Performance Prediction of Physical Computer Systems Using Simulation-Based Hardware Models & BREAST CANCER DETECTION

Dilip Valiya (201701458) Yogesh Prajapati (201701259) Bhavik Chhotala (201701217)

Assigned by:- Prof. Amit Mankodi

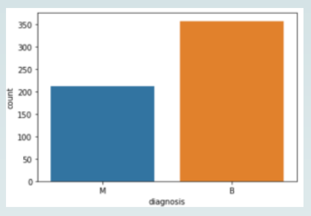
BREAST CANCER DETECTION

Introduction

Breast Cancer (BC) is a common cancer for women around the world, and early detection of BC can greatly improve prognosis and survival chances by promoting clinical treatment to patients early. So it's amazing to be able to possibly help save lives just by using data, python, and machine learning! So here I have shown methodology and results for the Breast cancer detection

Methodology

As this Project required I download data (csv file) from the Kaggle website (Source of all kinds of data). Read that csv file on my code. This csv file also consists of some redundant information, some NAN value as well as a value which are not in numeric form. So I have done some **pre-process** on this data to handle all the above cases. The below image shows the total number of Malignant(M) and Benign(B) cells in data.



Now in the **training phase** of my model I trained my model using below some Standard algorithms using SKlearn library.

I have used following models:
1. Logistic Regression 2. KNN 3. SVM(Linear) 4.SVM(RBF) 5.Gaussian Naïve bayes 6. Decision tree 7. Random forest

I get the accuracy of each model using The Confusion matrix. Below is an understanding of confusion matrix:

```
[[TN FP]
[FN TP]]
```

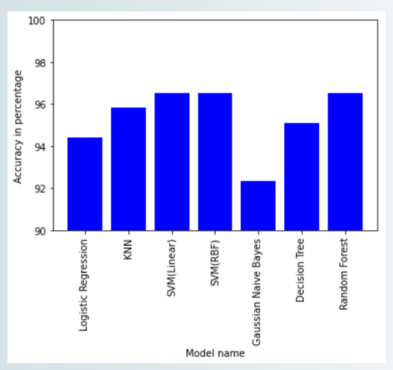
$$\text{Accuracy(\%)} = \frac{TN+TP}{TN+TP+FN+FP} * 100$$

To show the classification accuracy of each model in a different way I get accuracy_score from sklearn.metrics library and also print precision, recall, f1-score and support for each model.

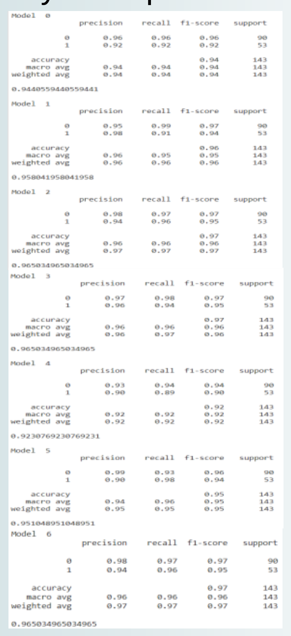
In the evaluation phase (Since I get the highest accuracy in a random forest) I print both predicted value using random forest and actual value.

Results

Below Image shows the accuracy of different models:

