International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Enhancing Cloud Cost Efficiency: A Predictive ML Approach for Optimized Resource Allocation based on Infrastructure Usage and Workload Behavior

Apurba Kumar Roy

Master of Science in Data Science, Liverpool John Moores University
(Student Id: 944585)

Abstract: ***There has been tremendous growth and development across the globe with respect to cloud adoption and usage. The traditional on-premises servers have given way to global cloud infrastructure with most of the vendors like Amazon, Microsoft and Google setting up dedicated cloud infrastructure and management provisions. These developments have allowed companies and organizations to concentrate more on building software applications and systems while the infrastructure part is totally taken care by the cloud providers themselves. While there has been rapid cloud growth and adoption across the industry, it has come with its share of cons too. Organizations worldwide started moving rapidly to cloud systems, migrating data and application, and paying only for what they used in terms of VMs, Clusters, CPUs, Memory, and components with zero maintenance cost associated with the infrastructure. But slowly few organizations began to realize that the cost that they started to pay for the usage of the Cloud infrastructure was becoming even more than what they used to pay for the traditional on-premises systems. Cloud adoption comes with numerous advantages, but when the resources and cost is not managed properly, the cost incurred can become huge and uncontrollable. This phenomenon requires an in-depth understanding and implementation of optimized resource management which would produce an optimized cloud cost resourcing model exploiting the workload characteristics and machine learning approach to produce substantial results. Using the real time Microsoft Azure workloads, we can predict the accurate optimized costing for future workloads which would help to take informed decisions towards cloud resource management and costing. The research proposes to take a closer look at the various aspects/Characteristics of Microsoft Azure Cloud implementation through the production Virtual Machine workloads and how we can take into consideration the cost perspective to take corrective steps to maintain a balance between the cost towards infrastructure and performance, at the same time reap the benefits of the cloud infrastructure in a more efficient and cost-effective way. The outcome of the research is an algorithm which considers all factors into account including uncertainty of workload behavior , different scaling options for different workload characteristics ,parallel running tasks factor and impact on workload behavior, adhering to the expected SLA and performance characteristic, efficient use of resource and avoiding resource wastage and higher cost expenditure, volume discounts for larger workloads, running times of VM having both short and long duration and predicting the future behavior based on the current selected workload thereby allowing provisioning strategies and pricing strategies from Azure costing portal. The research starts off with an analysis of the Azure workload and it characteristic followed by cleanup of the dataset. Thereafter the research intends to run various ML and Deep learning-based algorithms to ascertain the best features and co-relations among them to finally arrive at the best fit algorithm which considers all the factors and accurately predicts the cost & resource optimized model.***

List of Abbreviations

AC	Active Clustering
ACO	Ant Colony Optimization
ADP	Approximate Dynamic Programming
AE	Auto Encoder
ANP	Analytical Network Process
APRA	Automatic Proactive Resource Allocation
AR	Auto Regression
ARIMA	Auto Regressive Integrated Moving Averages
BD	Benders Decomposition
BRS	Biased Random Sampling
CECRPICAO	Competitive Algorithm Optimization
CNN	Convolutional Neural Network
DblSes	Double Exponential Smoothing
EDA	Exploratory data analysis
GA	Genetic Algorithm
HFA	Honeybee Foraging
KDE	Kernel Density Estimation
KF	Kalman Filter
LR	Linear Regression
MA	Moving Averages
MC	Markov Chain Algorithm
MCDM	Multi Criteria Decision Method
ML	Machine Learning
MOO	Multi-Objective Optimization Model
MPNN	Multilayer Perceptron Neural Network with Back Propagation
NB	Naïve Bayes
NF	Noise Filter
OCRP	Optimal Cloud Resource Provisioning
PCA	Principal Component Analysis
PDRV	Probabilistic Determination using Random Variable
PSO	Particle Swarm Optimization
RBM	Restricted Boltzmann Machine
RC	Resource Central
RNN	Recurrent Neural Network
ROC	Receiver Characteristic Curve
RPPS	Cloud Resource Prediction and Provisioning Schema
SAA	Sample Average Approximation
SAM	Server Allocation Matrix
SBS	Service Based System
SIP	Stochastic Integer Programming
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SVM	Support Vector Machines
VM	Virtual Machine

1. Introduction

1.1. Background of the study

Cloud computing has been gaining popularity all around the world in the last couple of years. There has been rapid strides of improvement and growth in terms of the services and features that cloud vendors now provide. The catalyst for these rapid changes has been the competition in the marketplace among these vendors like Amazon, Microsoft, and Google to capture the cloud computing marketplace. The biggest beneficiary of these technological advances due to competition have been the organizations who have realized the huge benefit that cloud computing provides at the same time reduces the overhead of maintenance of servers and datacenters of their own. This has led organizations to quickly incorporate cloud computing as part of their organization strategy towards implementation and adoption moving away from the traditional on-premises servers. The below diagram shows the benefits achieved based on adoption levels:

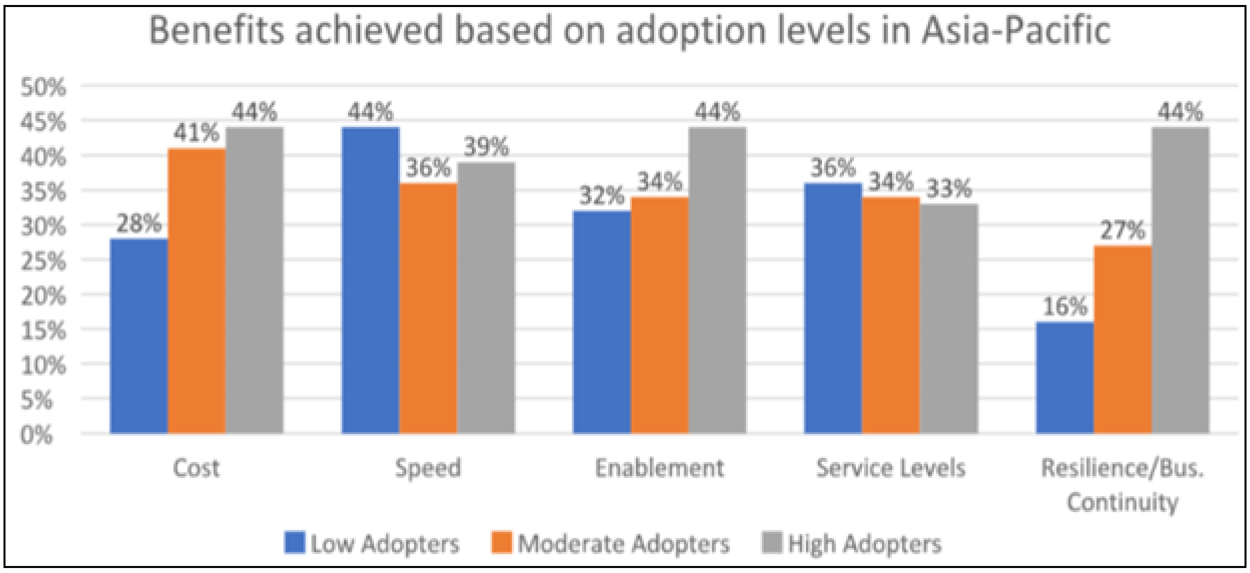


Figure 1.1: Cloud Adoption Level Benefits

The below figure also shows the adoption levels based on the region:

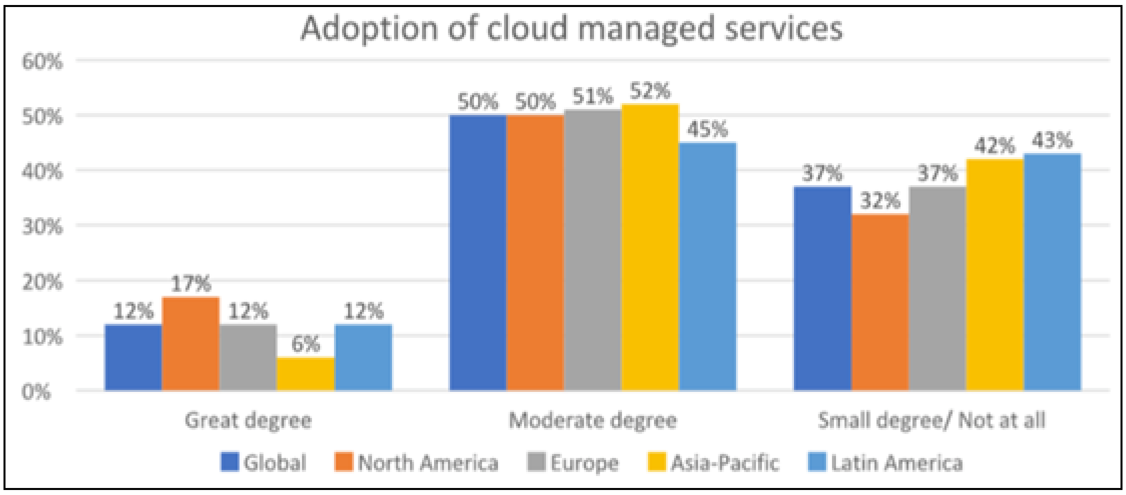


Figure 1.2: Cloud Adoption level by region

While the benefits of moving into cloud computing from the traditional server-based architecture are many, cost savings being one of the top 4 benefits as organizations only pay for the duration, they use the cloud infrastructure. However, if not properly implemented, it can also have adverse effects in terms of quickly becoming costlier than even the traditional on-premises servers

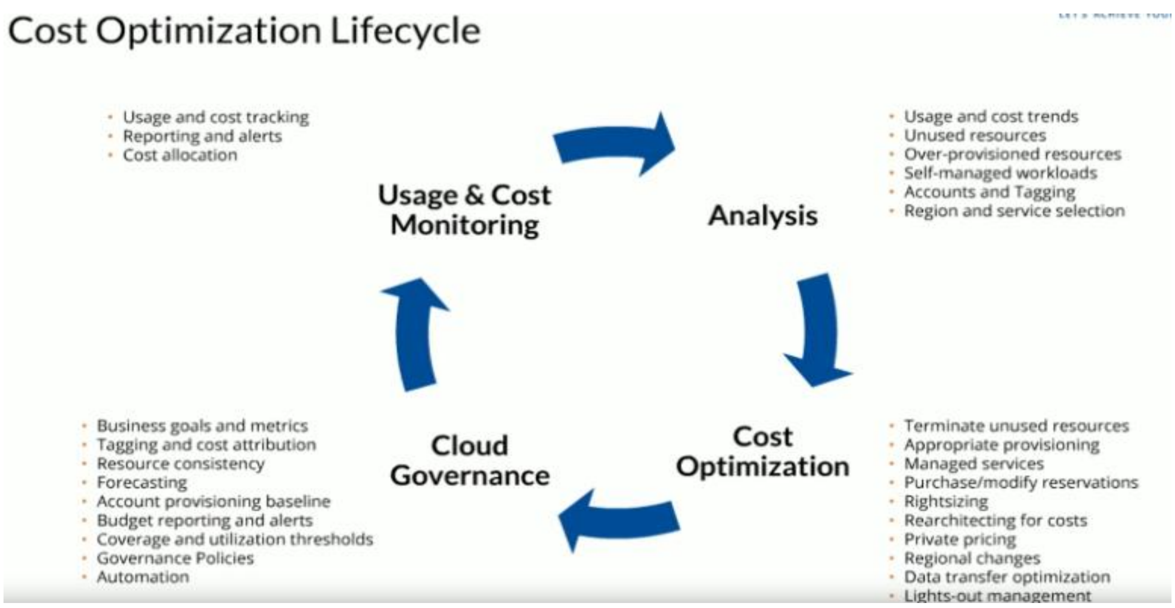


Figure 1.3: Cloud Cost Optimization Lifecycle

This research intends to present a predictive analytics approach to the resource and cost management aspects by considering real time production workload traces containing virtual machine and component information gathered over a period for spanning different regions and resource groups. Large vendors have been providing information of cost to customers based on current usage, but that information is quite limited. What is novel in this research is that this is the first time both optimized resource management and cost are combined, and predictive analytics used on real time workloads to predict the optimized cost and resources to be used and factors effecting the cost in the long run. Also, this research intends to do an in-depth analysis on the cloud component factors along with different combinations of configurations which can reduce the cost and empowers the organizations to take informed decisions on the capacity and configuration planning of the resources.

1.2. Aim and Objectives

The primary aim and goal of this in-depth research work undertaken is to identify and introduce a machine learning based Optimized cloud cost model based on Infrastructure Workload behavior. However, the main objective can be broken down in the below pointers which forms the base of the research conducted in this niche area.

1) To correctly predict workload classification: Model should be able to classify the workloads in terms of workload parameters and predict the Future workloads
2) To implement a scaling mechanism: Model should be able to explore the various resource scaling mechanisms and thereby select the best cost optimized scaling mechanism for the workloads in Microsoft Azure cloud environment.
3) To address uncertainty in real-time workload distribution: Model should also explore the parameters and characteristics which introduce uncertainty in the workload behavior and how this uncertainty can be taken into consideration while predicting the future workloads and thereby allow resource provisioning accordingly
4) To include a comparative study in terms of the Predictive models used and the identify the most balanced model
5) To suitably evaluate in terms of performance of the suggested Model

1.3. Research Questions

The below contains the research questions when we mention about a cost-effective resource optimized predictive approach for cloud providers especially considering Microsoft Azure cloud:

1) How does the cost effective optimal or just right resource provisioning can be done for all future workloads?
2) How can the scaling mechanism tackle the problem of dynamic workloads and ensure cost optimized cluster scaling is done?
3) How can the uncertainty of the workloads in terms of various factors can be efficiently handled?
4) How can the jobs which are running in the Azure cloud space complete without slippages in terms of SLA or user expectation?
5) How to reduce resource wastage and incorporate volume discounts?
6) How can we ensure that the Proposed Model is scalable to be used across all Azure Workloads?
7) How can the dominant features affecting cost can be identified from the Microsoft Azure workload?
8) How can we ensure that the proposed model is not bulky and is lightweight in terms of implementation and performance?

1.4. Scope of Study

The research work and study introduced in this research work is aimed at proposing a Machine Learning based predictive analytics approach to solving the problem of optimization efforts in cloud cost and resource management for Microsoft Azure workloads.

1) The study does a detailed analysis and deep dive into the factors to be considered in terms of the Azure

Workloads and how it effects the overall cost of the organization across the different components of the Azure Cloud infrastructure

2) The selected dataset for this study contains Microsoft Azure VM traces containing valuable information on the Cloud component parameters required to study and predict the cost
3) Deep dive into the traces from Azure workloads also considers the effective and efficient resource utilizations for the workloads.
4) Capacity planning and resource planning of the cloud resources leading to savings is another aspect that this study would explore and suggest the best approaches for the same.
5) The study aims to propose a methodology which will provide good performance, availability, and reliability under various workload conditions and at the same time effective cost andresource management.

1.5. Significance of the Study

As part of the work presented in this research work, I intend to study the Microsoft Azure workloads spanning different geographic locations and containing valuable information on the different component parameters of cloud computing seen to be working when a particular job or jobs run in any region.

The study would include characteristics of production Virtual Machine (VM) workloads and a thorough understanding of the servers, power consumption, CPU utilization, VM lifetime, Deployment size and its relation to the overall infrastructure cost.

Additionally, this research work also looks at the Azure functions being invoked for the various production workloads and what is the impact of such invocation on the overall cost incurred to the organization.

This study and prediction algorithm build would give an insight into the various factors in play when implementing a truly cost-effective solution of optimized resource provisioning and optimized performance while adhering to the accepted SLA s levels.

2. Literature Review

2.1. Introduction

The “literature review” part of the research study provides an overview of the relevant earlier research done which addresses the topic of prediction of an efficient cloud cost and resource optimized model and the detailed description of what each study tries to address and solve. It also shows the different approaches that earlier researchers have tried to explore, compare, and come up with accuracy in predictions. The research workshows the length and breadth of the issues that can come up during build of an algorithm which can accurately predict the cloud optimized cost and resource including good performance

2.2. Related work

(Al-Faifi et al., 2018) Al-Faifi, Biao Song, Mehdi Hassan et all in 2018 proposed a Naïve Byes classifier to predict the performance parameters and metrics of cloud nodes with reference to different options for configurations for resources. They used a Kernel Density Estimation technique to resolve the difficulty of zero variance of the distribution of features and thereby improve the accuracy of the prediction. The problem addressed in this research was to study and predict the performance metrics of cloud nodes based on the workloads and is solved by taking a huge dataset of labelled workloads from the cloud nodes and then building a Naïve Bayes Classifier from the labelled data. The Naïve Bayes classifier would then be able to predict unlabeled example based on the information learned from the labelled samples. A comparison is made between Naïve bayes and NB-Kernel Density Estimation and where NB-Kernel Density Estimation was found to be more efficient to predict the performance of cloud systems.

