A Machine Learning–based Framework for Building Application Failure Prediction Models

Tanjila Ahmed

UCDAVIS
UNIVERSITY OF CALIFORNIA

Outline

- Objective
- Motivation
- Why $\mathrm{F}^2\mathrm{PM}$?
- $\mathrm{F}^2\mathrm{PM}$ Framework
- Steps to Implement $\mathrm{F}^2\mathrm{PM}$
- Experimental Setup
- Results
- Conclusion

References

1. A. Pellegrini, P. D. Sanzo, and D. R. Avresky, “A Machine Learning-based Framework for Building Application Failure Prediction Models” , Parallel and Distributed Processing Symposium Workshop (IPDPSW), May 2015.

Objective

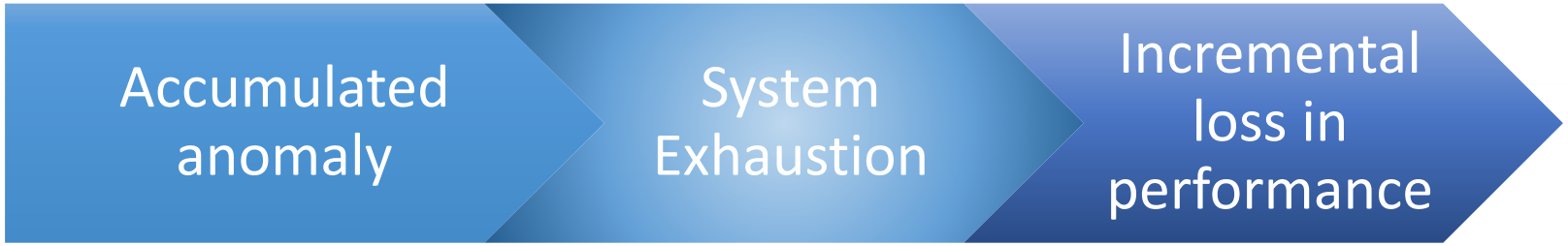
Framework for building Failure Prediction Models (F^2PM), a Machine Learning-based Framework to build models for predicting the **Remaining Time to Failure** (RTTF) of applications in the presence of software anomalies.

Features:

- Creates a knowledge base upon no of features.
- Application independent.
- Performs a feature selection to identify best features.
- Generated models can be compared using set of metrics produced.
- Experimental results of successful application of the model.

Motivation

- Anomalies: memory leaks, unterminated threads, unreleased locks, file fragmentation.



- Proactive Rejuvenation which preventively force the application or hosting system to a clean slate before predicted crash.

F^2PM

A framework, which is able to autonomously derive a set of different prediction models, enabling user to select the best-suited one.

- Operates in a non-intrusive way
- Exploits only system level features
- Sufficient no of observations are collected in advance of the monitored phenomena
- No of system features are monitored and their values are recorded while the application responsible for anomalies run
- When the user defined condition for failure is met, $\mathrm{F}^2\mathrm{PM}$ logs the occurrence time & system is restarted
- Collected data are used for building and validating a number of models generated by using different ML algorithms
- Uses: VM and cloud computing

F^2PM Framework

F2PM's Goal : Build optimized ML models for failure prediction

Input : Selected system feature

Condition: Failure conditions set by user

Output: RTTF

Steps to implement F^2PM:

1. Initial System Monitoring
2. Data-point aggregation and added metrics
3. Features Selection
4. Model Generation and Validation