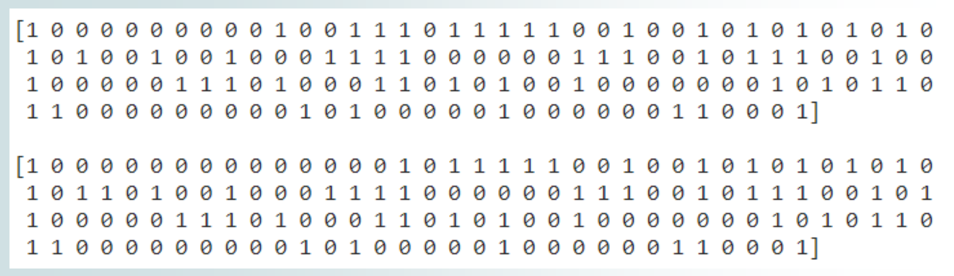


Here are another ways to represent accuracy:



```
Model  0
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        90
           1       0.92      0.92      0.92        53

    accuracy                           0.94       143
   macro avg       0.94      0.94      0.94       143
weighted avg       0.94      0.94      0.94       143

0.9440559440559441
Model  1
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        90
           1       0.98      0.91      0.94        53

    accuracy                           0.96       143
   macro avg       0.96      0.95      0.95       143
weighted avg       0.96      0.96      0.96       143

0.958041958041958
Model  2
              precision    recall  f1-score   support

           0       0.98      0.97      0.97        90
           1       0.94      0.96      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.97       143

0.965034965034965
Model  3
              precision    recall  f1-score   support

           0       0.97      0.98      0.97        90
           1       0.96      0.94      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.97      0.96       143

0.965034965034965
Model  4
              precision    recall  f1-score   support

           0       0.93      0.94      0.94        90
           1       0.90      0.89      0.90        53

    accuracy                           0.92       143
   macro avg       0.92      0.92      0.92       143
weighted avg       0.92      0.92      0.92       143

0.9230769230769231
Model  5
              precision    recall  f1-score   support

           0       0.99      0.93      0.96        90
           1       0.90      0.98      0.94        53

    accuracy                           0.95       143
   macro avg       0.94      0.96      0.95       143
weighted avg       0.95      0.95      0.95       143

0.951048951048951
Model  6
              precision    recall  f1-score   support

           0       0.98      0.97      0.97        90
           1       0.94      0.96      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.97       143

0.965034965034965
```

These are some Predicted v/s actual values(M = 1 & B = 0)



```
[1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1]

[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
 1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 1]
```

Performance Prediction of Physical Computer Systems Using Simulation-Based Hardware Models

Introduction

Nowadays we have several options for the selection of computer hardware systems. A software performance will differ on each hardware system according to its architectural features. We need to buy and measure the performance of a software on particular hardware which is an exhaustive task.

We can built several hardware systems on simulation tools and evaluate the performance of the software. Therefore, we need to use simulation-based hardware models to predict the performance of the physical hardware systems.

We want to build machine learning model on top of simulation model to evaluate **"Cross Performance Prediction"**. So that given hardware feature trained model can predict physical runtime for different algorithm. We have trained machine learning model on Gem5 simulation-based hardware models (475 models each with 9 feature) data of different benchmark algorithms (mser , svm , tracking , stitch ,dijkstra,sha).

We carried out work mention in the paper[2] and performed "Cross performance prediction".