(Al-Faifi et al., 2019) Al-Faifi, Biao Song, Mehdi Hassan et all in 2019 proposed an intuitive MCDM algorithm to assess and use a ranking system for cloud providers from the analysis of smart data. This method has taken into consideration the dependencies and interrelationshipsamong the key performance elements. TheMCDM contains 2 basic steps. Step 1 involves clustering the providers of cloud services using K-means clusteringalgorithm to combine them based on similar features. Step2 involves implementation of MCDM methods using DEMATEL-ANP. This step is used to take the final decision based on the ranking of the clusters received after implementation of the step 2 processes. The entire process involves implementation of the K-Means Algorithm to separate the providers based on a set of criteria and assignment of importance and weights for them considering the existing workload behavior observed in the organization. Cloud providers offer a variety of services all with differentattributesand components and therefore the differences are also seen in terms of the cost and performance for all such attributes specific to the cloud providers. However, there is a set of characteristic features which are found to have overlap. Hence for such scenarios ANP is a highly recommended solution as it deals with the different overlapping features and attributes. In addition to it, it is also observed that the number of cloud service providers is quite big and with it the gamut of criteria and sub criteria also increases. Hence it is a complicated and time-consuming process to calculate and form a super matrix of such criteria’s available to us. As a solution for this problem, clustering was used and they application of the Analytical Network Process (ANP). This has resulted in the extraction of highest quality representatives from each cluster and the simultaneous selection of those representatives.

(Calzarossa et al., 2019a)Calzarossa, Vedova, Tessera et al. in 2019 proposed a framework which deals with unexpected constraints or situations as well as the performance variations seen in cloud infrastructure workloads. This research work proposes an effective way provisioning and scheduling such uncertain workloads. The proposed methodology enables advance estimation of resource requirements even before the actual execution of applications, the settings required for resources which cope with uncertainties. The entire process involves the representation of the uncertainties associated with unexpected constraints and variations through random variables and thereby formation of an effective optimization problem statement. The resolution of the problem statement is envisaged and achieved through the implementation of a probability distribution modelling exercise. While considering provisioning and scheduling decision the uncertainties encountered in cloud environment is fully accounted for. For example: Considering different metrics like cost of leasing cloud resources,execution time and subsequent optimization efforts for the same. The various types of unexpected constraints or situation effecting the cloud workload and their sources are also investigated along with classification. According to this methodology, before the actual job execution occurs, scheduling and provisioning is completed well in advance of it, resulting in the prevention of risky provisioning paradigms like over-provisioning and under-provisioning. Hence also has an impact in terms of cost and ensuring an optimal usage of resources as per the scheduling plan.

(Di et al., 2012) Di, Kondo, Cirne et al. in 2012 evaluated and proposed a prediction algorithm which is based on Naïve Bayes model. This method is used in the prediction of mean load over an extended time slice in the future. The evaluation of the method was done using one-month worth of trace data from data centers from google consisting of huge number of machines. The evaluation and model running exercise conducted during research showed that the NB method is quite efficient and was able to show a mean squared error of. 0014.This means that this method was highly accurate. Additionally, the NB methodology was seen to enhance the load prediction accuracy. The value was found to be between 6 to 50% in terms of accuracy when it was compared with other sophisticated methods based on Auto-regression (AR), MA,noise filter (NF)etc. This method is particularly helpful in job scheduling as well as host load balancing decisions through implementation of prediction of the host load. All the characteristics that are studied in this research worklead to better resource utilization and helps to bring down the cost associated with the data centers by shutting down the idle machines leading to improvement in job performance. In this research methodology, the focus has been two critical metrics i.e., Memory and CPU. The generic approach has been to use the NBmodel due to its capability to hold information on noise and variations in load, thereby making suitable and appropriate predictions.

(Yang et al., 2014) Yang, Liu, Shang et al.,2014 proposed a model which uses the basic linear regression algorithm/mechanism. This algorithm is used to predict workload on a novel service cloud architecture and based on the workload predicted an autoscaling mechanism is proposed to scale the virtual resources. This method is proposed to be implemented at resource levels in service clouds which are essentially different from each other. Autoscaling combines pre-scaling. Tosatisfy the user Service level agreement (SLA) a at the same time keeping the costs low real time scaling is proposed. The research considers 3 scaling techniques: Resource Level Scaling, Self-Healing Scaling and VM level scaling which are implemented on top of the proposed novel cloud architecture. If two virtual machines which are part of a service are assigned the same cluster resource nodes, then resources of the virtual machines complement each other. This is the basic idea of the self-healing scaling highlighted in this research work. In this methodology the unallocated resources available at an instance of the cluster node are used up by the resource level scaling. As part of the proposed novel cloud architecture, there is no cost associated with this kind of self-healing scaling and hence it executed first followed by resource level scaling. The cloud architecture proposed has various components like Admission Controller, Load Balancer, workload Predictor and Scalability Manager. User requests are routed to the services through the Admission Controller, and it filtrates out invalid login requests followed by the requests getting routed to the Load balancer. The LB collects server stats from the clusters and allocates the requests to a suitable VM. Based on the historical data available on the workload information, a prediction algorithm is used to predict the service workload in the next interval of time. This algorithm is known as the Workload predictor.Scalability manager is another component which is proposed in this research work and is responsible for implementation of a scaling strategy in real time based in the prediction results of the earlier workload predictor as well as running states of the platform. Cumulative scaling stats show the approaches and performance achieved by the proposed algorithm considering various intervals. The performance and cost implication of horizontal scaling is not found to be satisfactory. Additionally, it leads to more idle resources than other proposed algorithms in this research work leading to more cost. The results of the methods implemented clearly depict that out of the 3 levels of scaling approaches the one that performs better and give satisfactory results with lesser cost is auto scaling and is about 13% lesser than other scaling techniques proposed. Utilization of the autoscaling, the lightweight scaling and horizontal scaling are around 82%,84% and 74% respectively. Auto and Lightweight scaling can also help in 100% utilization of the resources and the later one actually leads to resource wastage.

(Barnwal et al., 2019)Barnwal, Pugla 2019 proposed a model for developing a technique for balancing the workload effectively. The research work did a comparative study analysis of BRS, HFA and AC Techniques. The BRS algorithm can create a network system which has the capability to check if there is any workstation available to ensure dynamic allocation of jobs to such network systems. Edge dynamics is used in this research for balancing purpose. The overall result includes a directed graph, and random sampling is enabled by edge direction. A beehive-based load balancing algorithm was used at the application layer to maximize resource allocation. This work largely revolved around assigning servers in service-oriented architecture paradigm and subsequent assignment of web aps to the corresponding servers. Additionally, this research work also looks at maximizing actual usage and related capacity of the servers.A system of this sort is exceptionally helpful considering the case of cloud computing paradigms and their generic pay-as-you-go model. Hence their acceptance or rejection solely depends on the model selected.The overall return of profit is normally calculated considering the CPU usage timings depending on the server cost which hosts the virtual serverand considering the savings by calculating the time for which the virtual server was allocated and used. Additionally, the author also gives an in-depth analysis of well-established services being clubbed together to interact with load balancing through the method described. Similar types of instances are clubbed together to be used accordingly.Load balancing algorithms work well with similar workstations having similar types of characteristics and helps in load balancing. As the number of workstations increases, performance metrics of active clustering and Random sampling becomes better. However, there was not much improvement seen in case of honey been foraging implementations.

(Biswas et al., 2015) Biswas, Majumdar, Nandi, 2015 proposed a predictivemethod for dynamic automatic scaling of cloud resources that considers the dynamic system load and adjusts the number of resources required for the cloud accordingly. The method and system used is helpful not only in bringing about profit for the intermediate organization but also leads to reduced expenditure for the clients and users alike. There is a concept of a broker introduced in this research work where the broker has the entire responsibility of resource allocation coming through in the form of requests. The same broker also has the dual responsibility of automatic scaling operations. Instead of buying or leasing from the various enterprise cloud providers, users would have the flexibility to rent instances from the intermediate layer containing the broker hosted by the cloud provider. The primary concept of the broker here is that it charges the users of the private intermediate cloud with higher price, but to the tune of per second charged compared to cloud providers who charge per hour and therein lies the cost reduction for the end users and revenue generation or profit for the private or intermediary cloud. Additionally, users pay only for the time their request is running. This research work proposes an algorithm which increases the broke profit through proactive auto scaling mechanism. The broker is responsible to auto scale the resources through a ML based method/algorithmand is responsible for computing the latest no. of resources required to be provisioned for handling the upcoming requests. This leads to ensuring the required profit for the intermediary cloud. This process utilizes a ML based engine through the Software functions of a library known asWeka. Weka learns from historical request patterns from the user and is responsible for predicting future requests. Time series analysis is used to auto scale the resources proactively. Weka uses Linear Regression (LR) library to predict future requests. SVM is also used to simulate the scenario so that multiple model benefits can be recorded. However, when the comparison was made it was found that Support Vector Machines gave exceptional performance compared to Linear Regression techniques. On the other hand, it was found that execution time was more in case of SVM when compared to linear regression and using SVM increased the system timings to a great extent. This was looked upon as a possible overhead.

(Wang et al., 2013) Wang, Liu, Ni et al. 2013 proposed a cloud brokerage service which reduces the cost of the users by provisioning the resources from a large pool of instances from cloud providers and gives discounted rates to users. The research work proposes a system which initially reserves huge number of cloud resources rather than going for the traditional on demand route. Thereafterthis approach is seen to lower the pricing and eventually the cost by consolidating the user requests. The broker is implemented using dimensionality reduction technique using Approximate Dynamic Programming (ADP). Consequently 2 approximate algorithms are proposed which further is used for the short- and long-term predictions.

(Yousefyan et al., 2013) Yousefyan, Dastjerdi et al.,2013 proposed aneffective cost optimized cloud resource allocation and provisioning technique using CECRPICAO. As a first step the demand is predicted using a demand predictor algorithm and the same is used by CECRPICAO to precisely assign the virtual machines with on demand facility for reservations. CECRPICAO provides the best affordable solution as per the results obtained from the evaluation phase compared to all other resource provisioning methods. This method introduces a cost-effective resource provisioning technique which minimizes under provisioning and over provisioning significantly. The authors introduce a predictor algorithm to correctly predict the demands of resource provisioning requirement using a multi-layer perceptron neural network which has the capability for back propagation learning algorithm and is used to accurately predict future resource provisioning demandrequest made for resources like Virtual machines. The competitive algorithm or CECRPICAO is effectively incorporated to predict and provide accurate and efficient decision that leads to dividing the resources requirement of VMs among providers economically. The outcome of the evaluation phase of the method clearly indicates the total cost required to provision resources have been effectivelyreduced after the application of the mentioned method/Algorithm.

(Chaisiri et al., 2012) Chaisiri, Lee et al.,2012 proposed a virtual machine placement algorithm that has the capability to reduce the cost that is being incurred towards virtual machines being hosted in a multi-cloud provider environment. It uses Optimal Cloud Resource Provisioning (OCRP) to provision resources which are being offered by the various cloud providers. The authors used a SIP paradigm for acquiring a cost-effective solution to rent resources. This research technique specifically uses SIP for cost optimization followed by a decomposition algorithm known as benders decomposition and sample average approximation techniques (SAA)which is user to accurately predict the requests coming in the subsequent stages. From the evaluation of the techniques used, it is found that the algorithm can provide an optimal adjustment in terms of tradeoff between resource reservation and allocation of on -demand resources hence effectively reducing cost during provisioning of the cloud resources.

(Genaud and Gossa, 2011)Genaud, Gossa 2011 proposed a model which can be used to provision the cloud resources. This model is based on a comparison and a tradeoff between the cost associated versus the wait time for client-side modules dealing with provisioning of resource. The proposed model divides the users requesting the resources based on wait time that they can afford in running the jobs. This model considers the requests from users where the time required to execute is not of much importance. Additionally, this model also considers the users who have certain fixed deadlines set for their jobs and therefore time required for execution is critical under such situations. Implementation of this novel modelin case of the first group of users meant less resource requirement and lower cost as they can be provisioned with a wide variety of techniques and mechanisms. On the other hand, the second group of users apply more resources intensive techniques

(Mark et al., 2011)Mark, Niyato et al., 2011 proposed a model which aims at reducing the cost associated with resource provisioning cloud environment. This model has two parts where the first part deals with the usage of a forecasting algorithm for demandsto facilitate the prediction of future virtual machine (VM) requests from users through their historical data of requests made. To effectively do this the authors used a basic Kalman filter (KF) which followed by the usage of a DblSesor double exponential smoothing mechanism. Additionally, the author has also gone in-depth of Markov Chain Algorithm (MC) to predict the accurate results. In the subsequent section,optimization of the model is being presented by the author. In this section, a hybridized algorithm consisting of genetic Algorithms (GA), Particle Swarm Optimization (PSO) & Ant Colony Optimization (ACO) is effectively utilized. Overall, both the sections together combine to provide a means to cater to demands of VMs allocation and to provide minimization of cost of provisioning of the resources.

(Jagannatha et al., 2017)Jagannatha, Shravan, Kavya, 2017 proposed a model which is used to determine the resources which are in working condition at any instant of time and secondly how these available resources can be used effectively. The cost performance analysis is done using the Markov Model (MC) and the server allocation matrix (SAM). The research work delves into the area of resource allocation by considering the failure of the physical machines in the datacenters that constitutes the cloud. The model takes into consideration the generic failure of data center machines, repair time and uptime of the machines to take the incoming load of the jobs. The benefit of the system is mainly in predicting the failure of the system and thereby finding an optimal and cost-effective solution for provisioning the cloud resources based on the incoming workload.

(Rak et al., 2013)Rak, Cuomo, Villano 2013 proposed techniques which explores the tradeoff between cost and performance of cloud application through the user of benchmarks and simulation. This research work mainly deals with the possibility to predict performance indexes and resource consumption under generic workload conditions. This therefore enables to choose the deployment on the resources of the cloud provider and guarantees the desired performance levels and minimizes the cost for executing the application. As a preparation to the model, mOSAIC framework is used which derives the tags being passed onto the framework through the XML generated from the synthetic load and then benchmark the parameters derived. The second phase involves the optimization phase which takes input from the first preparation phase and is used to predict the resource usage, response time and throughput.

(Strunk, 2013)Strunk 2013 proposed a lightweight mathematical model to concur the energy cost of live migration of an idle VM. The live migration process in Cloud environments for VM s is a key feature which enables High throughput and cost savings through the optimal use of servers and switch off underutilized ones. Based on the profiled data obtained for migration time and power consumption this method utilizes the linear regression technique to deduce the lightweight energy cost module. Experimental results obtained shows that this model can estimate the energy overhead of live migration of Virtual Machines with an accuracy of more than 90%

(Shen et al., 2014)Shen, Chen et al.,2014 proposed a cost optimized resource provisioning model which considers a hybrid approach based on the pricing strategy of different instances having different price and effective leasing strategy to reduce the overall cost. This research work shows that the approach followed achieves the cost saving goal with few SLA breaches and very sparse wastage of resources. The model uses the 1998 World Cup website data which analyzes the workload pattern of the website and how it varies based on different changes in the workload behavior. The algorithm in the research work uses RecentVMQueue which maintains the most current VM amounts during a fixed timeframe. The approach in this research work uses a prediction Algorithm Kit along with AR model for facilitating the workload prediction. Thereafter predicted algorithm workload to VM amount mapping is conducted utilizing the pricing policies of multiple cloud providers to obtain an optimal or Just right cost solutions.

(Ma et al., 2016)Ma, Zhang et al.,2016 proposed a model to solve dynamic resource allocation problem based on the fluctuating load characteristics found in Service based system (SBS) utilizing cloud environment. The research work provides a cost optimized dynamic resource allocation model. Genetic Algorithm (GA) is implemented to solve the dynamic resource allocation model and improve the resolving efficiency problem. Component Service Performance model is used for computation of the performance component service under constant quantity and load conditions. This research work uses the relationship between the resources and service performance considering CPU, Memory, Storage and Bandwidth.

