# Index terms: Machine learning, Process Performance, Operating System

# 0. Abstract

This work presents a study on using machine learning techniques to predict the performance of processes in operating systems. In this study, machine learning-based approach uses various process attributes such as CPU usage, memory usage, and I/O operations as input features to train and evaluate machine learning models. Various machine learning algorithms are experimented, including linear regression and random forests, to identify the most suitable approach for predicting process performance. The performance of approach is evaluated by using a real-world dataset collected from Linux operating system and macOS operating system, and the results show that machine learning-based methods can accurately predict process performance. The findings demonstrate the potential of machine learning in improving the performance and efficiency of operating systems. Overall, this work provides a valuable contribution to the field of operating systems by showcasing the potential of machine learning techniques for predicting process performance. The proposed approach can help system administrators and developers optimize resource usage and improve the overall performance and reliability of the system.

# 1. Introduction

The process performance of an operating system is a critical aspect that determines the overall efficiency and responsiveness of the system. Processes are essentially the tasks or programs that are executed by the operating system, and their performance can have a significant impact on the overall performance of the system. The operating system needs to efficiently manage the allocation of resources such as CPU time, memory, and I/O operations to ensure that each process runs smoothly without causing delays or crashes. Therefore, predicting the performance of processes can help achieve to provide a more efficient system performance and help complete all programs efficiently.

The prediction of process performance can bring several benefits including early detection of performance issues, improved resource utilization, better capacity planning, reduced downtime, improved user experience, cost savings, etc. To make the prediction happen, monitoring various process metrics needs to be involved, such as CPU utilization, memory usage, I/O throughput, and response time, and adjusting the system configuration as needed to optimize performance.

On any operating system, the user can check process information in terminal by some build-in command lines. The information usually includes each task’s unique process ID, username of owner of each task, priority of each task, total virtual memory used by each task, amount of actual physical memory each process is consuming, shared Memory size in kilobytes unit used by each task, the state each process is in, CPU usage for each process, memory usage of each task, CPU time, command that is being run, etc. By leveraging the vast amounts of data with multiple attributes like those, machine learning algorithms can identify patterns and make predictions about performance of process.

In summary, in this work machine learning techniques are applied to learn the performance and behavior of processes in operating systems. The rest of work is organized as follows. In Section ?, the brief approach to collect data and the predictive machine learning models are introduced. Next, the details for data collecting and model building are shown in Section?. Then, the evaluation of machine learning techniques is represented in Section?. Finally, a conclusion is given for the work in Section ?.

# 2. Related Work

There are some works done related to predict the performance of process.

~~There have been numerous works done in the area of design space exploration. Eyerman et al. [9] uses a different heuristics to model the shape of the design space of superscalar out-of-order processor. Ipek et al. [2] use artificial neural networks (with cross-validation to calculate their prediction accuracy) to predict the performance of memory, processor and CMP design spaces. Meanwhile, Lee et al. [3] use regression models to predict performance and power usage of the applications found in the SPECjbb and SPEC2000 benchmarks. As in the previous reference, the data points are created using simulations. Kahn et al. [15] uses predictive modeling, a machine learning technique to tackle the problem of accurately predicting the behavior of unseen configurations in CMP environment. Ghosh et al. [10] have presented an analytical approach to the design space exploration of caches that avoids exhaustive simulation. The problem that they are trying to solve (only varying cache size and associativity) is very small compared to the ones that other researches are trying to solve. Dubach et al. [16] has used a combination of linear regressor models in conjunction with neural networks to a model that can predict the performance of programs on any microarchitectural configuration with only using 32 further simulations. In this work, we target system performance rather than processor performance. All of these works have based their models on simulation while our results use simulation results and already built existing computer systems. To our knowledge there has not been any work done in this area. The closest work is by Ipek et al. [11], where they use artificial neural networks to predict the performance of SMG2000 applications run on multi-processor systems. The application inputs and the number of processors the application runs on are changed during their analysis. Their accuracy results are around 12.3% when they have 250 data points for training. However, we must point that they also do not estimate the performance of the systems but rather simulate the execution of an application on one system.~~

# 3. Design

The project is divided into couple steps. There is data collecting on Linux operating system and macOS operating system, data preprocessing, machine learning model building for prediction, and test cases to verify the accuracy of prediction.

