Offloading of Mobile Apps in Hand-Held Devices using Machine Learning

Neelesh Chourasiya, Neeraj Mohan

***Abstract**:The usage of hand-held and mobile devices has been increased rapidly in recentyears. The execution of sophisticated softwares and Apps on mobile phonescan lead to poor performance with respect to energy consumption and response-time. With the emergence of the offloading concept of App workloads, an attempt has been made toimprove the performance of the hand-held devices by exploiting cloud service. The computation offloadingin hand-held devices consumes energy as well as time for transferring the datafrom hand-held devices to cloud. For the effective use of cloud services, there is a need to optimize the execution time of mobile App and energy consumedby the respective App. Many research endeavors have been made in recentyears to reduce the execution time and energy consumption during offloadingprocess. However, the usage of offloading has been evolved to quench the thirstof mobile users who execute multiple Apps simultaneously and are in dire needof seamless connectivity but some dynamic algorithms are needed to decidewhether offloading is favorable or not for a mobile App. If the mobile Apptakes more time and consumes more battery if executed on cloud then it isrecommended to use mobile platform rather than using cloud services. In thispaper, we are presenting machine learning based techniques which would help the mobile users in decision making to execute the App on mobile devices or on cloud using cloud services.*

***Keywords** :Offloading, machine learning, handheld devices, mobile computing, mobile Apps.*

I. INTRODUCTION

Smart phones have profoundly altered the way that people organize, socialize, work, and entertain themselves. Nowadays smart phones are an essential piece of business equipment for many professional organizations [29]. The number of smart phone users in the world is forecast to reach 4.7 billion by 2019 [28]. Most of the hand held devices comes with a word processor, Image processor, voice recognition and video processor, and these are such type of software applications, that consumes a huge computational power, system memory and bandwidth, resource constraints [1,2]. Battery power gets drained rapidly due to more number of applications. The growing trend of using compact devices has led people to stop using laptop and desktop computers [3]. The usage of smart phones is rising substantially which in turn demands more

hardware to fulfill the needs of the users [4].Offloading could be achieved in two different ways, either the entire mobile application is executed on the remote server by exploiting the cloud services or the mobile application can be decomposed into partitions and the partitioned components are then executed into chunks on the remote server and the results can be concatenated [5, 6]. In former method, the network cost is usually high, especially when there is a need to access mobile device sensors, the consumption of energy also exceeds due to transition of an application from mobile to cloud [7]. But in latter, this problem can be managed appropriately as the application is partitioned into offloadable and non-offloadable chunks. Non-offloadable chunks are those which are not appropriate for offloading. For example, graphicuserinterfaces, GPS, mobile device sensors, and network components [8].

The partitioning of an application is a mechanism of splitting up of an application into smaller chunks. These partitioned chunks are capable enough to operate independently in a distributed environment [9]. The application partitioning is a pre-requisite for compute-intensive application or for computational offloading. The partitioning can be achieved in different levels of granularity such as module based, object based, thread based, class based or the task based [10]. The Partitioning is performed to enhance the performance, minimizing the usage of memory, minimizing the energy consumption and to reduce the net-work overhead. The time and energy saving during offloading entirely depends on the network bandwidth, the amount of computations to be performed and the amount of data to be transmitted over cloud [11].

However, the hand-held devices are equipped with latest hardware and soft-ware configurations to suffice the needs of users up-to certain extent but still unable to meet the growing demand for sophisticated application softwares [12]. In order to suffice the needs of mobile users, numerous mobile applications have been launched in the recent years. The mobile applications comprise real-time games, multimedia, GPS navigation, e-commerce, banking and communication [13]. The mobile Apps are generally categorized into two types, compute-intensive and data-intensive. Both computational as well as data-intensive applications require minimum response time and maximum energy in terms of battery life.

Manuscript published on November 30, 2019.
* Correspondence Author

Neelesh Chourasiya*, Computer Science Engineering Dept., IKGPTU, Kapurthala, Punjab, India. Email: neelesh2412chourasiya@gmail.com

Neeraj Mohan, Computer Science Engineering Dept., IKGPTU, Kapurthala, Punjab, India. Email: erneerajmohan@gmail.com

In this paper, we present a decision making approach for mobile users to decide whether to use the cloud services for execution of mobile apps or the application should be executed on the mobile itself using machine learning techniques.

This paper has been organized into six sections. The first section provides introduction in the area of mobile devices and the offloading concepts. The second section elaborates the state-of-the-art work, while the third section of this paper provides detailed description on our methodology for predicting the decision when to go for offloading scheme using machine learning methods. The fourth section discusses about the performance matrices and the results obtained. The last section of the paper provides conclusion of the work presented in this paper.

II. RELATED WORK

The offloading concept has been evolved due to the rapid increase in hand-held devices such as mobile phones. Many state-of-the-art techniques have been proposed by researchers and academicians to offload the mobile applications on cloud for fulfilling their data and computational requirements. Most of the techniques are based on partitioning of mobile applications and rests of the techniques are based on decision making approaches. In this section, we have highlighted the state-of-the-art techniques which have made significant contribution in the area of offloading of mobile apps.