(Aldossary and Djemame, 2018)Aldossary and Djemame, 2018 proposed a performance and energy-based cost prediction framework to calculate the total cost of VM auto scaling by maintaining expected performance levels and considering the various aspects of resource usage and power consumption. Cloud Resource Prediction and Provisioning Schema (RPPS) based on ARIMA based model is presented. The model automatically predicts the future demand in terms of CPU usage of VMs and proactively performs resource provisioning for cloud-based applications workload. The prediction error on an average is on the lower side with less than 10% in most case.

(Humphrey, 2010)Humphrey, Mao, Li 2010 proposed a mechanism which is used to scale computing instances in the cloud based on workload information and desired performance characteristics. The mechanism enables the submitted jobs to be completed before the SLA breach occurs and at the same time reduces the cost by controlling the underlying instance numbers. Additionally, the mechanism helps in terms of VM instance start up and shut down activities thereby controlling them and allowing for cost savings for the instances.

(Aldossary et al., 2019)Proposed a cost aware prediction framework to estimate the cost associated with virtual machines based on their usage and power consumption. The research work proposes the framework to be divided into the parts: 1. VM workload prediction using Auto regressive Integrated Moving Averages (ARIMA) Model 2. PM Workload Prediction 3. PM energy consumption prediction using regression models, 4.VM Energy Prediction and 5.VM total cost estimation. Overall, the research work was able to establish a prediction framework that can predict the resource usage, power consumption, and estimate the total cost for the VMs during the operation of cloud services along with the optimizations to reduce the cost factor associated

(Singh et al., 2019)Singh, Gupta and Jyoti 2019 proposed an adaptive prediction model using linear regression, Auto regressive Integrated Moving Averages (ARIMA) and Support Vector Regression for web applications. This research work first deals with the problem of workload prediction based on workload features. Final prediction mechanism shows significant improvement in the root-mean-squared error and mean absolute percentage error. This model was also efficient in predicting the seasonal and non-seasonal variations seen within the workload patterns.

(Djemame et al., 2017)Djemame, Bosch et al.,2017 proposed a model wherein the Paas and Iaas components of Cloud based providers co-ordinate and share information successfully in terms of the workload and compute behaviors leading to increasing adaptability to changing needs for workloads, energy, and resource. The energy modeler utilized in this research work forecasts the power consumption using neural networks and estimates VM consumption using VM-observed data.

(Cortez et al., 2017)Cortez, Bonde, Muzio et al.,2017 proposed a prediction of Microsoft Azure workload behavior including key characteristics of VMs and how they can be predicted for future workloads. Based on these a novel Resource Central (RC) is introduced which collects the VM traces and learns the behaviors and provides predictions to various resource managers through a client-side library.

(Lansky et al., 2021)Lansky, Ali et al.,2021 proposed a comparative study of various Deep learning techniques that can be applied to successfully and accurately predict the various intrusions that can take place in computer systems and networks. This research work does an in-depth analysis of the various deep learning techniques and their advantages for any given problem statement. The research work starts with the classification of the IDS schemes according to the incorporated deep learning techniques and highlights how each scheme is trying to apply deep learning methods to recognize and predict the different types of intrusions that can happens to systems. The research work involves deep dive into: Auto Encoder based schemes, Restricted Boltzmann Machine based schemes, Recurrent Neural network-based schemes, and Convolutional Neural network-based schemes.

(Qu, 2018)Qu, Chiayi et al.,2018 proposed a deep learning approach to solve the problem of achieving fault detection with signal analysis and processing based on spare auto-encoder and is used to automatically extract features of complex data sets to detect fault. The auto encoder mechanism thereby can be used to extract features from signals which are not recognized and bring out intelligent identification of the features. The sparse encoder used in this research work is one with 1 hidden layer and 1 visible layer and is effective in detecting faults from the range of 60% to 70% accuracy. Furthermore, hidden layers can be introduced to achieve even more accuracy to the system.

(Calzarossa et al., 2019b)Calzarossa et al., 2019 proposed a technique which considers a combination of various workload characterization techniques for modelling and predicting workload access patterns. The research work starts with the identification workload parameters by exploratory analysis using a combination of statistical and visualization techniques. This is followed by a cluster analysis done and the result used as part of Principal Component Analysis (PCA) to provide the classes of clusters, in this case the workload characteristics. As part of future workload prediction behavior, the research work uses a fundamental timeseries analysis. For better understanding of the timeseries model and improve its forecast, a classical decomposition based on an additive model including deterministic and stochastic components is used.

(Ding et al., n.d.)Ding proposed a temporal learning algorithm approach to deduce four metrics to measure the efficiency of the provisioned cloud resources and services including resource utilization, instance utilization, cost utilization, and cost efficiency. The research work further uses the temporal learning algorithm to predict their future values and to detect anomalies in the system. Additionally, the research work uses an Autoregressive model to predict future demands of applications under normal situation. The proposed model automatically monitors the cloud environment in real time and identify potential opportunities for cost and resource optimizations.

(Wan et al., 2020)Wan et al.,2020 proposed a model which deals with the problem of VM running for a longer duration of time and classification of such VMs in runtime leading to dynamic resource allocation adjusting the VMs thereby leading to performance cost optimization of the cloud platform. This research work uses a combination of queueing theory and optimization methods to tackle the problem of system resource allocation. Further, the research work uses the queueing theory to provide a stable probability distribution of the system and provides theoretical support for multifaceted optimization work. For the pricing distribution, this research work considers various factors affecting the pricing and adopts a policy of execution of isomorphic tasks such as map reduce tasks. Overall, the research work divides the tasks into 4 phases: Phase 1 deals with the Queueing Model followed by a QBD Process Model to calculate the quantitative cost of the various states of the VMs.Phase 2 of the model deals with System level parameter measures Like performance, Cost, and elasticity (implemented though the probability distribution implementation). Phase 3 Deals with Multi-Objective Optimization Model (MOO) which is responsible for cost-prediction function and nominal time delay function and finally all the inputs from Phase1 to Phase 3 converge into the Phase 4 which proposes a cost-performance Optimization Algorithm

(Minarolli and Freisleben, 2014)Proposed a machine learning approach to a cross-correlation prediction model to predict resource demands of multiple resources of virtual machines in a cloud environment. The results of the prediction model help in proactive resource allocation schemes that assigns only the optimal resources to VMs there being cost effective. This research work introduces a concept called Automatic Proactive Resource Allocation (APRA) that proactively allocated resources to a VM by predicting the usage using Support Vector Machine (SVM) for time series forecasting. Thereafter a cross-correlation prediction mechanism is used to predict the scenario of multiple resources of a multi-tier application.

(Farahnakian et al., 2013)proposed an algorithm for consolidation of dynamic virtual machine instances to decrease the number of active physical servers on a data center and hence reduce the cost overall. The proposed model uses k- nearest neighbor regression algorithm to predict the usage of resources. Based on this prediction the methodology can determine when the host becomes under - utilized or over-utilized. The experimental results show that the methodology proposed is efficient in maintaining the SLA requirement as well.

2.3. Comparison of the various algorithms used

Table 2.3 1: Comparison of Algorithms used in the various Research Work

Reference Paper	Algorithm used																																							
	NB	KDE	MCDM	Kmeans	ANP	PDRV	AR	MA	NF	LR	BRS	HFA	AC	ADP	SVM	CECRPICAO	MPNN	OCRP	SIP	BD	SAA	KF	DblSes	MC	GA	PSO	ACO	SAM	RPPS	ARIMA	RC	AE	RBM	RNN	CNN	PCA	MOO	QBD	KNN	ARPA
(Al-Faifi et al., 2018)	☑	☑																																						
(Al-Faifi et al., 2019)			☑	☑	☑																																			
(Calzarossa et al., 2019a)						☑																																		
(Di et al., 2012)	☑						☑	☑	☑																															
(Yang et al., 2014)										☑																														
(Barnwal et al., 2019)											☑	☑	☑																											
(Biswas et al., 2015)										☑					☑																									
(Wang et al., 2013)														☑																										
(Yousefyan et al., 2013)																☑	☑																							
(Chaisiri et al., 2012)																		☑	☑	☑	☑																			
(Mark et al., 2011)																						☑	☑	☑	☑	☑	☑													
(Jagannatha et al., 2017)																								☑				☑												
(Shen et al., 2014)																														☑										
(Ma et al., 2016)																									☑															
(Aldossary and Djemame, 2018)																													☑	☑										
(Aldossary et al., 2019)																														☑										
(Singh et al., 2019)															☑															☑										
(Cortez et al., 2017)																															☑									
(Lansky et al., 2021)																																☑	☑	☑	☑					
(Qu, 2018)																																☑								
(Calzarossa et al., 2019b)																																				☑				
(Wan et al., 2020)																																					☑	☑		
(Minarolli and Freisleben, 2014)															☑																									☑
(Farahnakian et al., 2013)																																							☑	

2.4. Summary of Issues observed

From the study of the research done earlier I have concluded that several machine learning algorithms have been used to predict the cloud cost resource optimized model with different levels of effectiveness, basically because there are flaws within, which are difficult to visualize and predict. The issues are listed below.

The datasets taken for the various research work do not have a standard set of features that can be taken into consideration. Very often even after taking the requisite number of features into consideration there are vulnerabilities and uncertainties in the prediction which arise out of the not being able to accurately predict the behavior of the workloads

It is seen that incase of large number of research works submitted earlier, have taken only a subset of features due to small dataset selection based on which they have went ahead and tried to predict the cost effectiveness of the resources in the cloud environment. When the same algorithm is applied in real time loads, then the efficiency would decrease drastically as they have not considered all the scenarios which can arise out of the real time workload characteristics as the datasets in such cases are much bigger and complex leading to scenarios which arise and have not been seen earlier with a smaller dataset

The task of coming selecting the suitable Machine learning Algorithm which caters to all possible combination of scenarios while implementing a Cloud Cost effective resource optimized ML based model for workloads considering all attributes is still a daunting one.

3. Research Methodology

3.1. Introduction

The research methodology adopted in the research work consists of a publicly available Microsoft Azure Dataset. The dataset undergoes various data cleaning stages. The data for the pricing of the various Azure instances would be taken from the publicly available pricing information on Azure Pricing calculator website. Additional care would be taken to ensure that the data is balanced and only the balanced data is used in the model. Once we ensure data readiness, we will use appropriate machine learning and neural network-based models. The outcomes from the models would then be subjected to a comparative study and based on the performance metrics on various parameters, the best and appropriate model would be selected.

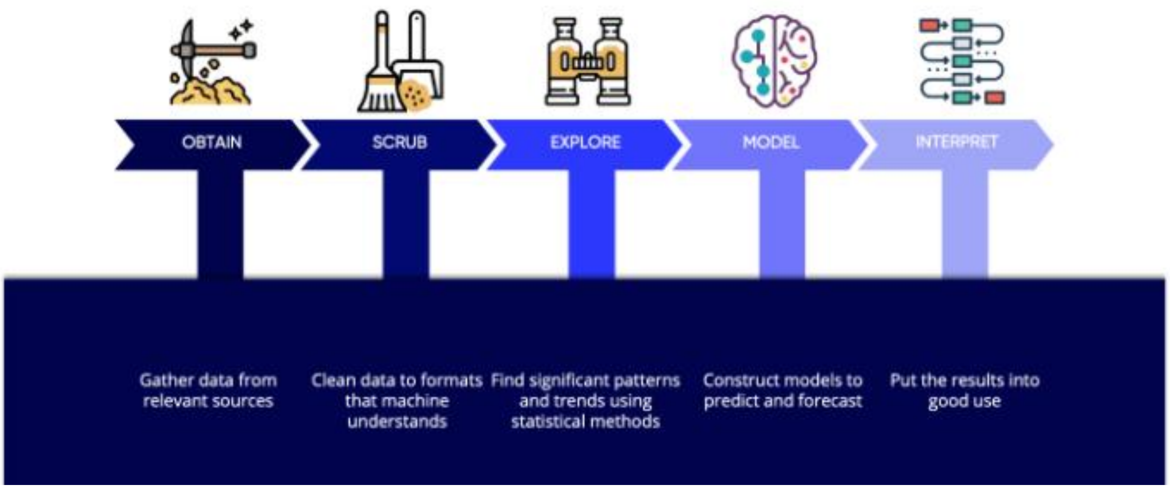

OBTAIN
SCRUB
EXPLORE
MODEL
INTERPRET
Gather data from relevant sources
Clean data to formats that machine understands
Find significant patterns and trends using statistical methods
Construct models to predict and forecast
Put the results into good use

Figure 3.1: High Level Research Methodology

3.2. Research methodology

3.2.1. Data selection

The dataset which has been selected to be utilized in this research work contains Azure VM traces. The Main characteristic of the dataset is as below.

Main Characteristics of V1

1) Size of the dataset
2) Number of files present
3) Observed number of VMs
4) Number of Azure Subscriptions available
5) Timeseries data consisting of virtual machine CPU utilization readings
6) Virtual machine information table

3.2.2. Data preprocessing

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. This preprocessing step ensures that data is ready and without any skewness for further consumption in the subsequent steps of the research methodology. The Primary Components are as below:

1) Removal of Missing values – Data cleaning should include removal of missing values.
2) Removal of irrelevant data – Remove Noisy or irrelevant data for the future steps
3) Removal of Duplicates – This step is very important to ensure that modeling exercise takes clean data as input.
4) Fix data type – Any conversion required for data type mismatches should be corrected
5) Clean Data - Imbalance and skewness should be checked and corrected.

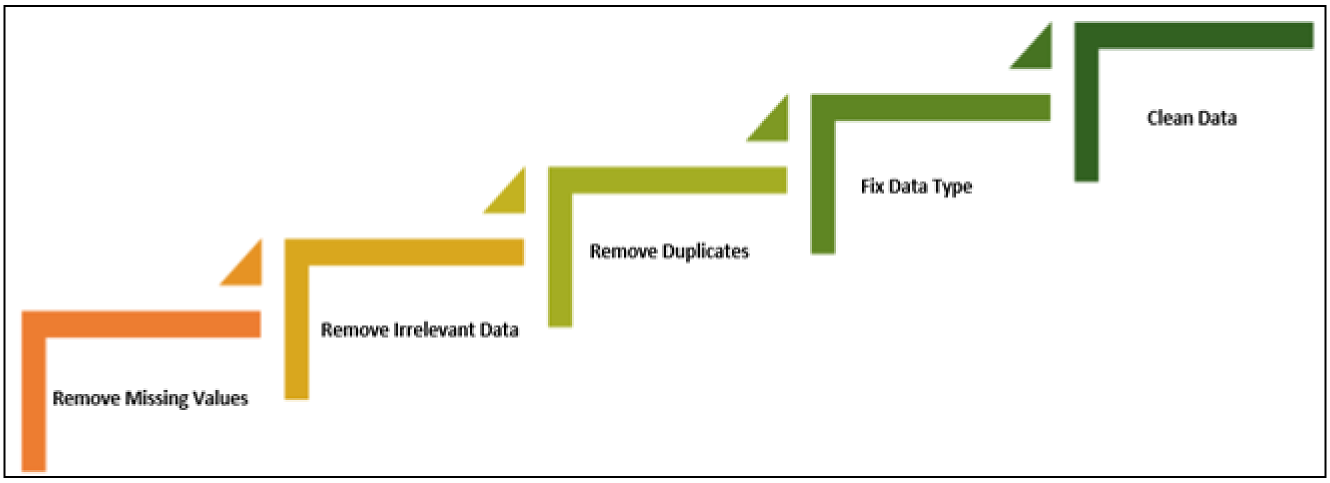


Figure 3.2: Data Cleanup Pre-process

3.2.3. Exploratory Data Analysis and Visualizations

A large section of EDA is performed using visualization and graphical methods which is used for analysis and investigation of acquired data sets.EDA also helps in summarizing the main characteristics observed within the dataset selected. This process is beneficial in identification, implementation and subsequent manipulation of datasets to produce the intuitive insights on the dataset, unearth patterns, spot discrepancies, test hypothesis and analyze and infer assumptions.