## 3.1 Data Collecting

Data collecting is an essential part of in this project. Process-related data are required to be collected from various operating systems. In this work, Linux operating system and macOS operating system are chosen because both systems are common enough to generate real-world general data to represent themselves.

## 3.1.1 Linux Operating System

A kernel module, also known as a driver, is a software component that can be loaded or unloaded dynamically into the kernel of an operating system. It can interact with hardware components and provide services to user-space applications. The kernel module is written in a low-level programming language which is C language. It adheres kernel's programming interface (API) and driver model.

In this project, a kernel module is written, and it helps collect the data, which are generated by operating system, in virtual file on Linux operating system. In addition, a c file is executed in user level to export content of virtual file into actual file.

The reason to build a kernel module to collect process-related data on Linux operating system is that the kernel module can be loaded and unloaded dynamically without rebooting the operating system. This flexibility of managing system resources does not provide a hard work for kernel of Linux operating system to handle.

## 3.1.1 macOS Operating System

Although building a kernel module on macOS is possible, it is generally not recommended due to the tightly controlled kernel architecture, the proprietary nature of the kernel code, and the availability of high-level programming tools and libraries for building user-space applications. Thus, the kernel module is not loaded on macOS. Instead, a function in python is built to collect data on macOS operating system.

One of advantage of building Python on a macOS environment is that it is relatively easy because the macOS operating system comes with pre-installed development tools such as a C compiler and the XCode development environment. These tools are essential for building Python from source code, and their availability on macOS makes it straightforward to set up and build Python on a MacBook.

## 3.1.3 Data Preparation and Input Values

Once the data are collected into virtual file, preprocessing data is another important part of the predictive modeling. In the process of data exporting for Linux operating system, each row is checked and all whitespaces in the row are replaced by comma, which matches the format of csv and reduces the step to transfer export file from txt format to csv format. Besides, it is simpler for data on macOS operating system. Python is applied to drop unnecessary data and fill the missing data in rows if needed.

## 3.2 Predictive Models

In this project, based on data collected on Linux operating system and macOS operating system, predictive modeling techniques are applied to obtain estimates of performance of process in both operating system respectively. Couple predictive models are built. Firstly, predictive models are developed by using linear regression and random forest **regression**

to estimate the performance of processes in operating system. Secondly, the process performance is proved to be accurately predicted by using data collected from Linux operating system and macOS operating system.

I ~~use a total of nine models. The four linear regression models are described in the next section. Section 3.2 discusses the five neural network based models developed in this work.~~

## 3.2.1 Linear Regression (LR) Models

Linear regression is a basic and commonly used type of predictive analysis in machine learning algorithms. The reason that linear regression analysis is used to predict the value is that it is doable to split the variable into dependent variable which is the data to predict and independent variable which is the data using for prediction [?].

Suppose there are n data pairs defined by {(xi, yi), i = 1, ..., n}. The underlying relationship between yi and xi involving this error term εi by yi= α + β xi+εi [?]. This relationship between the true (but unobserved) underlying parameters α and β and the data points [?].

In the work, ordinary least squares linear Regression is applied. It fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation [?].

## 3.2.2 Random Forest Regression (RFR) Models

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time [?]. For regression tasks, the mean or average prediction of the individual trees is returned [?].

In the work, sklearn package and library from python is applied to train random forest regression model.

## 3.2.3 Error Handling

Some machine learning models do not provide the estimated predictive error for themselves. Thus, an evaluation metric needs to be introduced to ensure the performance of a predictive model provides accurate results. Typically, an evaluation metric requires to train the model on a dataset, use the model to make predictions on a holdout dataset which is not used during training, then compare results between predicted values and expected values in holdout database.

In this work, Mean Square Error (MSE) and Mean Absolute Error (MAE) are applied. Both evaluation metrics are similar. However, the difference between them is that MSE takes the square of average of between predicted and original values.