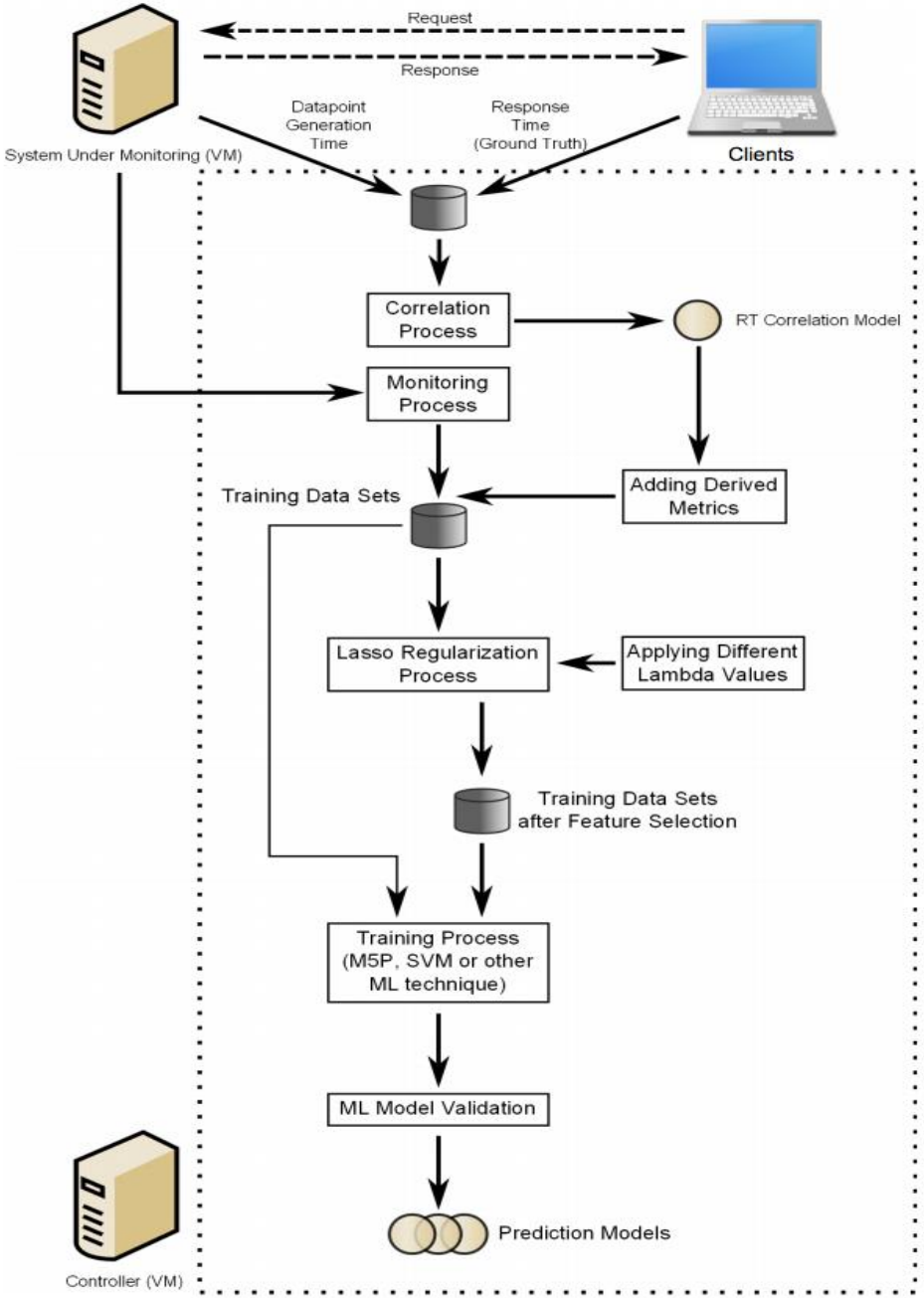


Fig. 1. F^2PM Architecture

Steps to Implement F^2PM

1. Initial system monitoring:

- this phase consists of collecting measurements of a no of system features while system runs application generating anomalies.
- Every time system failure condition is met, a fail event is added to the data history and system is restarted.
- This gives rise to a number of runs of system. Particularly, a given amount of data, which would be sufficient to build ML models with a given accuracy, has to be collected.
- Size of the dataset to be collected in this phase can be determined via the set of metrics that allow the user to evaluate the accuracy of the produced models
- If estimated accuracy is not sufficient, further system runs can be executed to collect new data into the training set, and to produce new models.

Steps to Implement F^2PM

1. Initial system monitoring:
Listed features are selected because, on basis of them, measure effect on system of kind of anomalies affecting application that we are studying (i.e. memory leaks and unterminated threads).

Output of this phase includes a set of row data representing the evolution of the system feature along a number of system runs.