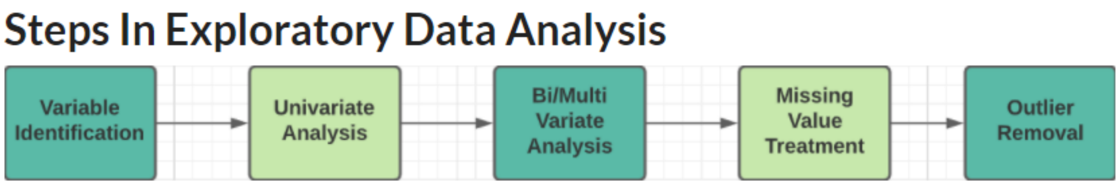
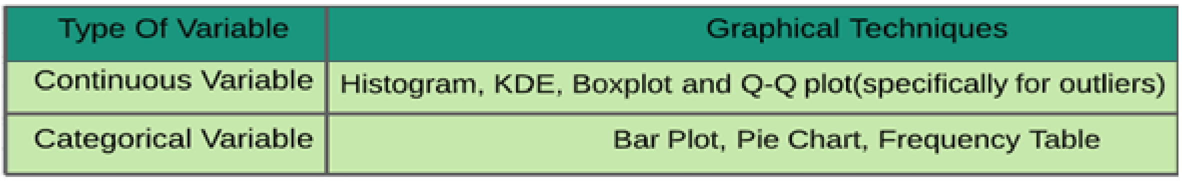


Figure 3.3: Exploratory Data Analysis

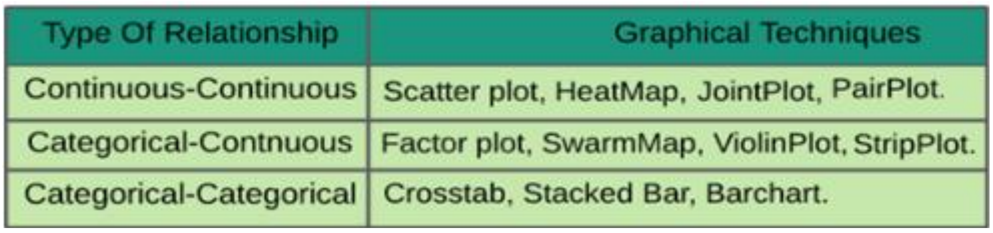
1) The concentration of data for categorical variables can be shown using a graphical image also known as a box plot
2) Sets of Continuous data can be described visually very effectively by a basic bar graph known as Histogram.



Type Of Variable	Graphical Techniques
Continuous Variable	Histogram, KDE, Boxplot and Q-Q plot(specifically for outliers)
Categorical Variable	Bar Plot, Pie Chart, Frequency Table

Figure 3.4: Graphical Techniques for EDA - Univariate

In a classic Bivariate Analysis, we study the relationship between any two variables which can be categorical-continuous, categorical-categorical, or continuous-continuous



Type Of Relationship	Graphical Techniques
Continuous-Continuous	Scatter plot, HeatMap, JointPlot, PairPlot.
Categorical-Contnuous	Factor plot, SwarmMap, ViolinPlot, StripPlot.
Categorical-Categorical	Crosstab, Stacked Bar, Barchart.

Figure 3.5: Graphical Techniques for EDA - Bivariate

3.2.4. Feature Engineering

Feature engineering is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data. Appropriate features to be created as part of this step which would be helpful in the final building of the predictive model for the cloud cost optimized model from the VM traces.

3.2.5. Model Selection, Building & Evaluation

This research work intends to build various models and analyze the results from the various factors affecting cloud workloads and how cost-effective approach can be brought in. It also identifies the most efficient and predictable model which provides accurate and precise results while maintaining SLA levels and expected performance factors. The below is the process flow of the various evaluations that the models would be undergoing to arrive at a suitable model.

1) Review and analyze the model by comparing with the statistical models
2) Analyze and conduct checking's to identify Variance Inflation Factors
3) Analyze the plot of Receiver Characteristic Curve or ROC
4) Determine the optimal point which can be considered as a cut off
5) Determine if there is a precision versus recall tradeoff seen
6) Verify the performance of the classification using a Confusion Matrix
7) Determine and estimate the overall accuracy and performance parameters

3.3. Proposed Methods: Model Building and Evaluation

3.3.1. Support Vector Machines (SVM)

Support vector machines (SVMs) are a set of supervised learning method/algorithms used for classification, regression, and detection. The objective of the model is to find a hyperplane in N dimensional space that can classify the data points. The primary advantages with this method/Algorithm can be described as below:

1) Comparatively found to be highly efficient in case where there is high dimensional space observed.
2) It does not lose its effectiveness in scenarios where thequantity of the total number of dimensions present is more than the number of samples available.
3) Found to be highly memory efficient in as it uses a subset of training points which is provided as an input to the decision function also known as support vectors.
4) The SVM method/Algorithm is also observed to be highly versatile since it can have varied kernel functions and those can be effectively mentioned for the decision function to be consumed. Custom kernels, however, are not possible and only generic kernels are given.

Some of the drawbacks associated to support vector machines include:

1) Over-fitting should always be avoided during the process of selecting the kernel functions specially in cases where the number of features is more in number compared to the samples. In such cases regularization term should be considered as a mandatory best practice as it is very critical to the overall outcome.
2) Support Vector Machines uses a very costly five-fold cross validation as they do not provide a means for probability estimates.

3.3.2. Time Series Forecasting

Time series analysis involves building models that can efficiently capture or describe an observed time series to understand the underlying causes. This is nothing but answers the question of “why” when taking into consideration the dataset which has the time series data.The methodology includes decomposition of the dataset into subsequent components with assumptions being made about the composition of the data and includes models like LSTM. The overall effectiveness and performance of the model is measured by how the whole dataset is being interpreted and represented. Thereby resulting in better interpretation of the problem at hand. The primary objective of time series forecasting analysis can be described as below:

1) The main goal is a better understanding of the phenomenon which is getting represented by the time series dataset
2) Forecasting or predicting the future values of the observed time series datasetwhich is built on the observation and modelling of the historical time series data

3.3.3. Deep Learning

Depp learning is an integral segment of the extended family of Machine learning. It can be further classified as supervised/unsupervised artificial neural networks. In today's world we have extensive usage deep learning paradigms like RNN, CNN, Deep reinforcement learning etc. which finds usage in the areas of speech recognition, computer vision, handwriting recognition, Image analysis in medical science etc. Normally a neural network with a one layer can be used to make predictions, however, more than one hidden layer(s) can be used additionally to get better optimization and accurate results.

Back propagation in neural networks training is quite common and techniques like gradient descent, can be used to fine tune weights of neural networks based on the errors obtained in previous iterations. Accurate and efficient tuning of the weights would result in low error rates thereby resulting in a better and generic model which increases the reliability.

3.3.4. Auto Encoders

An autoencoder is defined as an artificial neural network used for unsupervised learning of efficient coding. In simple words, autoencoders are specific type of deep learning architecture used for learning representation of data, typically for the purpose of dimensionality reduction. This is achieved by designing deep learning architecture that aims that copying input layer at its output layer. This research work proposes an auto encoder approach to reduce the dimensionality of the VM traces dataset so that accurate and precise prediction can be made.

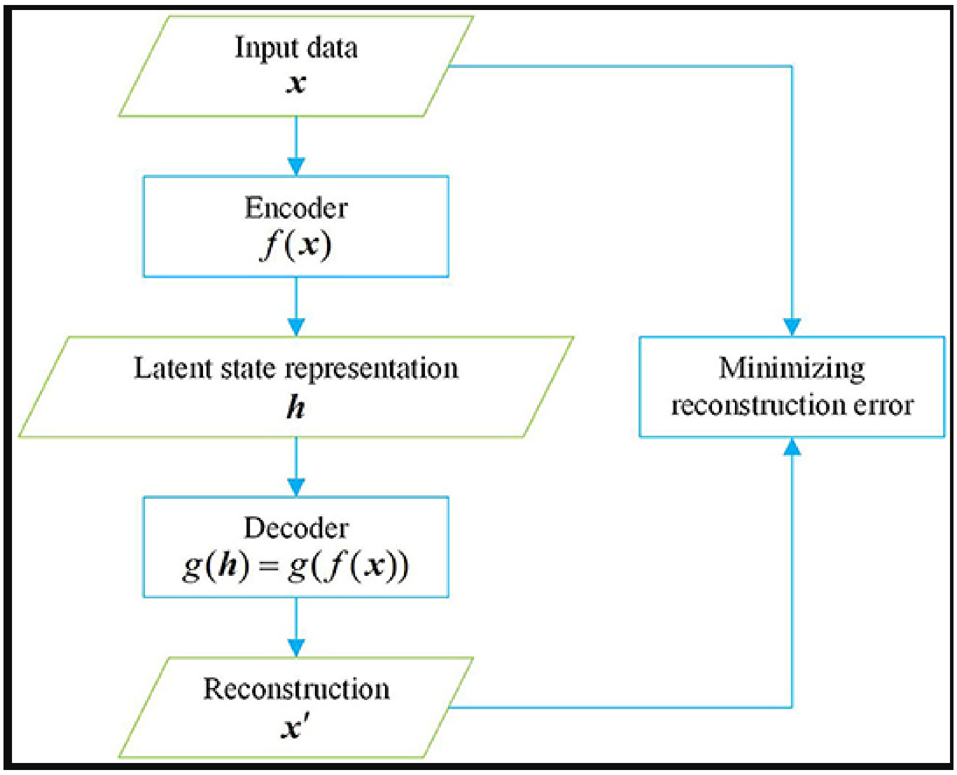


Figure 3.6: Auto Encoder Process Flow

3.3.5. KNN (K-Nearest Neighbor)

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1) Ease to interpret output
2) Calculation time
3) Predictive Power

Implementation of KNN involves the below steps:

1) Import the required Libraries
2) Import the dataset and make sure that the value of k is initialized
3) Split the dataset in train and test
4) This step would be used to get the predicted class. For this purpose, iteration would be required from one to total number of training data points
   a) The distance from test data to each row of training data need to be calculated.
   b) Euclidean distance will be used to calculate the distance metric. Similarly, other metrics that can be used are Cosine and Chebyshev. The generic Formula as below:

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

   c) Based on the distance values the calculated distance needs to be sorted accordingly.
   d) From the sorted distances array, we need to take the top k rows
   e) Determine and extract the frequently occurring classes of these rows
   f) After the extraction we can get the predicted class

3.3.6. Best practices for KNN

1) Missing Data: Addressing the issues of missing data is critical for success of KNN algorithm as the quantitative distance cannot be calculated for missing data. Hence data samples taken as part of this research work cannot be calculated. Therefore, requisite cleaning techniques should be adopted to clean or impute missing data.
2) Data Rescaling: It is always advisable to standardize the datasets and KNN performs exceptionally well with normalized data.
3) Dimensionality reduction of data: It is always advisable to select the variables based on feature selection and remove unwanted variables who do not contribute to the overall outcome.

3.3.7. Random forest

Random forest is an algorithm which is classified as a supervised learning algorithm and is a collection of several decision trees, usually trained with bagging method. This algorithm is mostly used to solve classification and regression problems. Bagging method is based on the premise that the accuracy and efficiency of the results is substantially increased when a combination of learning models is being used. This algorithm uses a voting process and selects the best outcomes after taking into consideration the predictions from all trees. This method precisely and efficiently handles nonlinearity through implementation of correlation between features.

3.3.8. LSTM

LSTM is a special type of recurrent neural network which has the unique ability to learn long term dependencies available in the dataset. There are 4 layers which interact with each other together forming the recurring module of the model. This model has the unique capability to selectively unlearn, learn and retain critical information from each unit which is facilitated with the help of 3 gates and has a cell state available. The number of steps or inputs are defined by the first layer. After defining the number of inputs, the model would be run against an optimizer, and this would create a sequence of values leading to the training of the LSTM model. Post training, the predictions can be made against the test data selected.

3.3.9. Gated Recurrent Unit (GRU)

The GRU can be called as an improved version of RNN as it no longer has the vanishing gradient problem faced by RNN. This happens through the "update gate" and "reset gate" functionality provided by GRUs. The two vectors are the deciding authority of what data would be passed as an output of the GRU model. What makes the GRU special is that they can retain information from a long time.

3.3.10. Expected Outcomes

1) A cost effective and resource optimized cloud model/algorithm having optimized resource provisioning with good performance attributes is expected from the research activities conducted as per the research work
2) This research work also intends to do an in-depth analysis of the various factors that characterize the different workloads along with the cost implications.
3) The research work intends to also take into consideration the parallel execution of workloads along with scaling parameters which yield maximum efficiency to the predictive model

3.4. Summary

As part of the research work, I intend to build various models which would explain the correlation between different diverse parameters of the cloud ecosystem. This insight would be helpful in maintaining the cost effectiveness of the cloud solution as a whole and infrastructure cost when catering to the needs of various applications and Jobs. The model which best demonstrates the capabilities that I am looking for would be selected after the experiments on the dataset selected.

4. Analysis

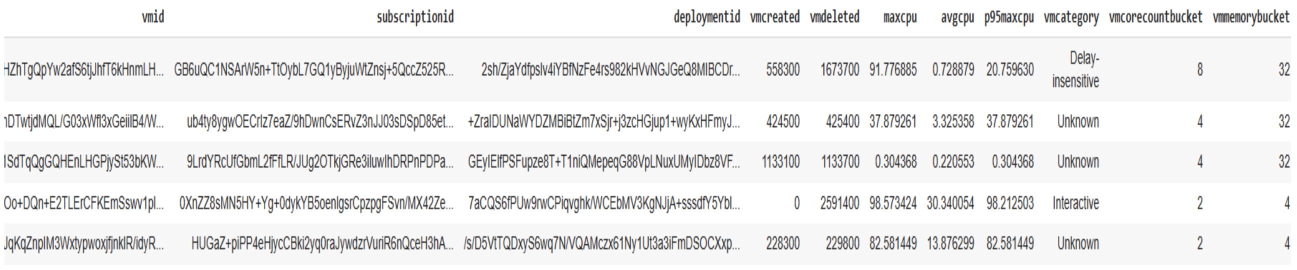
4.1. Introduction

The dataset selected to conduct the research has been collected from a public source and contains the Microsoft Azure traces having the Virtual Machine representative traces and Azure Function traces. Additionally, for the pricing details I have taken the Azure Pricing from the online Azure Pricing calculator for the real time pricing of Microsoft Azure components and Virtual machines with different configurations sizes in line with the configuration of the components provided in the traces.

4.2. Datasets Used

This section of the research work contains in depth details of the dataset used to conduct the research and their characteristic features available to us for use. When we talk about the results of an algorithm, the data set we select is very important factor which can decided the course of the entire research work. Hence it is important that the data set is selected wisely to derive maximum value out of the data in terms of insights it provides in an efficient manner.