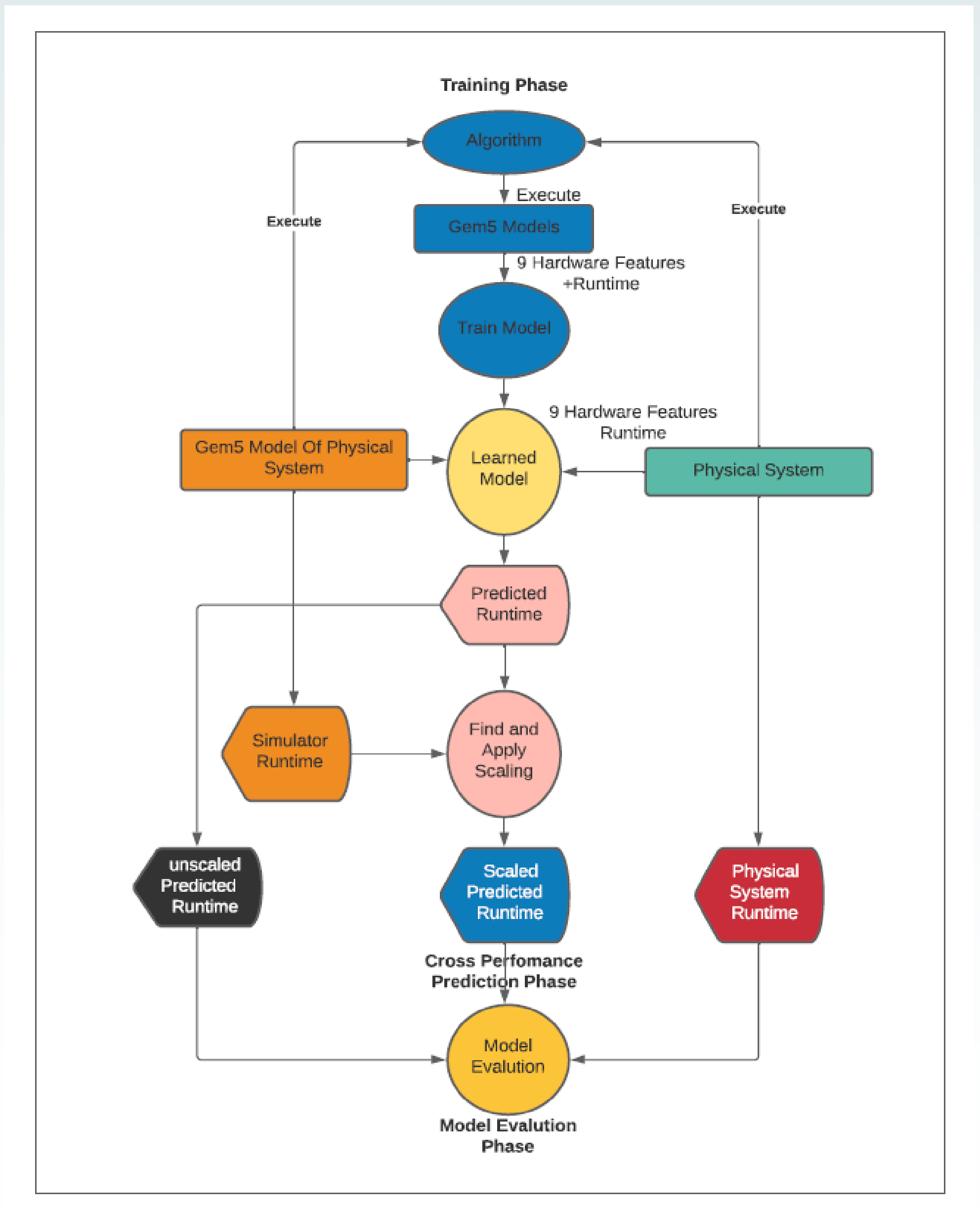
Methodology

Our cross performance prediction model is shown in Figure has three phases.

1. training phase
2. cross performance prediction phase
3. model evaluation phase.

In the **training phase**, the machine-learning regression model is trained for a given algorithm to learn the relationship between hardware model features and the actual simulator performance ("actual runtime") of the same algorithm. We select sample hardware models (M) for the training phase with feature values that represents the general population of hardware models available today denoted as $Xs_j, 1 \le j \le M$.We encode features with text from Xs_j into $\widehat{Xs_j} \in R^d$ where 'd' represents no. of features. We represent actual simulator runtime of selected (M) samples for a given algorithm as $ys_j \in R$ in the training phase of train model. The objective of the training phase is to find function $f(\widehat{Xs_j}) \approx ys_j \,\forall\, j$. The mapping between hardware features and actual simulator run time.

In the **cross performance prediction phase** first, we collect features Xp_i of each of the (N) physical systems $i, 1 \le i \le N$. We encode this data Xp_i into $\widehat{Xp_i} \in R^d$. This physical system features $\widehat{Xp_i}$ are provided as input to learned model to predict the run time y_{pred_i} on physical computer system using learned relationship from training phase $f(\widehat{Xp_i}) \approx y_{pred_i}$.