T_{gen}	is the timestamp denoting the elapsed time since the system has started;
n_{th}	is the number of active threads in the system;
M_{used}	is the amount of memory used by applications running in the system;
M_{free}	is the amount of memory freely available for usage by applications;
M_{shared}	is the amount of memory used for buffers shared by applications;
M_{buff}	is the amount of memory used by the underlying operating system to buffer data;
M_{cached}	is the amount of memory used to cache disk data;
SW_{used}	is the amount of swap space, which is currently used;
SW_{free}	is the amount of swap space, which is currently free;
CPU_{us}	is the percentage of CPU time dedicated to userspace processes;
CPU_{ni}	is the percentage of CPU time occupied by user-level processes with a positive nice value (lower scheduling priority);
CPU_{sys}	is the percentage of CPU time spent in kernel mode;
CPU_{iow}	is the percentage of CPU time spent waiting for a I/O operations to complete;
CPU_{st}	is the percentage of time a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor;
CPU_{id}	is the percentage of CPU time spent doing un-fruitful work (i.e., the system is underloaded).

Steps to Implement F^2PM

2. Data-point aggregation and added metrics:

1. Aggregated data points are generated on the basis of a user-defined time interval.
2. Each input data point (shown in black in the figure) is placed, on the basis of the value of a feature, on the time axis.
3. All data points falling in the same time interval are used to generate one aggregated data point.

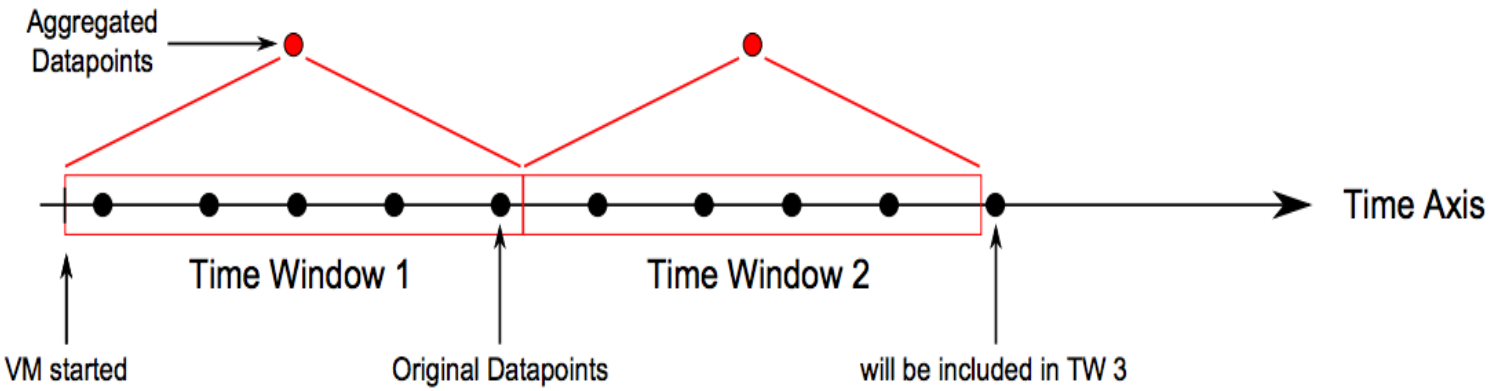


Fig. 2. Datapoint Aggregation

Steps to Implement F^2PM

2. Data-point aggregation and added metrics:

- adding some metrics to each the aggregated datapoint. Specifically, for each system feature j, the slope is calculated according to the following formula:

$$slope_j = \frac{x_j^{end} - x_j^{start}}{n},$$

- where x_j^{start} and x_j^{end} are the values of the feature j of the first and the last original datapoint falling in the time interval
- If system crashes due to memory exhaustion, SWused will start growing faster when approaching crash point. Therefore, slope can be used effectively to build the prediction model.

Steps to Implement F^2PM

2. Feature Selection :

Identifying those features having (incrementally) more impact (weight) in prediction of the RTTF.

In statistics & machine learning, lasso is a regression analysis method that performs both variable selection and regularization in order to enhance prediction accuracy.

$$\frac{1}{n}\sum_{j=1}^{n} V(y_j, \langle \beta, x_j \rangle) + \lambda ||\beta||_1$$

n is the number of data points from the aggregation step, xj is a vector of values of input features (independent variables) of each data point, yj is the associated value of the dependent variable (RTTF) for the specific data point

However, this is an optional step.