As mentioned in the introduction section, the dataset selected for the dissertation work has been taken from a public source and contains traces from Microsoft Azure machines 2019 and would help to unearth the various resource and costing aspects when dealing with the optimized Cloud infrastructure usage. I have also taken the VM hourly stats for the server and infrastructure usage components. I have taken the pricing information from Microsoft pricing calculator for the specific combination of memory and cores and applied them to get the cost information. The dataset columns for the vm table can be seen as below:



vmid	subscriptionid	deploymentid	vmcreated	vmdeleted	maxcpu	avgcpu	p95maxcpu	vmcategory	vmcorecountbucket	vmmemorybucket
HZhTgQpYw2afS6tjJhfT6kHnmLH...	GB6uQC1NSArW5n+TtOybL7GQ1yByjuWtZnsj+5QccZ525R...	2sh/ZjaYdfpslv4iYBfNzFe4rs982kHVvNGJGeQ8MIBCDr...	558300	1673700	91.776885	0.728879	20.759630	Delay-insensitive	8	32
nDTwtjdMQL/G03xWfl3xGeiilB4/W...	ub4ty8ygwOECrlz7eaZ/9hDwnCsERvZ3nJJ03sDSpD85et...	+ZraIDUNaWYDZMBiBtZm7xSjr+j3zcHGjup1+wyKxHFmyJ...	424500	425400	37.879261	3.325358	37.879261	Unknown	4	32
ISdTqQgGQHEnLHGPjySt53bKW...	9LrdYRcUfGbmL2fFfLR/JUg2OTkjGRe3iluwIhDRPnPDPa...	GEyIElfPSFupze8T+T1niQMepeqG88VpLNuxUMyIDbz8VF...	1133100	1133700	0.304368	0.220553	0.304368	Unknown	4	32
Oo+DQn+E2TLErCFKEmSswv1pl...	0XnZZ8sMN5HY+Yg+0dykYB5oenlgsrCpzpgFSvn/MX42Ze...	7aCQS6fPUw9rwCPiqvghk/WCEbMV3KgNJjA+sssdfY5Ybl...	0	2591400	98.573424	30.340054	98.212503	Interactive	2	4
JqKqZnpIM3WxtypwoxjfjnklR/idyR...	HUGaZ+piPP4eHjycCBki2yq0raJywdzrVuriR6nQceH3hA...	/s/D5VtTQDxyS6wq7N/VQAMczx61Ny1Ut3a3iFmDSOCXxp...	228300	229800	82.581449	13.876299	82.581449	Unknown	2	4

Figure 4.2 1: Dataset column summary

The schema of the vm table file contains

Table 4.2 1: Schema of the selected dataset

Column	Definition
vmid	virtual machine id
subscriptionid	Azure Subscription Id to which the vm belongs to
deploymentid	Unique Id for which the deployment was done
vmcreated	timestamp when the vm got created
vmdeleted	timestamp when the vm got deleted
maxcpu	Maximum value of CPU usage
avgcpu	Average Value of CPU usage
p95maxcpu	95 percentile data for max cpu. It's a stress testing parameter for CPU readings

vmcategory	Type of vm
vmcorecountbucket	No of cores used by the jobs spun by the vm
vmmemorybucket	memory being used by the jobs spun by the vm

As part of the thesis work, I have also considered an additional vm trace file with the below description:

Table 4.2 2: Schema of the vm trace file

Column	Definition
Timestamp	Timestamp
vm id	Virtual machine id
min cpu	Minimum value of CPU usage
max cpu	Maximum value of CPU usage
avg cpu	Average value of CPU usage

As part of thesis work, I have taken the compute and core pricing from publicly available pricing from azure pricing portal and then mapped the same to the traces dataset to compute the cost such as pay as you go cost, one-year reserved cost, three-year reserved cost and spot cost. Please find the table containing the cost information used in my analysis of the cost implications on the various parameters of the trace's dataset.

Table 4.2 3: Azure Pricing Information from Microsoft

Instance	vCPU	RAM	Temporary	Pay as you go	1 year	3 year	Spot
F1	2.00	2.00	8.00	$0.05	$0.03	$0.02	$0.02
F2s v2	2.00	4.00	16.00	$0.10	$0.06	$0.04	$0.04
D2d v4	2.00	8.00	75.00	$0.11	$0.07	$0.04	$0.04
G1	2.00	32.00	384.00	$0.49	$0.01	$0.01	$0.05
F4	4.00	8.00	64.00	$0.20	$0.12	$0.08	$0.08
E4ads v5	4.00	32.00	150.00	$0.26	$0.15	$0.10	$0.10
D8ds v4	8.00	32.00	300.00	$0.45	$0.27	$0.17	$0.18
E8d v4	8.00	64.00	300.00	$0.58	$0.34	$0.22	$0.12
D16s v3	24.00	64.00	128.00	$0.77	$0.46	$0.29	$0.31
F32s v2	32.00	64.00	256.00	$1.35	$0.80	$0.50	$0.28

4.3. Preparation of Data

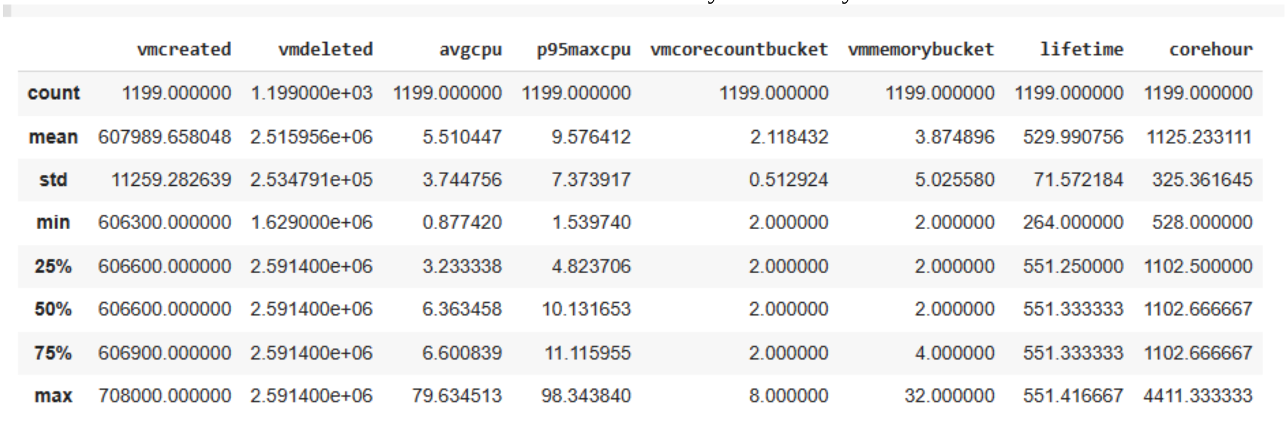
The preparation of data is the first step towards building various models based on the prepared data. That is why this step is one of the most important steps that would help us build out the various algorithms and select a suitable model which would produce optimal and efficient results.

This section has been divided into various stepwise sections which are essential building blocks to a clean data. The main steps are data analysis, data cleaning, data interpretation, data co-relations data splitting and data validation.

4.3.1. Data Analysis

The vm table traces data set contains 2695548 rows and 13 columns. The overall dataset was divided as per the vm category and the interactive vms had the below statistics:

Table 4.2 4: Dataset Analysis summary

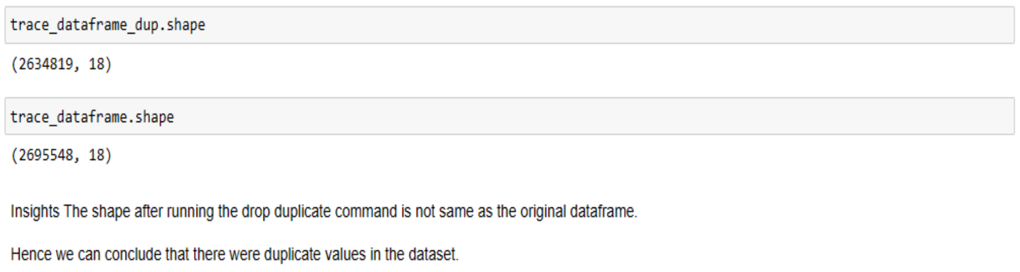


	vmcreated	vmdeleted	avgcpu	p95maxcpu	vmcorecountbucket	vmmemorybucket	lifetime	corehour
count	1199.000000	1.199000e+03	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000	1199.000000
mean	607989.658048	2.515956e+06	5.510447	9.576412	2.118432	3.874896	529.990756	1125.233111
std	11259.282639	2.534791e+05	3.744756	7.373917	0.512924	5.025580	71.572184	325.361645
min	606300.000000	1.629000e+06	0.877420	1.539740	2.000000	2.000000	264.000000	528.000000
25%	606600.000000	2.591400e+06	3.233338	4.823706	2.000000	2.000000	551.250000	1102.500000
50%	606600.000000	2.591400e+06	6.363458	10.131653	2.000000	2.000000	551.333333	1102.666667
75%	606900.000000	2.591400e+06	6.600839	11.115955	2.000000	4.000000	551.333333	1102.666667
max	708000.000000	2.591400e+06	79.634513	98.343840	8.000000	32.000000	551.416667	4411.333333

There are totally 3 vm categories: 1. Interactive 2. Delay-Insensitive and 3. Unknown. So,I have divided the dataset as per the category to get the insights pertaining to each category and then predict the maximum cpu and p95maxcpu parameters required for future workloads so that appropriate scheduling and provisioning can take place. Additionally, the cost information taken from Azure Pricing portal has been utilized to predict the parameters which effects the cost and hence increase in those parameters leads to increase in cost.

4.3.2. Data Cleaning

I had analyzed the vm table data and found that there were duplicates which existed in the dataset. This required the data to be cleaned up.



```
trace_dataframe_dup.shape
```

```
(2634819, 18)
```

```
trace_dataframe.shape
```

```
(2695548, 18)
```

Insights The shape after running the drop duplicate command is not same as the original dataframe.

Hence we can conclude that there were duplicate values in the dataset.

For the second part of my analysis of data the vm category of unknown was found to be redundant and hence those records had to be removed from the dataframe

4.3.3. Data Co-Relations

The vm traces data that I have taken was found to have the below correlations with respect to the various vm parameters present in the trace's dataset

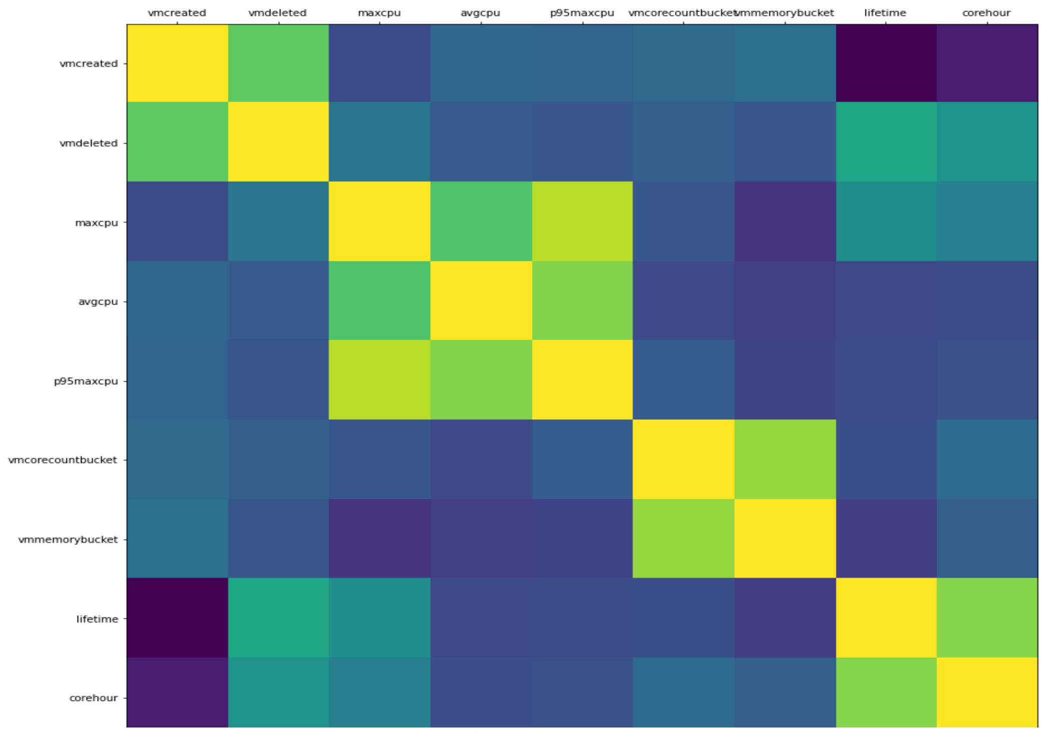
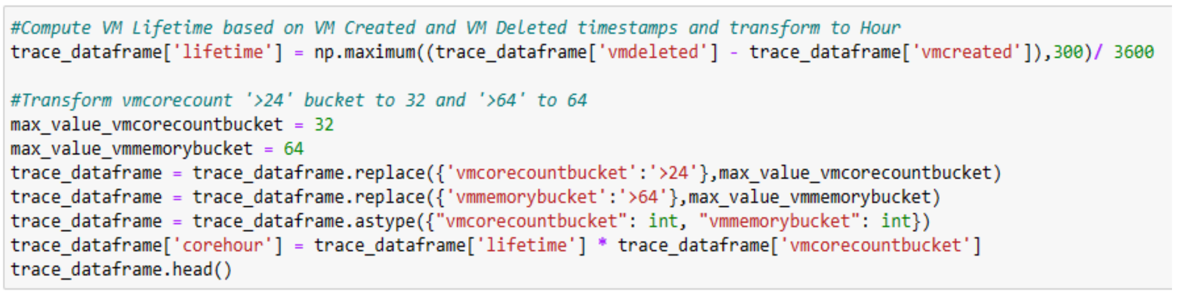


Figure 4.3 1: Heatmap of the dataset showing co-relations

4.3.4. Feature Engineering

The below feature engineering techniques were used to produce additional parameters of the associated data and make the data interpretable

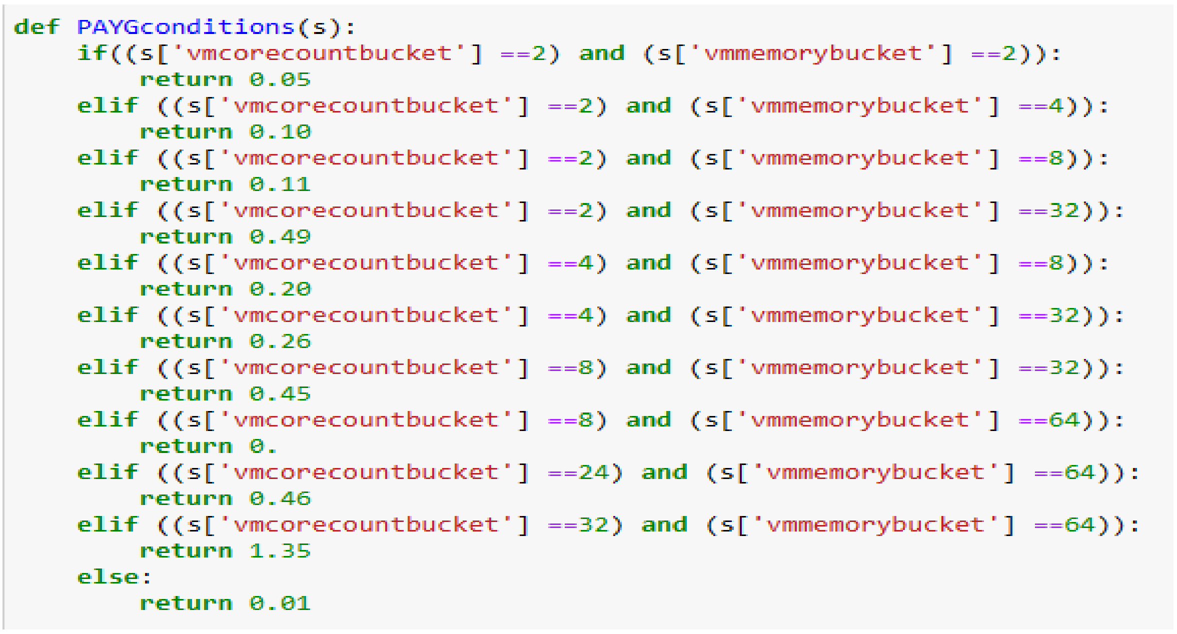
1) Lifetime, vmcorecount and corehour



```
#Compute VM Lifetime based on VM Created and VM Deleted timestamps and transform to Hour
trace_dataframe['lifetime'] = np.maximum((trace_dataframe['vmdeleted'] - trace_dataframe['vmcreated']),300)/ 3600

#Transform vmcorecount '>24' bucket to 32 and '>64' to 64
max_value_vmcorecountbucket = 32
max_value_vmmemorybucket = 64
trace_dataframe = trace_dataframe.replace({'vmcorecountbucket':'>24'},max_value_vmcorecountbucket)
trace_dataframe = trace_dataframe.replace({'vmmemorybucket':'>64'},max_value_vmmemorybucket)
trace_dataframe = trace_dataframe.astype({"vmcorecountbucket": int, "vmmemorybucket": int})
trace_dataframe['corehour'] = trace_dataframe['lifetime'] * trace_dataframe['vmcorecountbucket']
trace_dataframe.head()
```

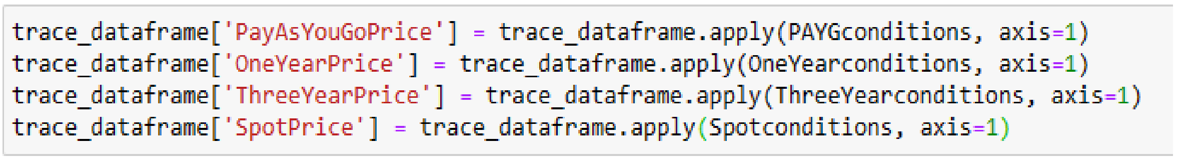
2) PayAsYouGoPrice, OneYearPrice, ThreeYearPrice and SpotPrice