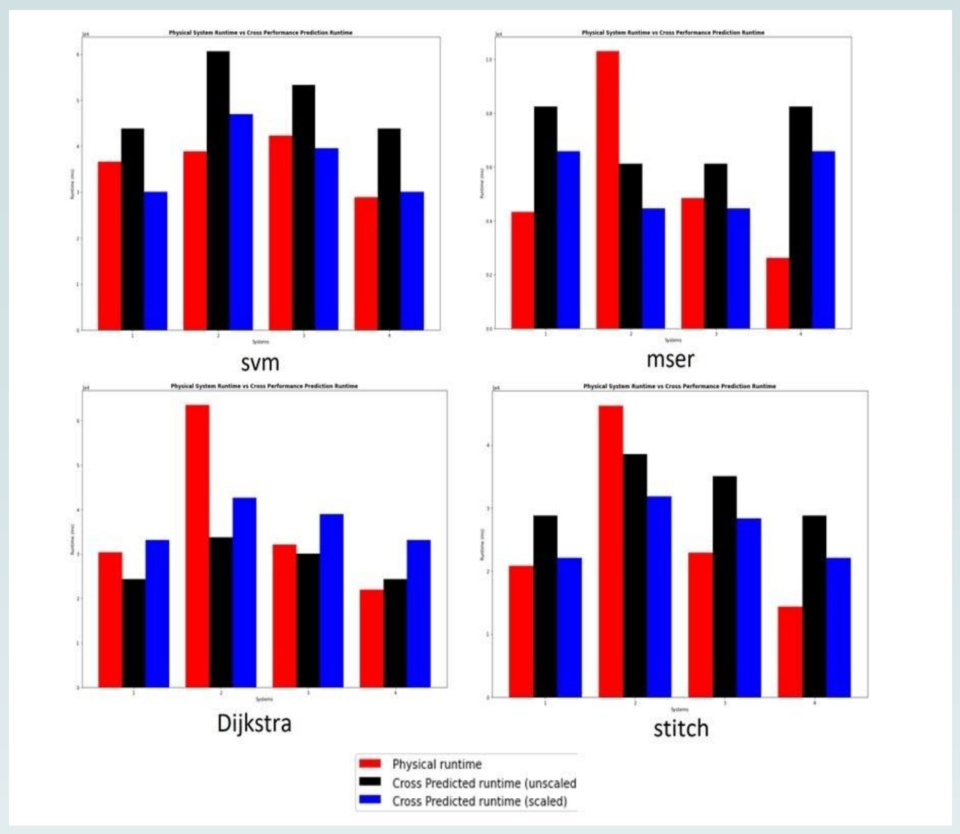
In the **model evaluation phase** we validate the y_{pred_i} to actual run time (y_{pi}) collected by executing an algorithm on physical systems (N) for which the cross prediction was performed.

To evaluate the Learned Model accuracy, we collect runtime y_{spi} by executing an algorithm on Gem5 simulator−based models with the features of the physical systems and compare y_{spi} with y_{pred_i} and this two run time will have difference of a factor we call it scaling factor.

We determine scale factor by comparing y_{spi} with y_{pred_i} before comparing it with actual physical system run time(y_{pi}). Then by applying scale factor to y_{pred_i} we get scaled $(y_{pred_i})*$. runtime At the and compare all three y_{pred_i}, scaled $(y_{pred_i})*$, and y_{spi}.

For each algorithm model is trained by finding best parameters using GridSearchCV method

Results



We have compared the actual runtime from physical systems to cross predicted runtime value from Learned Model trained only on simulation - based hardware. Results are similar to research paper results[2]. From the plot we see that server-like systems (2,4) have larger error between cross predicted runtime and the actual runtime as compared to general-purpose systems (1,3).

This is because simulation-based hardware models are representative of the general class of machines, therefore, the difference between hardware features between simulation-based model and the server-like machine is larger, resulting in a larger difference in runtime.

Also when we compare multiple actual runtime from physical system it has lot of variance for same system itself because of number of process running in background also errors in compute-bound algorithms are higher than memory-bound algorithms because of scaling factor for actual physical system there is gap in feature input this also leads to higher variance in runtime for compute-bound algorithm.

We achieved more than 90% accuracy for trained model accuracy on testing for dataset with 475 models

Acknowledgements

1) Link of Kaggle website for data to download: https://www.kaggle.com/
2) For confusion matrix https://en.wikipedia.org/wiki/Confusion_matrix
3) A. Mankodi, A. Bhatt and B. Chaudhury, "Performance Prediction of Physical Computer Systems Using Simulation-Based Hardware Models," 2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS), Shenzhen, China, 2020, pp. 1-5, doi: 10.1109/HPBDIS49115.2020.9130599.