UCDAVIS

Steps to Implement F^2PM

4. Model Generation and Validation

This phase aims at generating and validating a set of prediction models, which are built by using the training sets produced in the previous phases.

a. Linear Regression
b. M5P
c. REP-Tree
d. Lasso as a Predictor
e. Support-Vector Machine
f. Least-Square Support-Vector Machine

Steps to Implement F^2PM

For each model, the following metrics are provided:

1. **Mean Absolute Prediction Error (MAE):** it is the average of the differences between predicted and real RTTF.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|,$$

where fi is predicted value, yi is observed value, and n is number of samples in the validation set.

2. Relative Absolute Prediction Error (RAE): RAE normalizes total absolute error by dividing it by total absolute error of the simple predictor.

$$\mathrm{RAE} = \frac{\sum_{i=1}^{n}|f_i - y_i|}{\sum_{i=1}^{n}|Y - y_i|}, \qquad Y = \frac{1}{n}\sum_{i=1}^{n}|y_i|.$$

Steps to Implement F^2PM

- **Maximum Absolute Prediction Error (MAE):** it is the maximum prediction error, i.e. the maximum value in the set |fi − yi | for each sample i in the validation set.
- **Soft-Mean Absolute Prediction Error (S- MAE):** it is calculated as the MAE, except that when the value |fi − yi | is less a given threshold it is considered to be equal to zero.
- **Training Time:** it is the time required by the learning method for building the model.
- **Validation Time:** it is the time required for completing the validation of the model, including the calculation of the above mentioned errors.

The above metrics provide the user with useful information for comparing the different models produced by F^2PM.

Experimental Setup

- A controlled experiment on a virtual architecture was carried out, which was built on top of a **32-core HP ProLiant NUMA server**. The server is equipped with a **Debian GNU/Linux distribution** (kernel version 2.6.32-5-amd64). **VMware Workstation 10.0.4** is the virtual environment hypervisor. All virtual machines of the experimental environment were equipped with **Ubuntu 10.04 Linux Distribution** (kernel version 2.6.32-5-amd64).
- 2 different virtual machines (VM) were used. One VM runs our FMS (to collect the hardware features), and generates the workload targeting the second VM. The second VM hosts the application, experiencing occurrence of anomalies.
- Multi-tier e-commerce web application that simulates a on-line book store, following the standard configuration of TPC-W benchmark was tested

Experimental Setup

- The experiment was continuously run for one week, having an emulated browsers continuously issue requests to the TPC-W server. Upon a crash, VM hosting the TPC-W is automatically restarted, so as to start serving again requests by emulated browsers as soon as possible.

