(t)}, \mathbf{x}^{(t)}, y^{(t)}) = C \cdot \|\mathbf{w}^{(t)}\|_2^2 + \max(0, 1 - y^{(t)} \cdot \mathbf{w}^{(t)\top}\hat{\mathbf{y}}^{(t)})$$

The update rule of the weight vector is as follows:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta_t \cdot \nabla L(\mathbf{w}^{(t)}, \mathbf{x}^{(t)}, y^{(t)})_{\mathbf{w}}$$

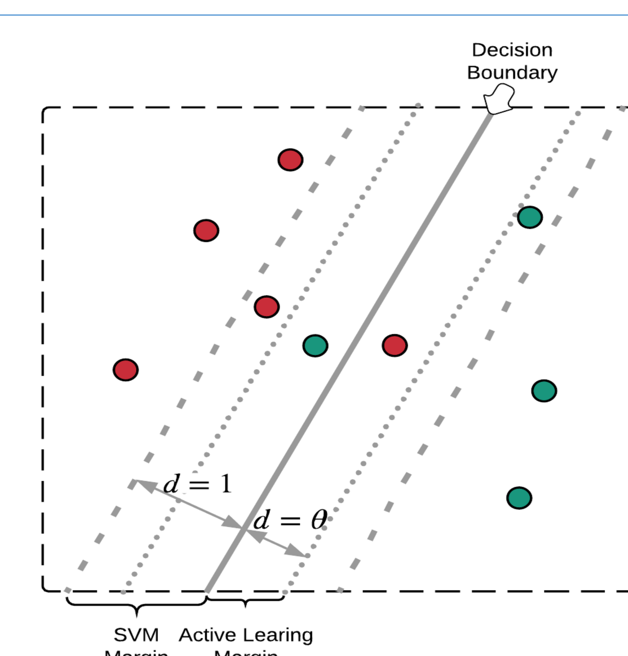
It is proved that if we set $\eta_t = \frac{1}{\sqrt{t}}$, the online optimization will converge to the optimal "offline" solution.

Active Learning Policy:

In order to update the weight vector, the ground truth for the label is required. To minimize the amount of labels that ACONA requires, we leverage active learning in an online setting. In each time step, we calculate the distance of the prediction vector to the decision hyperplane as follows:

$$d(\hat{\mathbf{y}}^{(t)}, \mathbf{w}^{(t)}) = \frac{\mathbf{w}^{(t)\top}\hat{\mathbf{y}}^{(t)}}{\|\mathbf{w}^{(t)}\|_2}$$

Then we adopt a threshold $\theta \in (0,1)$, the builds that lies within the "active learning margin" will be fed to CI system and perform a real CI build to query for true label.



An illustration of active learning

Experiments

Experiment settings

Dataset: We synthesized the data from two sources, *TravisTorrent* and *GitHub.* We select the projects with over 1,000 LOC and 200 builds which results in 534 projects with a total amount of 365,766 builds.

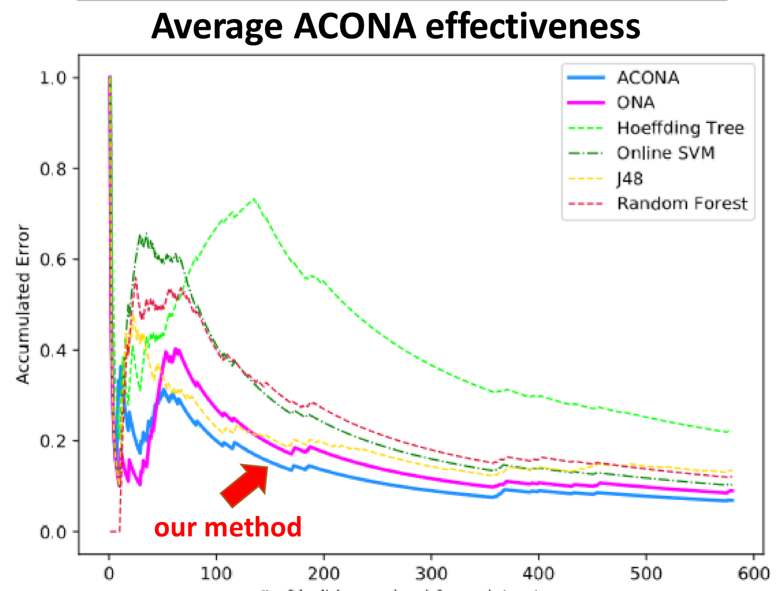
Evaluation: We use the following two metrics:

- *F-Measure:* As the class of failed builds is critical in CI outcome prediction, we focus on the F-Measure of it.
- *Accumulated Error:* With an online setting, it is a widely used metric for evaluation along the time. It is the average error rate of all previous predictions.
- *Query Rate:* Since querying real label from CI system brings cost, query rate is used to evaluate the percentage of labeled data a model demands for learning.

Model pool adaptation effectiveness

Time Step	F-Measure ↑		Ac. Error ↓	
	Value	Imp. %	Value	Imp. %
Before ACONA	0.310	-	0.492	-
After 50 Builds	0.360	16.1%	0.173	-64.8%
After 100 Builds	0.395	27.4%	0.172	-65.0%
After 200 Builds	0.418	34.8%	0.184	-62.6%
After All Builds	0.434	40.0%	0.181	-63.2%

Average ACONA effectiveness



Accumulated error with project progress

Impact of active learning

Query Quota	Evaluation Metrics	Active Selection	Random Selection					Greedy Selection				
		ACONA	ONA	HT	O-SVM	J48	RF	ONA	HT	O-SVM	J48	RF
20	F-Measure ↑	**0.419**	<u>0.373</u>	0.293	0.164	0.327	0.363	<u>0.382</u>	0.258	0.157	0.242	0.273
	Ac. Error ↓	**0.168**	<u>0.170</u>	0.363	0.296	0.300	0.321	<u>0.254</u>	0.452	0.366	0.370	0.423
50	F-Measure ↑	**0.432**	<u>0.411</u>	0.337	0.169	0.337	0.374	<u>0.388</u>	0.315	0.150	0.278	0.316
	Ac. Error ↓	0.181	<u>**0.180**</u>	0.416	0.290	0.291	0.314	<u>0.311</u>	0.434	0.331	0.325	0.345
100	F-Measure ↑	**0.432**	<u>0.405</u>	0.366	0.176	0.350	0.384	<u>0.380</u>	0.355	0.167	0.320	0.354
	Ac. Error ↓	**0.205**	<u>0.264</u>	0.413	0.287	0.286	0.306	0.378	0.425	0.310	<u>0.305</u>	0.324

Performance of different selection heuristics with fixed query quota

Conclusion:

1. ACONA utilizes a pool of well-trained models built on old projects, which solves the "cold start" problem for newly started projects.
2. It only learns a combination weight of the model pool and with active learning, it selects the most valuable builds to query the CI system for update which greatly reduces its total cost.
3. ACONA boosts *F-Measure* by 40.0% and reduced *Accumulated Error* by 63.2%. After adaptation, it outperforms all previously used methods.