In [10], the authors have presented a multiple attribute decision analysis technique using analytic hierarchy process (AHP) and by using another MAD approach known as TOPSIS (the technique for order preference by similarity to ideal solution) in a fuzzy environment to decide which cloud is the most appropriate for offloading. The aim of proposed technique is to select the best cloud services based on MCDM approaches for offloading among the available public clouds that suffices the given user criteria. The proposed technique evaluates the public clouds based on the attributes such as speed, bandwidth, security, price and availability. First It makes use of AHP for finding out the weights of the criteria for selecting the cloud path and then make use of TOPSIS method to get the final ranking of the available clouds. In [14], the authors have proposed a methodology which helps to save energy using Frequency Scaling (DVFS) and Dynamic Voltage during mapping and scheduling stages. Mapping takes place while assigning a resource to the user application and scheduling takes place while executing application on the appropriate operating frequency. Two leveled genetic algorithm is proposed for handling mapping and scheduling in two stages. Applications are partitioned to form numerous task interaction graphs(TIGs), which are offloaded either to cloud resources or to surrogates. The execution of the task interaction graph must be is completed within overall deadlines. The proposed task successfully schedules tasks with 35 percent energy saving in a mobile device.

In [15], the code offloading decision making approach is proposed to find the most suitable cloud resources where the code could be offloaded based on the availability of the wireless network. Three cloud based resources are used such as public cloudlet, cloud, and Manet. The best wireless interface is selected based on six attributes such as energy cost, availability, link quality, monetary cost, link speed, and conjunction level of channel. AHP and TOPSIS MCDM techniques are used for decision making and selection. The suitable cloud resource is for the execution of the mobile application. In [13], the authors present a decision making approach for mobile consumers to decide when to make use of cloud services and when to use mobile apps for the execution of the application using machine learning techniques. This work focuses on leveraging the application processing services of cloud data-centers with minimal instances of compute-intensive component migration at runtime. The size of data transmission and the cost of energy consumption is reduced in computational offloading. The analysis of the results depicts that by employing the proposed technique, the size of data transmission over the wireless network is reduced by 84 % and the cost of energy consumption is reduced by 69.9 % during offloading different components.

In [16], the authors have optimized the existing offloading decision algorithm for independent tasks using one computing access point and a remote server on cloud. The proposed method reduces the total energy consumption cost, the cost of execution and delay. The problem is formulated as a non-convex quadratically constrained quadratic program. In order to solve this problem, decision algorithm using semi-definite relaxation and randomization mapping methods are proposed. The algorithm achieves optimal performance using computing access point. In [17], the authors have strived for achieving the efficient computational offloading by making use of game theory. An offloading decision algorithm as a decentralized computation offloading game is proposed. The mobile users are able to arrange themselves to obtain the offloading decision in a decentralized system. This characteristics of the proposed approach helps to minimize the managing issues of the centralized system which is achieved by assembling the information obtained from multiple mobile users. In this work, the decentralized computation offloading game is generated initially and later, the game is inspected in two wireless accesses cases such as homogeneous and heterogeneous cases. The authors claim that the proposed algorithm can achieve efficient computational offloading performance and scale up well with the increase in system size.

In [18], the authors have proposed a dynamic offloading decision algorithm that uses DPS (depth first search) for calculating the offloading point. The offloading point is calculated by using the call-graph. The call graph comprises nodes and each node has its own weight that represents the cost of execution cost and the edge weight provides the transmission cost. A call sequence is made of call graph that uses topological sorting (DFS). Afterwards, the linear search is applied on all the call sequences to determine the best beginning and ending offloading points.

The proposed technique reduces the energy consumption as well as execution time. In [19], a new offloading approach is proposed that performs method level offloading by migrating the required context information for the execution of the user application or task. It determines the requirement of memory contexts before the beginning of the execution of the task by decomposing the task offline into manageable executables chunks.

The results of decomposed tasks are stored along with the executables of application. These results are consumed by offloading algorithm to decide to migrate the required context information to the remote server over cloud. The proposed method helps to save the cost of transmission as well as consumption of energy.

In [8], the authors have presented a framework for execution of mobile application over cloud. First of all, then application on mobile device is stopped, and then the data les having the saved state of the application are forwarded to the cloud for execution. The loss of input data is escaped by employing application replay technique with the state saving scheme. The synchronization of data is prioritized by adding application state saving scheme and data categorization. The proposed methods reduce the amount of data to be transferred and save the consumption of energy. In [20], a partitioning algorithm is proposed for allocating the components of applications to remote server in public cloud. It minimizes the required band-width between the modules of application, but does not impact the execution time of tasks. It generates partitions using multi-level Kernighan-Lin (KL) based algorithm. It uses the combination of simulated annealing (SA) and multi-level KL graph partitioning algorithm to generate the hybrid partitioning algorithm. The advantage of using the combination of these two algorithms is to addresses the issue of randomness of simulated annealing algorithm. This hybrid approach also insures that none of the cloud server gets overloaded with the mobile applications.

In [21], a novel offloading framework is proposed which is based on the Inversion of Control mechanism to overcome the shortcomings of the existing offloading approaches. The proposed framework minimizes the burden on programmers by implementing the application partitioning and code offloading via remote proxy classes and callback functionality. In order to make decision on the application components to be offloaded, a call graph model based is developed. An offloading decision making algorithm is proposed to find out the classes to be offloaded during runtime. The proposed graph model suits well to the application partitioning problem. The offloading factory concept manages the resource accesses between the cloud and the mobile device at runtime. It has also been proven in the experimental study that offloading the optimal application partitioned components over remote servers can minimize the execution time and energy consumption of mobile devices significantly.