RESULTS

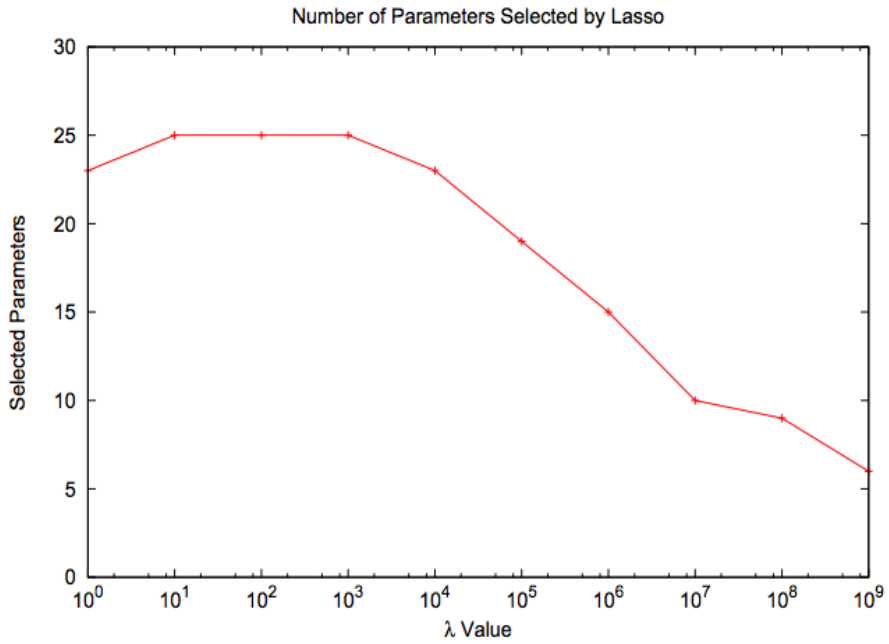


Fig. 4. Parameters selected by Lasso

higher values of λ are generally associated with a smaller number of features selected by Lasso (namely, Lasso associates a higher number of features with a zero weight in the β vector).

RESULTS

- In order to evaluate the accuracy of prediction models, we can see that the best accuracy is provided by REP-Tree. In comparison with REP-Tree, M5P increases the error in order of 10%. All other ML methods show higher errors. We note that this could be due to the fact that both REP-Tree and M5P divide the model space in smaller portions, and evaluate for each portion a different linear approximation.

TABLE II. SOFT MEAN ABSOLUTE ERROR—10% THRESHOLD

Using all parameters		Using only parameters selected by Lasso	
Algorithm	**Error (seconds)**	**Algorithm**	**Error (seconds)**
Linear Regression	137.600	Linear Regression	156.603
M5P	79.182	M5P	118.292
REP Tree	69.832	REP Tree	108.476
SVM	132.668	SVM	146.594
SVM2	132.675	SVM2	146.607
Lasso ($\lambda = 10^0$)	405.187	Lasso ($\lambda = 10^0$)	405.187
Lasso ($\lambda = 10^1$)	405.187	Lasso ($\lambda = 10^1$)	405.187
Lasso ($\lambda = 10^2$)	405.186	Lasso ($\lambda = 10^2$)	405.186
Lasso ($\lambda = 10^3$)	405.178	Lasso ($\lambda = 10^3$)	405.178
Lasso ($\lambda = 10^4$)	405.124	Lasso ($\lambda = 10^4$)	405.124
Lasso ($\lambda = 10^5$)	404.823	Lasso ($\lambda = 10^5$)	404.823
Lasso ($\lambda = 10^6$)	404.041	Lasso ($\lambda = 10^6$)	404.041
Lasso ($\lambda = 10^7$)	399.023	Lasso ($\lambda = 10^7$)	399.023
Lasso ($\lambda = 10^8$)	399.240	Lasso ($\lambda = 10^8$)	399.240
Lasso ($\lambda = 10^9$)	392.469	Lasso ($\lambda = 10^9$)	392.469

UCDAVIS

RESULTS

- It is evident that when using all parameters training times are significantly higher. Based on presented results, user can make a choice between less time in training or having a higher accuracy of the prediction model. Similarly, as we can see in Table IV, more time is required for validating prediction models when all parameters are used.

TABLE III. TRAINING TIME

Using all parameters		Using only parameters selected by Lasso	
Algorithm	**Error (seconds)**	**Algorithm**	**Error (seconds)**
Linear Regression	0.30	Linear Regression	0.08
M5P	3.10	M5P	1.58
REP Tree	0.56	REP Tree	0.17
SVM	417.41	SVM	164.96
SVM2	391.69	SVM2	205.65

TABLE IV. VALIDATION TIME

Using all parameters		Using only parameters selected by Lasso	
Algorithm	**Error (seconds)**	**Algorithm**	**Error (seconds)**
Linear Regression	0.42	Linear Regression	0.12
M5P	0.36	M5P	0.09
REP Tree	0.55	REP Tree	0.11
SVM	0.39	SVM	0.13
SVM2	0.38	SVM2	0.13

Conclusion

- One advantage of this approach is that $\mathrm{F}^2\mathrm{PM}$ can be used out of the box, without any need for manual modification/intervention in applications.
- it can be customized by user according to a specific class of application and/or type of anomalies.
- $\mathrm{F}^2\mathrm{PM}$ uses different machine-learning methods to generate models, allowing users to decide, on basis of a set of metrics, the best suited one for his needs.
- $\mathrm{F}^2\mathrm{PM}$ allows us to select prediction models for application failure, with small training time and high accuracy.

UCDAVIS