The main advantage to take MSE is that it is easier to calculate the gradient whereas in the case of mean absolute error it takes complicated programming tools to calculate the gradient. By taking the square of errors it pronounces larger errors more than smaller errors, larger errors can be focused more. The MSE is of the form [?]

The main advantage to take MAE is that it gives how we predict from the actual output. The MAE is of the form [?]

Cross-Validation

# 4. Evaluation

## 4.1 Experiment Setup

## 4.1.1 Data Collecting and Input Values on Linux Operating System

Proclog.c is the file built for kernel module. Couple kernel functions are added to help achieve to log information related to processes into a virtual file. The file needs to be compiled and couple files with same file name and different file types are generated in the process of compiling.

And it is required to load the kernel module on Linux operating system to start its work of logging data. The one file ending by .ko is the critical file used to install in Linux operating system. When it is loaded, it is added to the list of running kernel modules and no additional steps is required to execute kernel to collect process-related information. The user can be reminded by a msg if .ko file is loaded successfully.

What this kernel module does is create a virtual file called log\_file stored in /proc/ folder. log\_file is writable by kernel-level functions but not user-level applications. As Linux environment is on, log\_file is consistently updated by kernel module to log process-related data. So it can grow big in file size dramatically in a very short time. With help of its attribute as virtual file, it does not take significantly large space on hard disk.

As kernel modules are typically not designed to write actual files because they operate at a low-level in the operating system and are primarily used to interact with hardware components, system resources, and other kernel components, a user-level c file is helpful to write content of virtual file into actual file. And because it is not likely to export the comprehensive data from virtual data due to its consistent update by kernel module for data logging, an input step for user to manually input a positive integer is created to set up the number of lines to export. The kernel module does its work as the system is on. If the file is open for exporting, the read never reaches to the end and the program for data export never ends.

## 4.1.2 Data Collecting and Input Values on macOS Operating System

Python has its library called psutil to retrieve process-related data by couple lines of python code. To consistently collect data related to process, a input step is set up for user to choose interval and end time for data collection. The python function can retrieve process data every interval that user inputs and store the data in a file in csv format. The entire process for data collection can finish on time that user inputs.

## 4.1.3 Predictive Models

Python sklearn module is applied for training random forest regression model. RandomForestRegressor function is used. There are parameters which can be filled when the model is built. Some of them determine the number of decision trees to build, criterion used to determine model outcomes, maximum possible depth of each tree, the maximum number of features the model should consider when determining a split, etc.

## 4.1.4 Error Handling

## 4.2. Experiment Result

# 5. Challenges

However, every coin has two sides that building kernel module has various restrictions too. Debugging on kernel module is difficult because there are no detailed debug hits given by operating system to help developers fix issues.

# 6. Conclusion

In this work, two different machine learning techniques, linear regression and random forest regression, are applied to learn and estimate process performance and behavior, based on data from Linux operating system and from macOS operating system. The data used during the experiment are collected from both Linux operating system and macOS operating system.

# 7. Reference

[7] Montgomery D. C., Peck E. A. and Vining G. C. Introduction to Linear Regression Analysis. Wiley, New York, NY, 2001.

[?]https://en.wikipedia.org/wiki/Simple\_linear\_regression

[?]https://docs.kernel.org/index.html

[?]https://scikit-learn.org/stable/modules/linear\_model.html

[?]https://www.linuxjournal.com/article/8110

[?]http://tldp.org/LDP/lkmpg/2.6/html/index.html

[?]https://kodu.ut.ee/~dumas/pubs/WhiteBoxJSEP.pdf

[?]https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html

[?]https://en.wikipedia.org/wiki/Linear\_regression

[?]https://en.wikipedia.org/wiki/Random\_forest

[?]http://alumni.cs.ucr.edu/~kishore/papers/tencon.pdf

https://dev.to/arepp23/how-to-write-to-a-csv-file-in-c-1l5b

https://towardsdatascience.com/how-to-build-your-first-machine-learning-model-in-python-e70fd1907cdd

https://towardsdatascience.com/how-to-use-random-seeds-effectively-54a4cd855a79

https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

https://www.usenix.org/conference/nsdi21/presentation/fu

https://ieeexplore.ieee.org/document/4085157