Since many researchers have proposed offloading techniques for optimizing energy consumption and execution time but still there is a need of a mechanism which can clearly guide the mobile user in making decision whether it really make sense to offload the mobile app on cloud or it is better to use mobile services only. In order to address this issue, we are proposing machine learning based methods which would clearly specify whether the application should be execution on mobile device or cloud environment.

III. PROPOSED WORK

The proposed machine learning based approach uses a few parameters to make decision whether a mobile application should be executed on a cloud or on a mobile device. The mobile application may involve a module to process data with the size of ‘n’ input, which is stored in memory. The application can also be decomposed into smaller chunks and each chunk can be treated as independent task. However there could be dependency in the execution of the tasks. We are just focusing on the attributes that makes significant contribution in decision making of offloading the mobile app on cloud. The data of the mobile app is processed by the CPU cycles and that cycles may belong to mobile itself or a cloud resource. Finally, the processed results are written back to the mobile memory. It is the need of the hour to provide QoS (quality of services) to the user without any interruption while a user is moving from one place to another. The offloading assists the user if during transition mobile resources such as memory or processor are not sufficient enough to provide seamless service to the user. It may be possible that if a user offloads the application on cloud without ascertaining the other factors, the application can consume more power and execution time rather than saving energy consumption and execution time.Our proposed methodology is capable enough to provide insights to the user to make right decision for offloading an application on cloud in appropriate situations. In Fig. 1, the proposed mobile offloading system is being presented where the decision of offloading a mobile app on cloud is to be made by the machine learning techniques.

The procedure of our proposed offloading decision is summarized below:

1. The user initiates the request to our machine learning based app for making the decision of offloading.
2. First of all the application gets partitioned into smaller chunks if possible so that energy consumption and execution time is saved.
3. The machine learning based mechanism invokes to get the readings of the parameters that help to make decision.
4. The machine learning based classifiers helps the user to determine whether the user should run the application or its partitioned part.

A. Partitioning of Big application into smaller tasks using graph theory

A graph partition is applied for the reduction of a graph into smaller graphs by partitioning its nodes into mutually exclusive groups. Edges of the original or main graph produce edges in the partitioned graph.

If the number of resulting edges is small as compared to the original graph, then the partitioned graph may be better suited for analysis and problem-solving than the original graph. The mobile application can be presented as a graph by showing dependency of tasks and here, in this paper, the graph is partitioned using betweeness centrality concept of graph theory.

Betweenness in graph theory measures the centrality of a node and is calculated as the fraction of shortest paths between node pairs that pass through the node of interest [22]. For each pair of vertices, there is a shortest path between the vertices such that the sum of the weights of the edges is minimized. We have used betweeness centrality to partition the mobile application graphs, where the betweenness is calculated for each edge and the edge with highest betweenness centrality is removed from the original graph. The partitioning is modelled as the component graph of the application. We have applied faster algorithm [23] to calculate the betweenness and to partition the original application graphs.The betweenness is calculated in two steps a) Calculate the length and the no. of shortest paths between all pairs of vertices, b) Sum all the pair-dependencies.

This algorithm requires (n + m) space and runs in (nm) time on graphs, where n depicts the no. of vertices and m depicts the no. of edges in a graph. The application component graphs are conveniently represented as a graph G = (V; E), where the set V of vertices represents tasks, and the set E of edges represents dependency between the tasks. The no. of vertices are represented by 'n' and the no. of edges by 'm'.

1. Calculate the betweenness for all edges in the dependent tasks of Apps.
2. Remove the edge with the highest betweenness.
3. Recalculate betweennesses for two subgraphs generated by removing the edge.
4. Repeat from Step 2 until the number of specified edges is removed.

B. Parameters considered for offloading decision making

Table- I: Parameters for Offloading Decision

S.No	Attribute	Description
1	temperature	temperature during offloading (0: cold, 1: normal, 2: hot)
2	mode	specifies the Android mode (0: fastest, 1: normal, 2: game, 3: ui)
3	t_bandwidth	Transmission Bandwidth
4	m_bandwidth	Memory Access Bandwidth
5	basic power	Basic Power when idle
6	Amt_DataProcessing_Unit	Amount of Processing Data into processing unit
7	Result_Data_Processing_Unit	Amount of resulting data from the processing unit
8	cpu speed	Mobile CPU Speed
9	coprocessor speed	Mobile Co-processor Speed
10	cloud speed	Speed of the cloud
11	cpu running power	Represents mobile CPU running power
12	co-processor running power	Represents mobile Coprocessor running power
13	module exe_time	Module Execution time on mobile CPU
14	Net_intpower con	Network Interface Power Consumption
15	Offload Target	{0,1} {0: Run the app on mobile & 1: run the app on cloud}

Dataset - Data is an essential part to train any machine learning based system. All the data, we are using for either for training the offloading system or testing the system. The training data is prepared and the system has been trained to make the decision whether the application or partitioned application component can be executed on mobile itself or over cloud.

1. Training Data - The data which is used to train our ML based system is known as training data. The proposed ML based model sees the data, learns to detect patterns and determine which features are most important during prediction.
2. Validation Data - Validation data is used for tuning model parameters and comparing different models in order to determine the best ones. Our validation data is different from the training data, and is not used in the training phase.
3. Testing Data - This is completely unseen data. In our proposed model, the new data which is collected during run time can predict whether the app should be offload on cloud or could be executed on mobile device. The testing data contains the data points that were not used in building ML based model.