Create a python Method to calculate the price based on the Microsoft provided table and map that method to create a new column within the original dataset



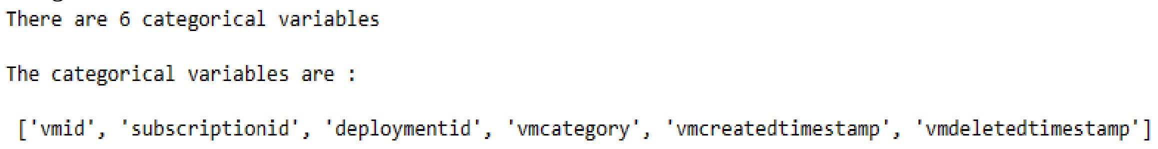
```
def PAYGconditions(s):
    if((s['vmcorecountbucket'] ==2) and (s['vmmemorybucket'] ==2)):
        return 0.05
    elif ((s['vmcorecountbucket'] ==2) and (s['vmmemorybucket'] ==4)):
        return 0.10
    elif ((s['vmcorecountbucket'] ==2) and (s['vmmemorybucket'] ==8)):
        return 0.11
    elif ((s['vmcorecountbucket'] ==2) and (s['vmmemorybucket'] ==32)):
        return 0.49
    elif ((s['vmcorecountbucket'] ==4) and (s['vmmemorybucket'] ==8)):
        return 0.20
    elif ((s['vmcorecountbucket'] ==4) and (s['vmmemorybucket'] ==32)):
        return 0.26
    elif ((s['vmcorecountbucket'] ==8) and (s['vmmemorybucket'] ==32)):
        return 0.45
    elif ((s['vmcorecountbucket'] ==8) and (s['vmmemorybucket'] ==64)):
        return 0.
    elif ((s['vmcorecountbucket'] ==24) and (s['vmmemorybucket'] ==64)):
        return 0.46
    elif ((s['vmcorecountbucket'] ==32) and (s['vmmemorybucket'] ==64)):
        return 1.35
    else:
        return 0.01
```

Call the method to create a new column in the dataframe with updated pricing information



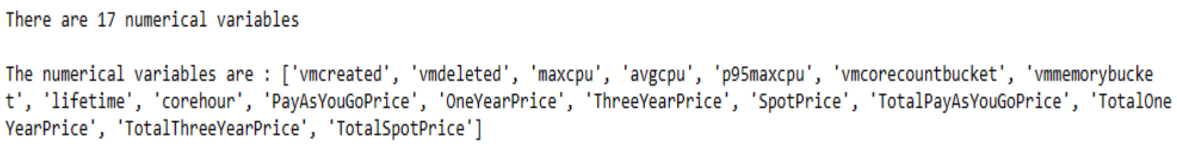
```
trace_dataframe['PayAsYouGoPrice'] = trace_dataframe.apply(PAYGconditions, axis=1)
trace_dataframe['OneYearPrice'] = trace_dataframe.apply(OneYearconditions, axis=1)
trace_dataframe['ThreeYearPrice'] = trace_dataframe.apply(ThreeYearconditions, axis=1)
trace_dataframe['SpotPrice'] = trace_dataframe.apply(Spotconditions, axis=1)
```

Categorical Variables:



```
There are 6 categorical variables

The categorical variables are :

 ['vmid', 'subscriptionid', 'deploymentid', 'vmcategory', 'vmcreatedtimestamp', 'vmdeletedtimestamp']
```

Numerical Variables:



```
There are 17 numerical variables

The numerical variables are : ['vmcreated', 'vmdeleted', 'maxcpu', 'avgcpu', 'p95maxcpu', 'vmcorecountbucket', 'vmmemorybucke
t', 'lifetime', 'corehour', 'PayAsYouGoPrice', 'OneYearPrice', 'ThreeYearPrice', 'SpotPrice', 'TotalPayAsYouGoPrice', 'TotalOne
YearPrice', 'TotalThreeYearPrice', 'TotalSpotPrice']
```

4.4. Details of the Implementation

The implementation of the cost and resource optimized cloud resourcing has two parts, the first part deals with the optimized cost/pricing and what are the parameters to appropriately predict the cost based on the parameter's scalability and usage in future. The second part contains a series of models run to predict the parameters which can help in resource provisioning in the future based on the current workload behavior found based on the trace's dataset.

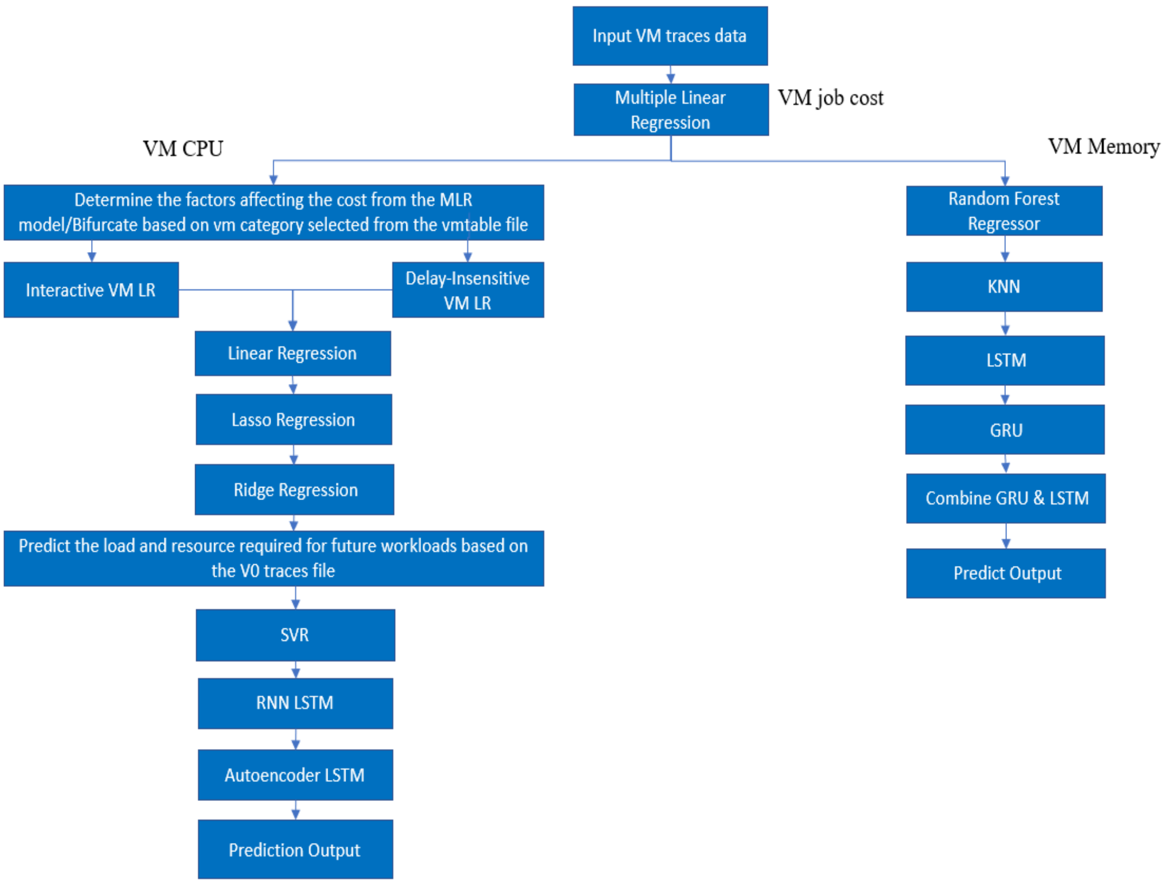


Figure 4.4 1: High level flow of the implementation methodology

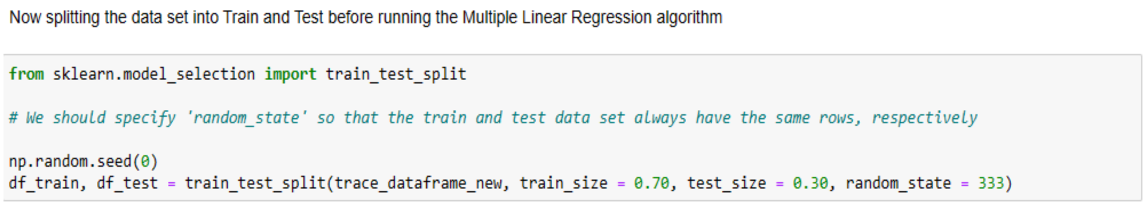
The high-level flow has been depicted as per the figure above. As part of the flow, we would first determine the cost optimization model by considering the cost from MS Pricing for various VM CPU cores and Memory utilized and then calculate the total pay as you go price, one year price,three-year price and spot prices. Thereafter we conclude on the factors which affect the cost, relations to the cost. Furthermore, we consider the traces file containing vm workloads and predict the outcomes based on the factors which were seen to be affecting the price. These factors would also help in resource provisioning in future as they would indicate the resource types which we would require in future to spin up vms and how frequently those would be required.

4.4.1. Multiple Linear Regression Model for cost predictions

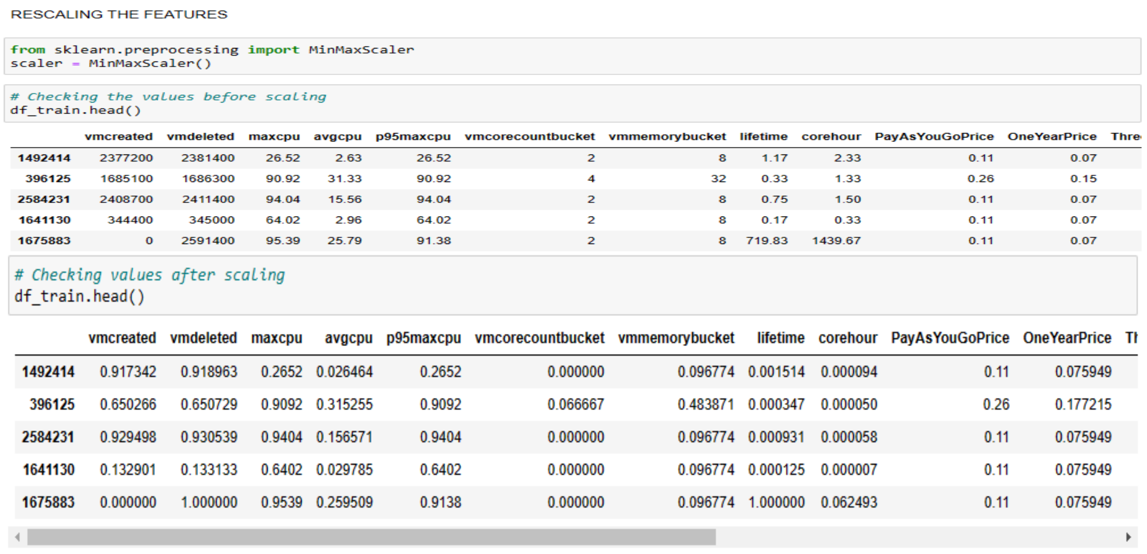
As mentioned above, I am using the multiple linear regression model to find out the factors affecting the cost, while I have taken the TotalPayAsYouGo (derived parameter) price as the variable which would need to be predicted based on the predictor variables received as part of the model building exercise.

Step 1:
As part of data preparation steps, I had created python methods, I had removed data duplicates, created the derived variables for total price based on the MS pricing information and created dummy variables for the vm categories found as part of the dataset

Step 2:
Created the test and train split and applied minmaxscaler scaling to the data prior to running the multiple linear regression model. Dropped the "TotalPayAsYouGoPrice" variable from the train dataset as this was the variable to be predicted.


Now splitting the data set into Train and Test before running the Multiple Linear Regression algorithm

```
from sklearn.model_selection import train_test_split

# We should specify 'random_state' so that the train and test data set always have the same rows, respectively

np.random.seed(0)
df_train, df_test = train_test_split(trace_dataframe_new, train_size = 0.70, test_size = 0.30, random_state = 333)
```



RESCALING THE FEATURES

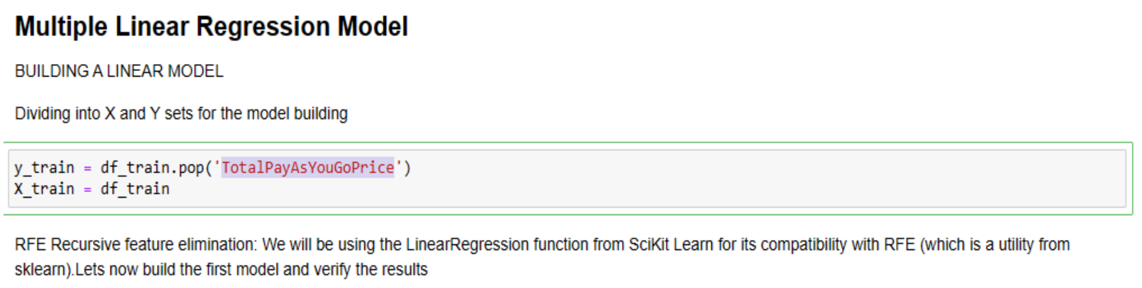
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
# Checking the values before scaling
df_train.head()
```

	vmcreated	vmdeleted	maxcpu	avgcpu	p95maxcpu	vmcorecountbucket	vmmemorybucket	lifetime	corehour	PayAsYouGoPrice	OneYearPrice	Thre
1492414	2377200	2381400	26.52	2.63	26.52	2	8	1.17	2.33	0.11	0.07	
396125	1685100	1686300	90.92	31.33	90.92	4	32	0.33	1.33	0.26	0.15	
2584231	2408700	2411400	94.04	15.56	94.04	2	8	0.75	1.50	0.11	0.07	
1641130	344400	345000	64.02	2.96	64.02	2	8	0.17	0.33	0.11	0.07	
1675883	0	2591400	95.39	25.79	91.38	2	8	719.83	1439.67	0.11	0.07	

```
# Checking values after scaling
df_train.head()
```

	vmcreated	vmdeleted	maxcpu	avgcpu	p95maxcpu	vmcorecountbucket	vmmemorybucket	lifetime	corehour	PayAsYouGoPrice	OneYearPrice	Th
1492414	0.917342	0.918963	0.2652	0.026464	0.2652	0.000000	0.096774	0.001514	0.000094	0.11	0.075949	
396125	0.650266	0.650729	0.9092	0.315255	0.9092	0.066667	0.483871	0.000347	0.000050	0.26	0.177215	
2584231	0.929498	0.930539	0.9404	0.156571	0.9404	0.000000	0.096774	0.000931	0.000058	0.11	0.075949	
1641130	0.132901	0.133133	0.6402	0.029785	0.6402	0.000000	0.096774	0.000125	0.000007	0.11	0.075949	
1675883	0.000000	1.000000	0.9539	0.259509	0.9138	0.000000	0.096774	1.000000	0.062493	0.11	0.075949	



Multiple Linear Regression Model

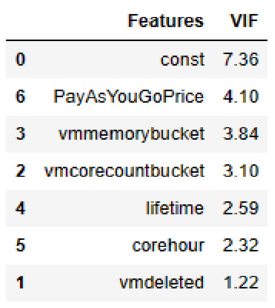
BUILDING A LINEAR MODEL

Dividing into X and Y sets for the model building

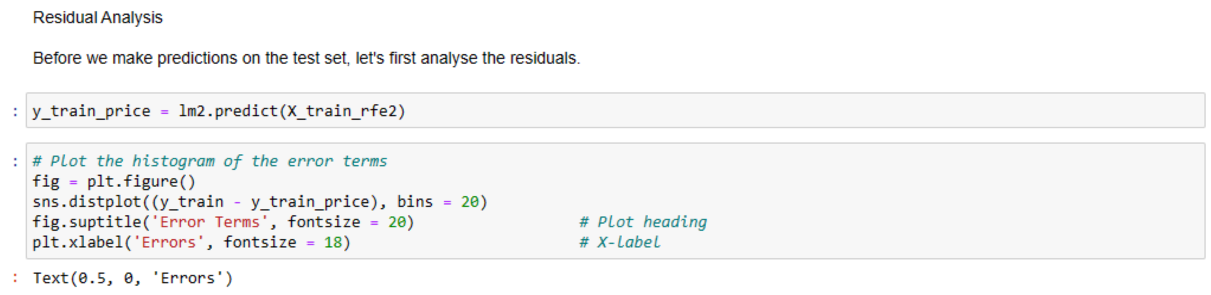
```
y_train = df_train.pop('TotalPayAsYouGoPrice')
X_train = df_train
```

RFE Recursive feature elimination: We will be using the LinearRegression function from SciKit Learn for its compatibility with RFE (which is a utility from sklearn).Lets now build the first model and verify the results

Step 3:
Ran the RFE &Model building code to finally arrive at features which are responsible for accurately predicting the Cost of the future workloads. This was achieved through multiple steps of feature elimination based on VIF values and corresponding P values for the features received.



	Features	VIF
0	const	7.36
6	PayAsYouGoPrice	4.10
3	vmmemorybucket	3.84
2	vmcorecountbucket	3.10
4	lifetime	2.59
5	corehour	2.32
1	vmdeleted	1.22



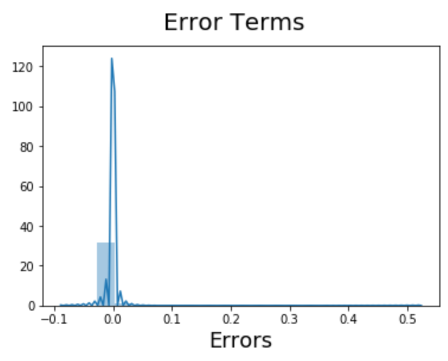
Residual Analysis

Before we make predictions on the test set, let's first analyse the residuals.