The proposed offloading decision mechanism begins with the data cleansing and data visualization. The human brain is capable enough to easily process complex data using visuals such as charts and graphs. Hence we are making use of data visualizations before applying machine learning based classifiers.

1. Kernel Density Estimation (KDE): The KDE plot depicts the distribution of data. It explains whether the data has normal distribution or it has any kind of skewness. The KDE plot for transmission and memory bandwidth are shown in Fig. 1 and Fig. 2,CPU speed and Cloud speed are shown in Fig. 3 and Fig. 4 respectively, basic power of mobile and co-processor running power are shown in Fig. 5 and Fig. 6respectively.

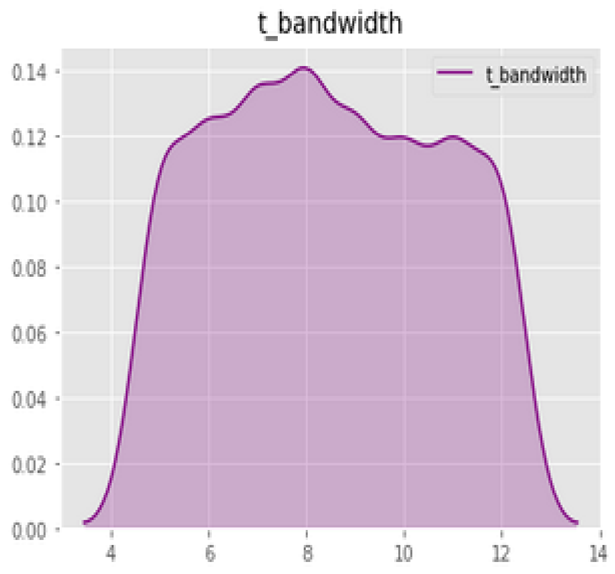

Fig. 1.KDE plot for transmission bandwidth

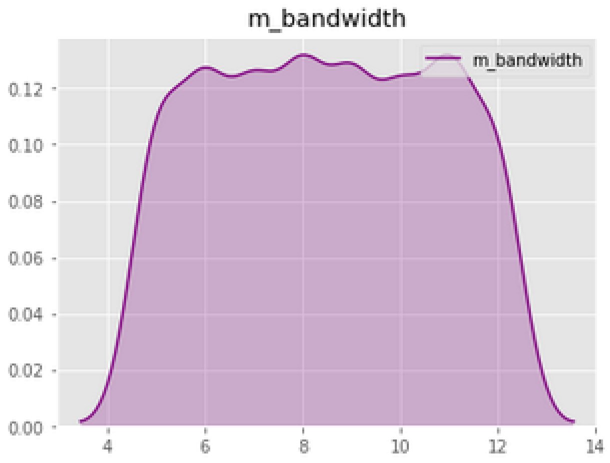

Fig. 2.KDE plot for memory bandwidth

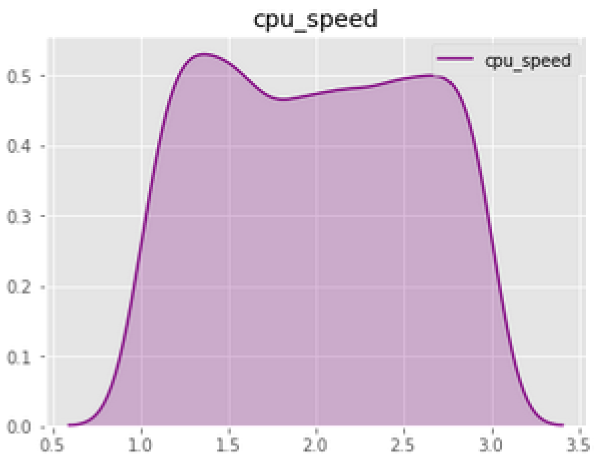

Fig. 3.KDE plot for CPU speed

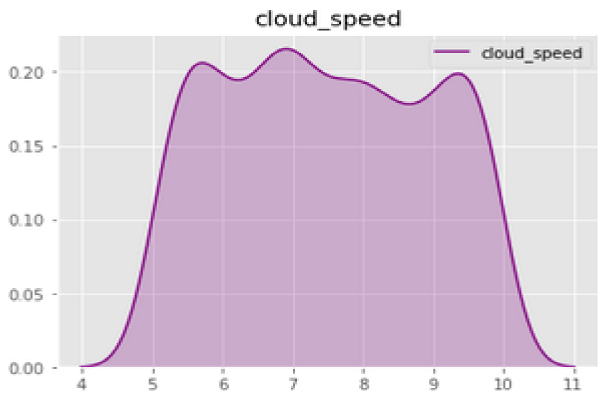

Fig. 4..KDE plot for cloud speed

We can observe from the KDE plots that the data is not in uniform distribution but we can also observe less skewness in most of the values depicted in KDE plots. Since it is not in our hand to make changes in the data as the data is being collected from the real environment. We have used the same data for training our ML based model.

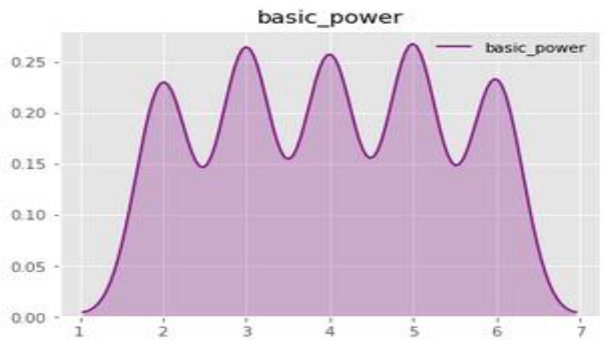

Fig. 5.KDE plot for basic power of mobile

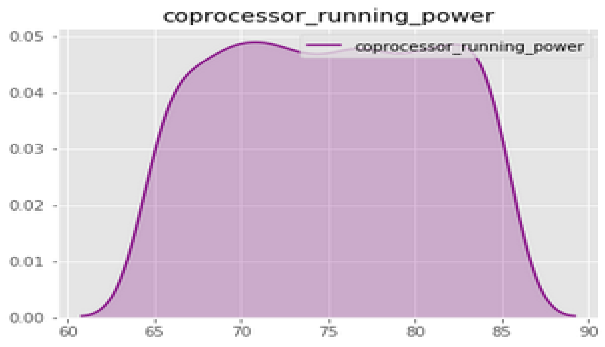

Fig. 6.KDE plot for co-processor running power

C. Offloading mechanism using machine learning (ML) techniques

This subsections describes the details of our proposed machine learning based offloading mechanism. We have used three machine learning based algorithms, K-NN, Logistic Regression, Support Vector Machine and Bagging classifier to predict whether the mobile app or partitioned app is suitable for execution on the cloud or on mobile itself. The performance of each algorithm is evaluated by using Classification Accuracy Score, confusion matrix, sensitivity score, precision,F1 score and RoC(receiver operating characteristic) curve.

K-NN based ML model- The first algorithm that we have used for our ML based offloading decision model is K-NN [25]. K-Nearest Neighbors(K-NN) is a supervised learning algorithm that uses the entire data for the training purpose. When we need to predict the class of the unseen data , it searches for 'K' similar instances and data with most similar instance is returned. It stores all the available cases and classifies the new data based on the similarity measure. It is one of the simplest machine learning algorithm which is used for regression and classi cation. We have applied here on our problem statement to depict whether the mobile app is to be executed on cloud or on mobile itself.
The evaluation of the proposed algorithm is made using confusion matrix as shown in Fig. 7, the ROC curve is shown in Fig. 8 and the other performance evaluation matrix is shown in Table 2.

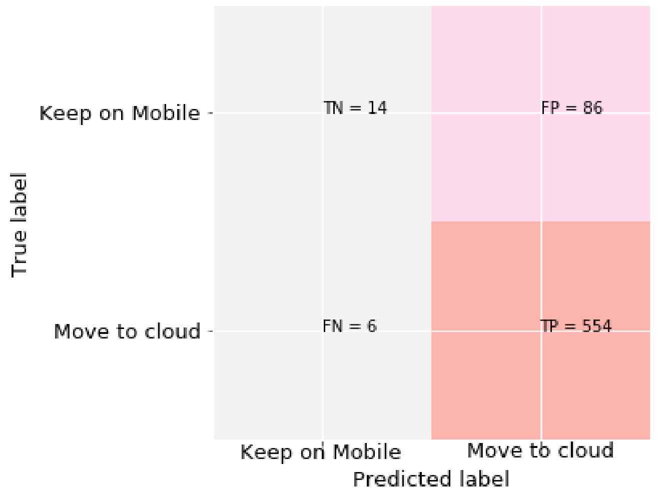

Fig. 7.Confusion matrix for K-NN based ML model

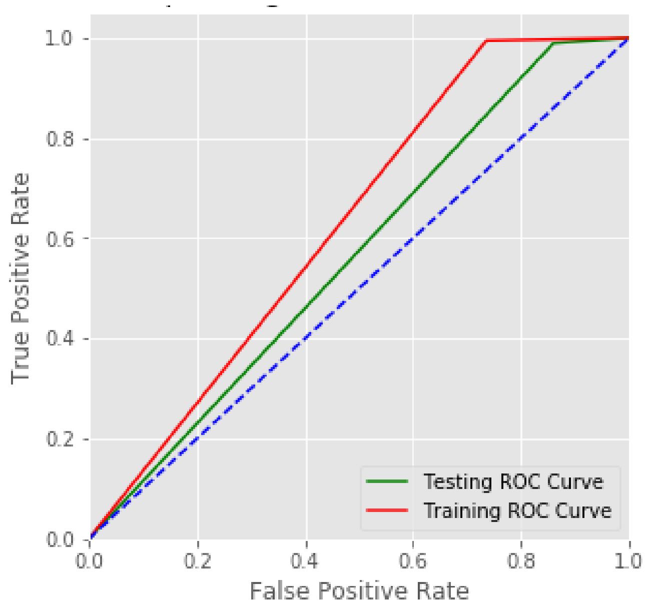

Fig. 8.ROC for K-NN based ML model

Logistic regression based approach- The next algorithm that we have tried for our ML based model is logistic regressor. Logistic regression is a kind of technique that is borrowed by machine learning from the field of statistics [26]. It is the method used for binary classification problems and prediction problems. This is suitable for our problem statement, hence we have made use of logistic regression for offloading decision. Logistic regression is a linear method, but the predictions are transformed using the logistic function. We have used logistic regression to predict whether the mobile app is to be executed on cloud or on mobile itself.