```
: y_train_price = lm2.predict(X_train_rfe2)
```

```
: # Plot the histogram of the error terms
fig = plt.figure()
sns.distplot((y_train - y_train_price), bins = 20)
fig.suptitle('Error Terms', fontsize = 20)                  # Plot heading
plt.xlabel('Errors', fontsize = 18)                         # X-label
```

```
: Text(0.5, 0, 'Errors')
```



Step 4: Making the prediction on the test set of data selected and plot a graph for the same to validate the accuracy.

Note: The multiple linear regression model above was used to predict the variables which contribute to cost hence the cost factors can be optimized accordingly. Once I had the variables, I am now running the below algorithms to further predict variables and its occurrence for future workloads based on certain criteria of timestamp etc. This would be helpful in scheduling as well as making sure that we have the appropriate resources available for future workloads based on CPU and Memory of the virtual machines.

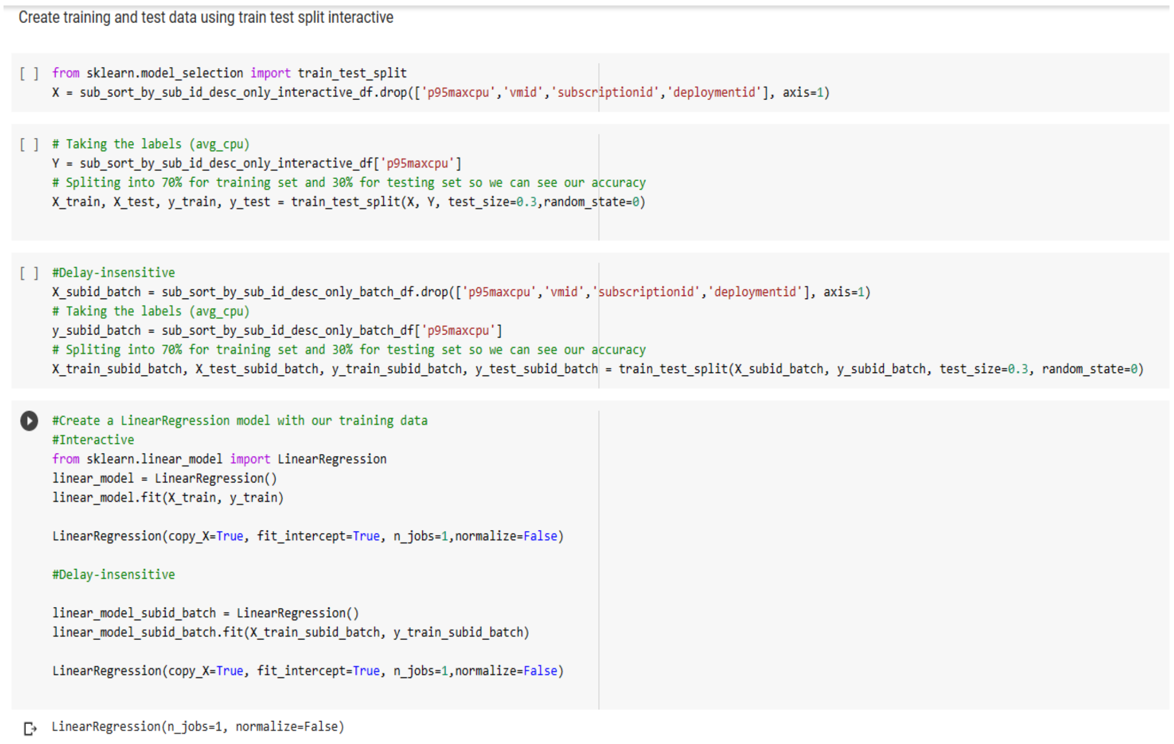
The vm table dataset is taken and is bifurcated based on the vm category. We have 3 categories as part of the dataset selected: Interactive, delay-insensitive and Unknown.

4.4.2. Predictions for virtual machine cpu and future workloads

4.4.2.1. Linear Regression Model

Step 1: Since the vm category of "Unknown" would not produce any significant insights into the vm types, I deleted the traces for that vm category.

Step 2: As mentioned above we divided the dataset into two vm categories: Interactive and delay insensitive. I have created the splits accordingly and run the linear regression model for both separately as below:



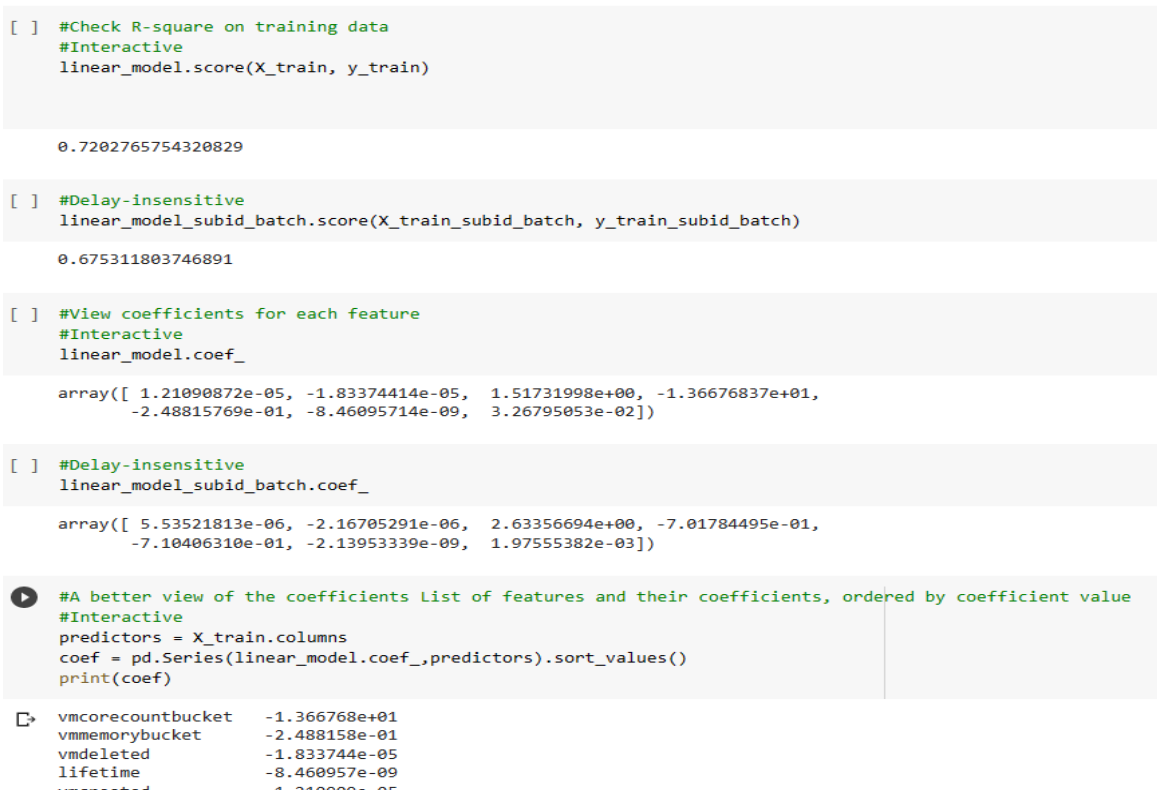
Create training and test data using train test split interactive

```
[ ] from sklearn.model_selection import train_test_split
    X = sub_sort_by_sub_id_desc_only_interactive_df.drop(['p95maxcpu','vmid','subscriptionid','deploymentid'], axis=1)
```

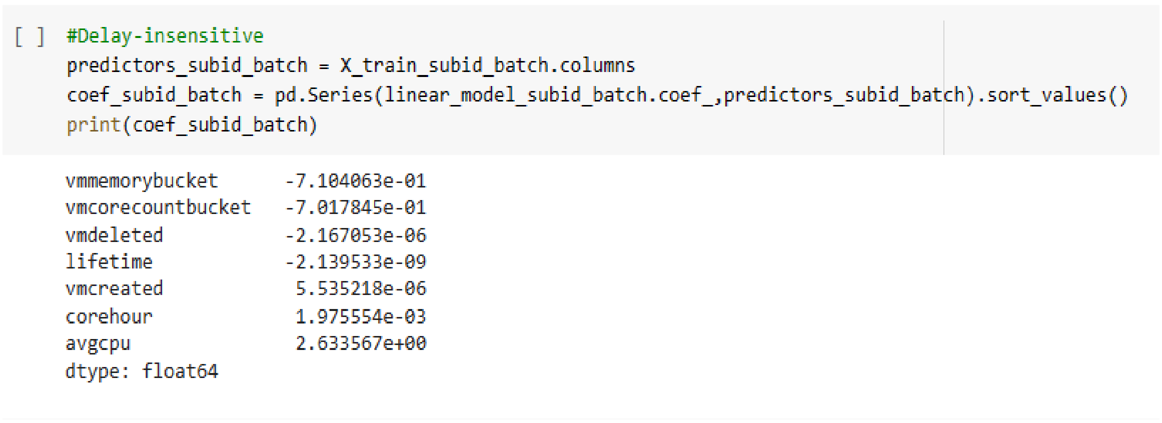
```
[ ] # Taking the labels (avg_cpu)
    Y = sub_sort_by_sub_id_desc_only_interactive_df['p95maxcpu']
    # Spliting into 70% for training set and 30% for testing set so we can see our accuracy
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,random_state=0)
```

```
[ ] #Delay-insensitive
    X_subid_batch = sub_sort_by_sub_id_desc_only_batch_df.drop(['p95maxcpu','vmid','subscriptionid','deploymentid'], axis=1)
    # Taking the labels (avg_cpu)
    y_subid_batch = sub_sort_by_sub_id_desc_only_batch_df['p95maxcpu']
    # Spliting into 70% for training set and 30% for testing set so we can see our accuracy
    X_train_subid_batch, X_test_subid_batch, y_train_subid_batch, y_test_subid_batch = train_test_split(X_subid_batch, y_subid_batch, test_size=0.3, random_state=0)
```

```
    #Create a LinearRegression model with our training data
    #Interactive
    from sklearn.linear_model import LinearRegression
    linear_model = LinearRegression()
    linear_model.fit(X_train, y_train)

    LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,normalize=False)

    #Delay-insensitive

    linear_model_subid_batch = LinearRegression()
    linear_model_subid_batch.fit(X_train_subid_batch, y_train_subid_batch)

    LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1,normalize=False)

    LinearRegression(n_jobs=1, normalize=False)
```



```
[ ] #Check R-square on training data
    #Interactive
    linear_model.score(X_train, y_train)

    0.7202765754320829

[ ] #Delay-insensitive
    linear_model_subid_batch.score(X_train_subid_batch, y_train_subid_batch)

    0.675311803746891

[ ] #View coefficients for each feature
    #Interactive
    linear_model.coef_

    array([ 1.21090872e-05, -1.83374414e-05,  1.51731998e+00, -1.36676837e+01,
           -2.48815769e-01, -8.46095714e-09,  3.26795053e-02])

[ ] #Delay-insensitive
    linear_model_subid_batch.coef_

    array([ 5.53521813e-06, -2.16705291e-06,  2.63356694e+00, -7.01784495e-01,
           -7.10406310e-01, -2.13953339e-09,  1.97555382e-03])

    #A better view of the coefficients List of features and their coefficients, ordered by coefficient value
    #Interactive
    predictors = X_train.columns
    coef = pd.Series(linear_model.coef_,predictors).sort_values()
    print(coef)

    vmcorecountbucket   -1.366768e+01
    vmmemorybucket      -2.488158e-01
    vmdeleted           -1.833744e-05
    lifetime            -8.460957e-09
```



```
[ ] #Delay-insensitive
    predictors_subid_batch = X_train_subid_batch.columns
    coef_subid_batch = pd.Series(linear_model_subid_batch.coef_,predictors_subid_batch).sort_values()
    print(coef_subid_batch)

    vmmemorybucket      -7.104063e-01
    vmcorecountbucket   -7.017845e-01
    vmdeleted           -2.167053e-06
    lifetime            -2.139533e-09
    vmcreated            5.535218e-06
    corehour             1.975554e-03
    avgcpu               2.633567e+00
    dtype: float64
```

Step 3:
Make predictions on the data and calculate metrics on how well the model performs

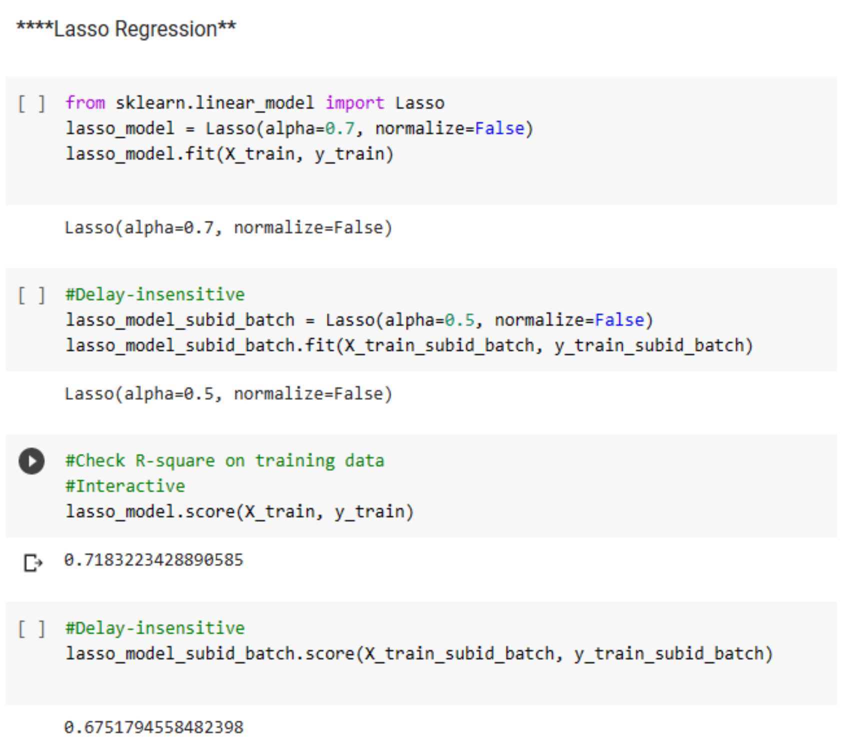
4.4.2.2. Lasso Regression Model

The lasso regression model creates the cost function as per the below:
RSS + α*(sum of absolute values of coefficients)

RSS = Residual Sum of Squares Larger values of α should result in smaller coefficients as the cost function needs to be minimized

The same bifurcated data as per vm category is used to now run the lasso regression model

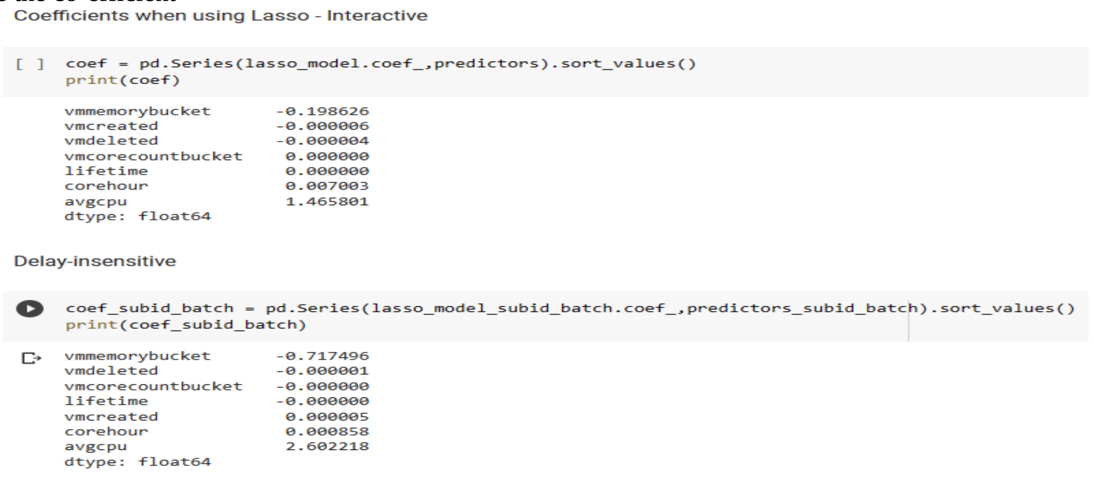
Step 1:



****Lasso Regression**