The evaluation of the proposed algorithm is made using confusion matrix as shown in Fig. 9, the ROC curve is shown in Fig. 10 and other evaluation parameters are shown in Table 2.

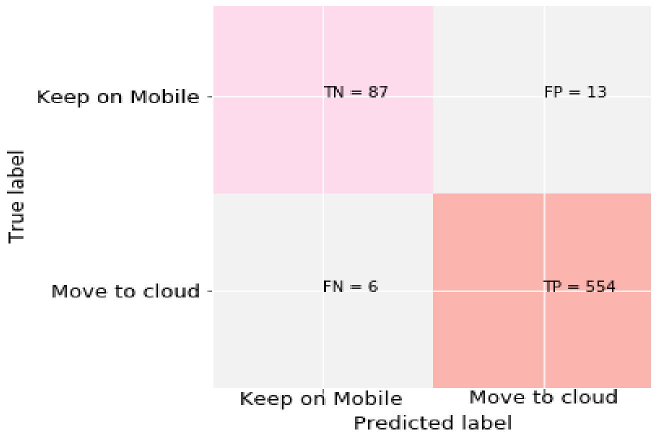

Fig. 9.Confusion matrix for Logistic Regressor(LR)

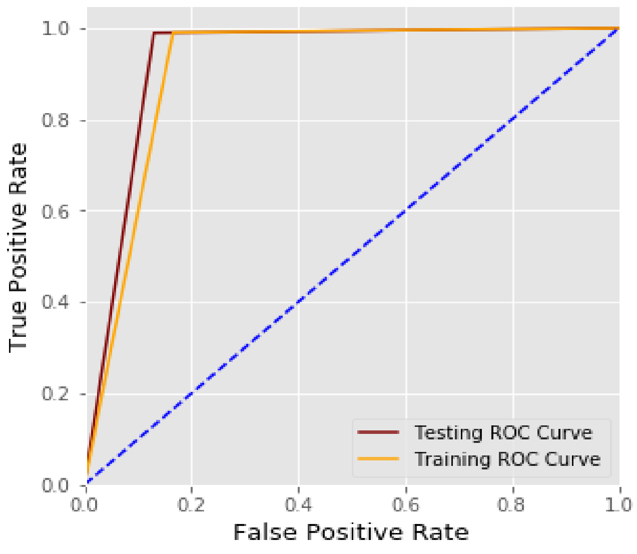

Fig. 10. ROC for LR based ML model

SVM based approach- The next algorithm that we have used for our ML based model is SVM(Support Vector Machines) is a supervised machine learning algorithm which is used for both classi cation and regression challenges [27]. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin. Support vectors are data points that are closer to the hyperplane and impact the position and orientation of the hyperplane.

Using these support vectors, we maximize the margin of the classifier. We have applied SVM on our problem statement since our problem is of binary classi cation.

The evaluation of the proposed algorithm is represented using confusion matrix as shown in Fig. 11, the ROC curve is shown in Fig. 12 and in Table 2.

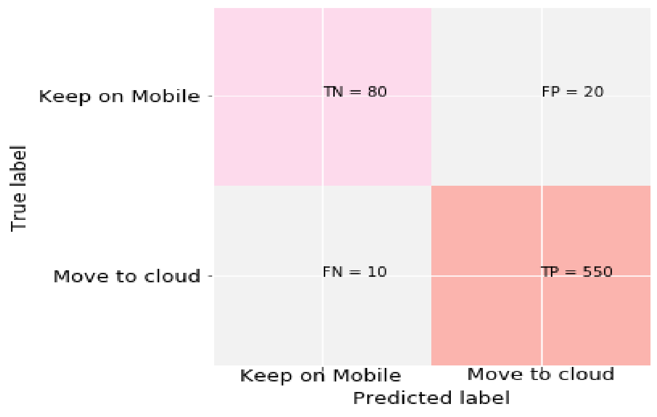

Fig. 11. Confusion matrix for SVM based ML model

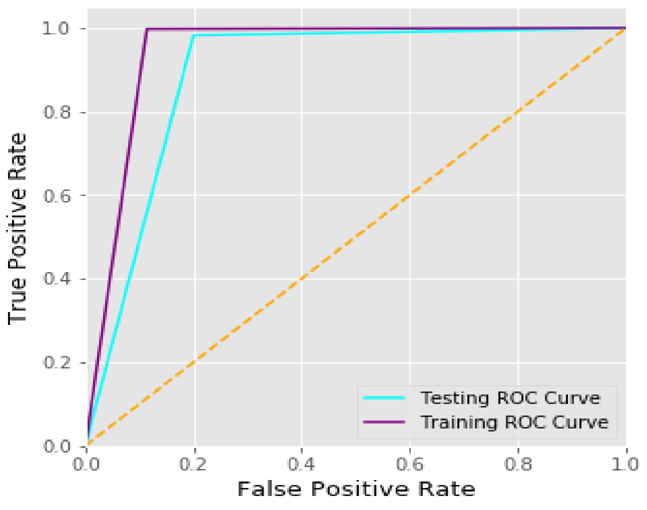

Fig. 12. ROC for SVM based ML model

Bagging Classifier–We have used Bagging classifier which is ensembling algorithm and is used for classification as well as for prediction [28]. Bagging stands for bootstrap aggregation. Bagging tries to employ similar learners on small samples of datasets and then takes a mean of all the predictions. Bagging uses bootstrap sampling to obtain the data subsets for training the base learners. For aggregating the outputs of base learners, bagging uses voting for classi cation. Bootstrapping is a sampling technique in which we choose n observations out of the original dataset of 'n' rows as well. The key concept is that each row is selected with replacement from the original dataset, so that each row is equally likely to be selected in each iteration. The evaluation of the bagging classifier is represented using confusion matrix as shown in Fig. 13, the ROC curve is shown in Fig. 14 and in Table 2.

We have used four different methods of machine learning to predict whether the mobile app or application component is to be executed on cloud or the mobile itself to minimizebattery consumption, app execution time and to increase throughput. We have presented the overall comparison of the performance of the algorithms considered for our study in Table 2.

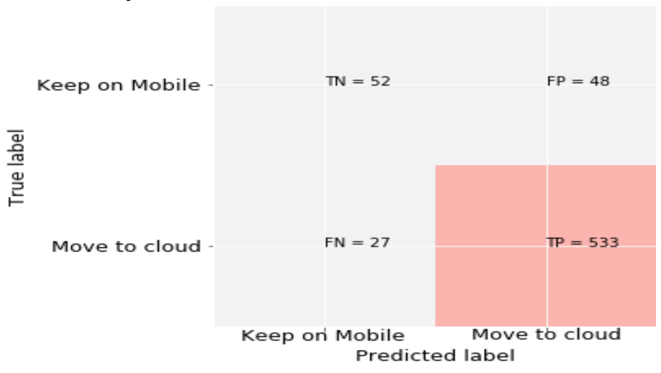

Fig. 13. Confusion matrix for Bagging classifier

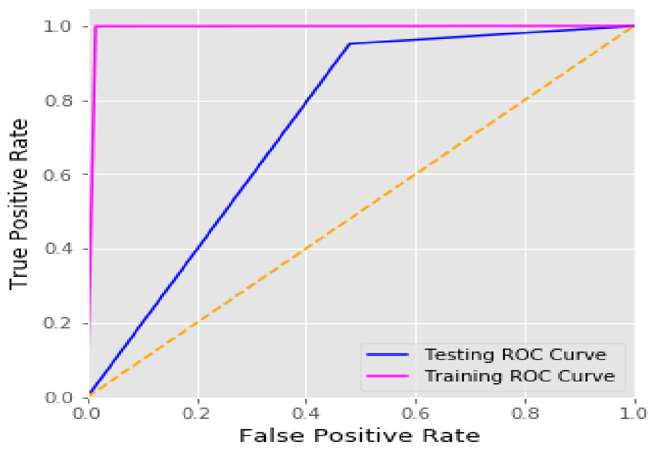

Fig. 14. ROC for Bagging classifier

Table- II: Performance evaluatio matrix for ML based models

ML Algos	Train-ningAUC Score	Testing AUC Score	Train-ning AUC Score	Testing AUC score	F1 Score	Preci-sion	Recall
K-NN	88.9%	86.0%	62.5 %	56.4%	0.91	0.93	0.86
LR	93.7%	94.1%	91.2 %	92.6%	0.94	0.94	0.94
SVM	97.1%	95.4%	94.1 %	89.1%	0.95	0.95	0.96
Bagg-ing	99.0%	89.0%	98.1 %	75.4%	0.95	0.94	0.95

IV. DISCUSSION ON RESULTS

In our proposed approach, first of all we have partitioned the mobile app into smaller chunks which can be executed separately to minimize the battery consumption and overall execution time. Then we have used machine learning based approach, where we have worked with the four different algorithms to see their impact with respect to performance using evaluation matrices and accuracy of results. Initially we have used K-NN and the training accuracy score is 88.96 % and the testing accuracy score is 86.06 %. To improve the accuracy, we have tried other algorithms such as logistic regression, Support vector machine (SVM) and bagging classifier. For logistic regression, the training accuracy score is 93.79 % and the testing accuracy score is 94.12 %.

For SVM, the training accuracy score is 97.1 % and the testing accuracy score is 95.4%. The last algorithm (Bagging classifier) considered for study gives training accuracy score 99% and testing accuracy score is 89%. We can conclude that machine learning based algorithms are suitable for our problem statement to predict whether the mobile app or component is to be executed over cloud resources or mobile resources irrespective of the type of app (Data-intensive or Compute-intensive).The dataset has been taken from standard repository of UCI for experimental study and has 70%of data is used for training the machine learning algorithms and remaining 30% is used for testing the proposed ML based model. The results presented in the paper are based on testing data.

V. CONCLUSION

In this paper, we have discussed our proposed machine learning based approach for making decision for mobile apps whether to use mobile resources or cloud resources for their execution in order to minimize battery consumption and execution time. Since mobile computing is the need of the hour and mobile users are increasing at the rapid rate.Along with the increase in users, the usage of apps is also increasing which consumes bandwidth, battery, processing capacity, space for data storage. There is a need to address the requirement of mobile users for data and compute-intensive apps to provide them resources in such a way, so that the execution time is minimized, battery consumption is minimized and the overall throughput is maximized. This can only be possible when the proposed system can direct the user whether the mobile app is to be submitted over cloud or it should be executed on mobile only. Hence, we have considered necessary parameters for prediction and applied machine learning based approaches which help the users to predict the environment (Cloud or mobile) for execution of the apps or app components.

REFERENCES

1. H. J. La, S. D. Kim, A conceptual framework for provisioning context aware mobile cloud services, IEEE 3rd International Conference on Cloud Computing, 2010, pp. 466-473.
2. J. Liu, E. Ahmed, M. Shiraz, A. Gani, R. Buyya, A. Qureshi, Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions, in Journal of Network and Computer Applications, 48, 2015, pp. 99-117..
3. J. Niu, W. Song, M. Atiquzzaman, Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications, in Journal of Network and Computer Applications, Vol. 37, 2014, pp. 334-347.
4. Z. Sanaei, S. Abolfazli, A. Gani, R. Buyya, Heterogeneity in mobile cloud computing: Taxonomy and open challenges, IEEE Communications Surveys Tutorials, Vol. 16, Issue no. 1, 2014, pp. 369-392.
5. E.Cuervo, Maui: Making smartphones last longer with code offload, in Proceedings of the 8th International Conference on Mobile Systems Applications and Services, 2010, pp. 49-62.
6. Z. Zhang, S. Li, A review of computational offloading in smart mobile devices for mobile cloud computing, in2nd IEEE International Conference on Computer and Communications (ICCC), 2016, pp. 2738-2743.
7. L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, A. Chan, A framework for partitioning and execution of data stream applications in mobile cloud computing, SIGMETRICS Perform. Eval. Rev. Vol.40, Issue no. 4, 2013, pp. 23-32.
8. S. H. Hung et al., Executing mobile applications on the cloud: Framework and issues, in Computers and Mathematics with Applications, Vol. 63, Issue no. 2, 2012 pp. 573-587.
9. S. A. Saab, F. Saab, A. Kayssi, A. Chehab, I. H. Elhajj, Partial mobile application offloading to the cloud for energy-efficiency with security measures, Sustainable Computing:Informatics and Systems, Vol. 8, 2015, pp.38-46.
10. H. Wu, Q. Wang, K. Wolter, Methods of cloud-path selection for offloading in mobile cloud computing systems, in 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, 2012, pp. 443-448.
11. K. Kumar, Y.H. Lu, Cloud computing for mobile users: Can offloading computation save energy? inComputer, Vol. 43, Issue no. 4, 2010, pp. 51-56
12. X. Wang, W. Xu, Z. Jin, A hidden markov model based dynamic scheduling approach for mobile cloud tele-monitoring, in IEEE EMBS International Conference on Biomedical Health Informatics (BHI), 2017, pp. 273-276
13. M. Shiraz, A. Gani, A. Shamim, Energy efficient computational offloading framework for mobile cloud computing, in Journal of Grid Computing, Vol. 13, Issue no. 1, 2015, pp. 1-8
14. P. Balakrishnan, C.K. Tham, Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing, inProceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, UCC ’13, IEEE Computer Society, 2013, pp. 34-41.
15. B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, R. Buyya, A context sensitive offloading scheme for mobile cloud computing service, in IEEE 8th International Conference on Cloud Computing, 2015, pp. 869-876.
16. M. Chen, B. Liang, M. Dong, A semidefinite relaxation approach to mobile cloud offloading with computing access point, inIEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), 2015, pp. 186-190.
17. X. Chen, Decentralized computation offloading game for mobile cloud computing, in IEEE Transactions on Parallel and Distributed Systems, Vol. 26, Issue no. 4, 2015, 974-983.
18. V. Pandey, S. Singh, S. Tapaswi, Energy and time efficient algorithm for cloud offloading using dynamic profiling, in Wireless Personal Communications, Vol. 80, Issue no. 4, 2015, 1687-1701.
19. Y. Li, W. Gao, Code offload with least context migration in the mobile cloud, in IEEE Conference on Computer Communications (INFO-COM), 2015, pp. 1876-1884.
20. T. Verbelen, T. Stevens, F. D. Turck, B. Dhoedt, Graph partitioning algorithms for optimizing software deployment in mobile cloud computing, in Future Generation Computer Systems, Vol. 29, Issue no. 2, 2013, 451-459.
21. M. Kaya, A. Kocyigit, P. E. Eren, Mobile cloud computing framework using a call graph based model, in Journal of Network and Computer Applications, Vol. 65, 2016,pp. 12-35.
22. M. Newman, A measure of betweeness centrality based on random walks, Social networks, Vol. 27, Issue no. 1, 2005, pp. 39-54.
23. U. Brandes, A faster algorithm for betweenness centrality, Journal of Mathematical Sociology, Vol. 25, Issue no. 2, 2001, pp.163-177.
24. D. B. et al., Applying PCA for traffic anomaly detection: problems and solutions, in IEEE conference INFOCOM, 2009.
25. S. K. Sahu, P. Kumar, A. P. Singh, Modified k-NN algorithm for classification problems with improved accuracy, International Journal of Information Technology, Vol. 10, Issue no. 1, 2018, pp. 65-70.
26. R.E.Wright, Logistic regression: Reading and understanding multivariate statistics, American Psychological Association, USA, 1995, pp. 217-244.
27. C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn., Vol. 20, Issue no. 3, 1995, pp.273-297.
28. L. Breiman, Bagging predictors, Mach. Learn., Vol. 24, Issue no. 2, 19956, pp. 123-140.
29. S. Husnjak, D. Peraković, and I. Forenbacher, Data Traffic Offload from Mobile to Wi-Fi Networks: Behavioural Patterns of Smartphone Users, Wireless Communications and Mobile Computing, Vol. 2018, Article ID 2608419, 13 pages.