```
[ ] from sklearn.linear_model import Lasso
    lasso_model = Lasso(alpha=0.7, normalize=False)
    lasso_model.fit(X_train, y_train)

    Lasso(alpha=0.7, normalize=False)

[ ] #Delay-insensitive
    lasso_model_subid_batch = Lasso(alpha=0.5, normalize=False)
    lasso_model_subid_batch.fit(X_train_subid_batch, y_train_subid_batch)

    Lasso(alpha=0.5, normalize=False)

    #Check R-square on training data
    #Interactive
    lasso_model.score(X_train, y_train)

    0.7183223428890585

[ ] #Delay-insensitive
    lasso_model_subid_batch.score(X_train_subid_batch, y_train_subid_batch)

    0.6751794558482398
```

Step 2:
Calculate the co-efficient



Coefficients when using Lasso - Interactive

```
[ ] coef = pd.Series(lasso_model.coef_,predictors).sort_values()
    print(coef)

    vmmemorybucket        -0.198626
    vmcreated             -0.000006
    vmdeleted             -0.000004
    vmcorecountbucket      0.000000
    lifetime               0.000000
    corehour               0.007003
    avgcpu                 1.465801
    dtype: float64
```

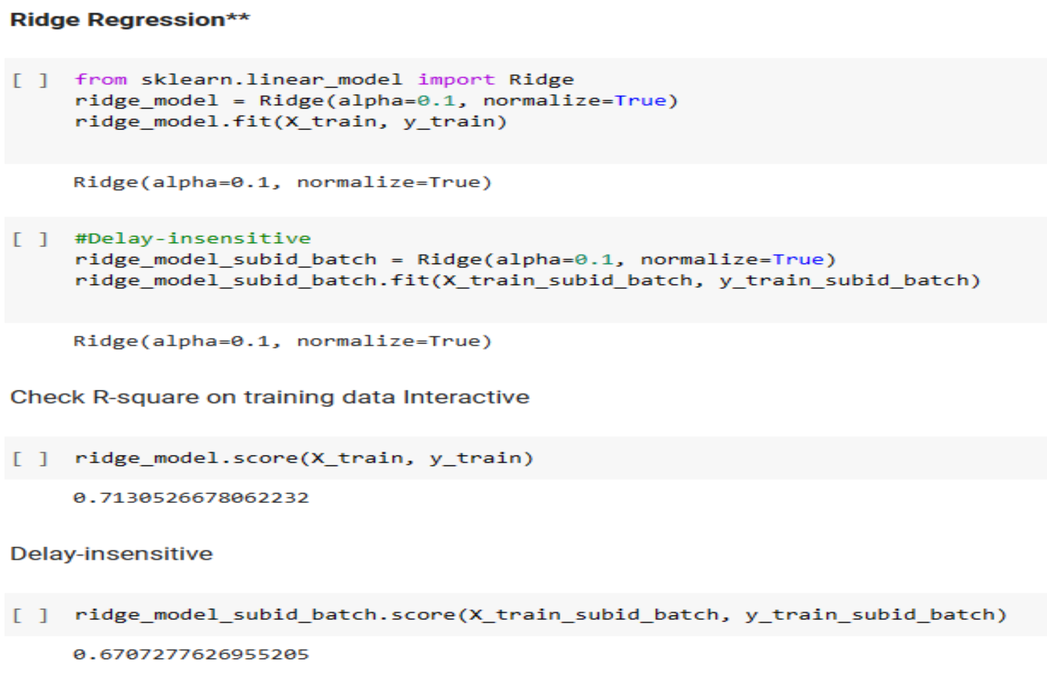
Delay-insensitive

```
    coef_subid_batch = pd.Series(lasso_model_subid_batch.coef_,predictors_subid_batch).sort_values()
    print(coef_subid_batch)

    vmmemorybucket        -0.717496
    vmdeleted             -0.000001
    vmcorecountbucket     -0.000000
    lifetime              -0.000000
    vmcreated              0.000005
    corehour               0.000858
    avgcpu                 2.602218
    dtype: float64
```

Step 3: Make predictions on the test data and calculate Metrics

4.4.2.3. Ridge Regression Model

Step 1: Run the model as below



Ridge Regression**

```
[ ]  from sklearn.linear_model import Ridge
     ridge_model = Ridge(alpha=0.1, normalize=True)
     ridge_model.fit(X_train, y_train)

     Ridge(alpha=0.1, normalize=True)
```

```
[ ]  #Delay-insensitive
     ridge_model_subid_batch = Ridge(alpha=0.1, normalize=True)
     ridge_model_subid_batch.fit(X_train_subid_batch, y_train_subid_batch)

     Ridge(alpha=0.1, normalize=True)
```

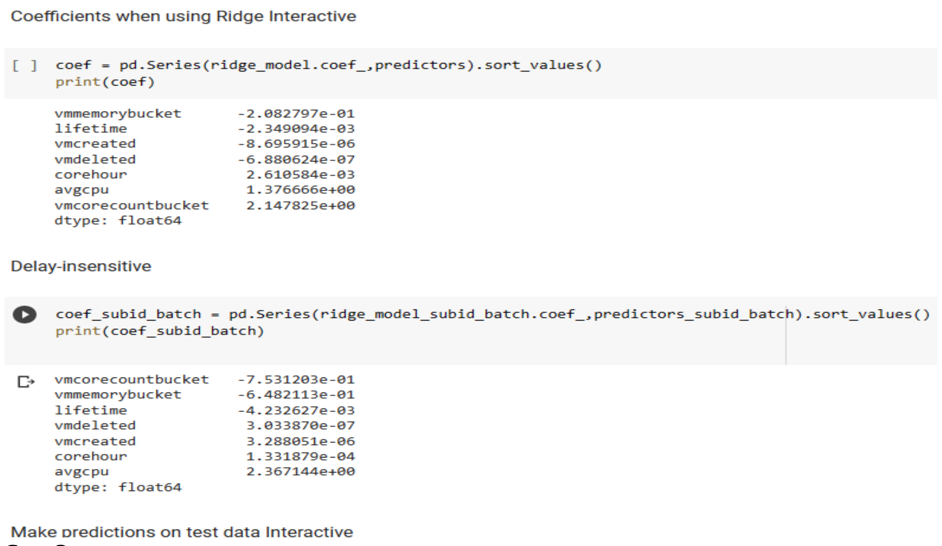
Check R-square on training data Interactive

```
[ ]  ridge_model.score(X_train, y_train)

     0.7130526678062232
```

Delay-insensitive

```
[ ]  ridge_model_subid_batch.score(X_train_subid_batch, y_train_subid_batch)

     0.6707277626955205
```

Step2:
Coefficients using Ridge



Coefficients when using Ridge Interactive

```
[ ]  coef = pd.Series(ridge_model.coef_,predictors).sort_values()
     print(coef)

     vmmemorybucket      -2.082797e-01
     lifetime            -2.349094e-03
     vmcreated           -8.695915e-06
     vmdeleted           -6.880624e-07
     corehour             2.610584e-03
     avgcpu               1.376666e+00
     vmcorecountbucket    2.147825e+00
     dtype: float64
```

Delay-insensitive

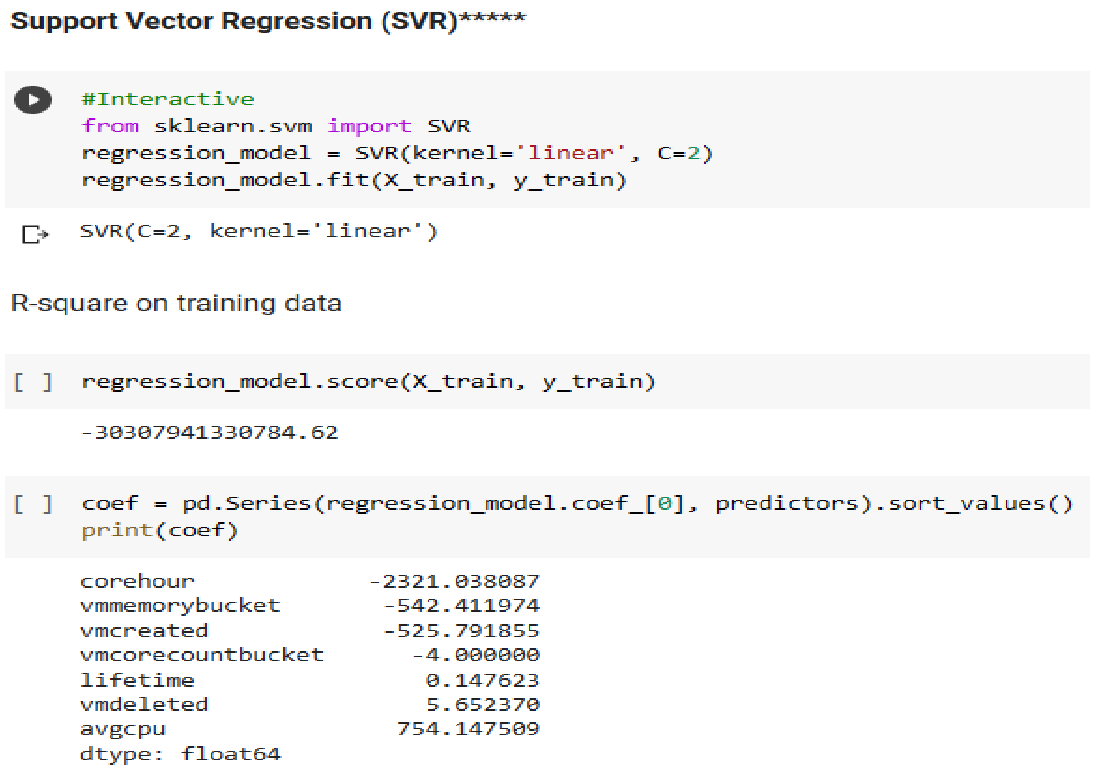
```
     coef_subid_batch = pd.Series(ridge_model_subid_batch.coef_,predictors_subid_batch).sort_values()
     print(coef_subid_batch)

     vmcorecountbucket   -7.531203e-01
     vmmemorybucket      -6.482113e-01
     lifetime            -4.232627e-03
     vmdeleted            3.033870e-07
     vmcreated            3.288051e-06
     corehour             1.331879e-04
     avgcpu               2.367144e+00
     dtype: float64
```

Make predictions on test data Interactive

Step3:
Make predictions on the test data and gather metrics on the same

4.4.2.4. Support Vector Regressor (SVR) Model

Step1:
Run the SVR model as below



Support Vector Regression (SVR)*****

```
#Interactive
from sklearn.svm import SVR
regression_model = SVR(kernel='linear', C=2)
regression_model.fit(X_train, y_train)
```

```
SVR(C=2, kernel='linear')
```

R-square on training data

```
[ ]  regression_model.score(X_train, y_train)
```

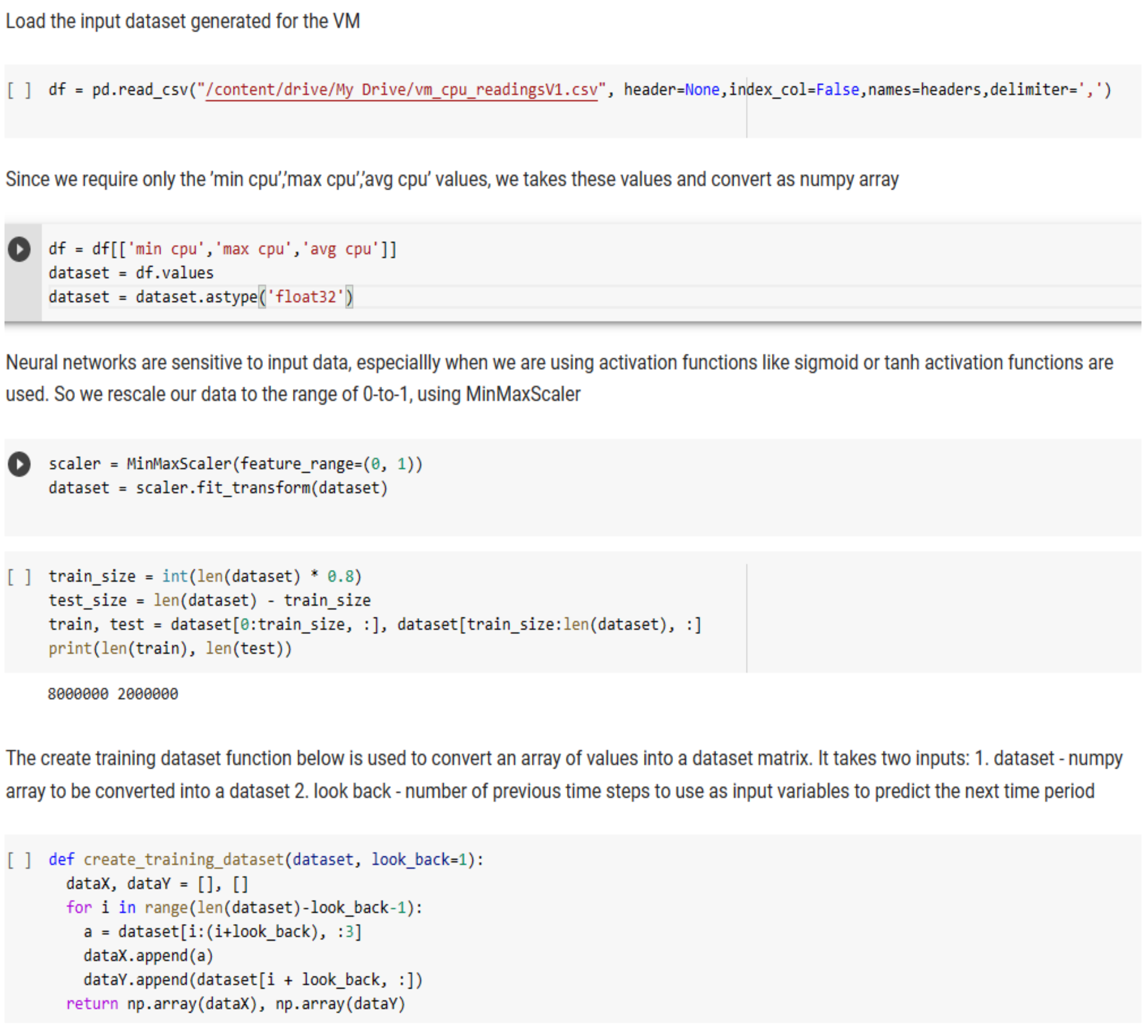
```
-303079413307 84.62
```

```
[ ]  coef = pd.Series(regression_model.coef_[0], predictors).sort_values()
     print(coef)
```

```
corehour              -2321.038087
vmmemorybucket         -542.411974
vmcreated              -525.791855
vmcorecountbucket        -4.000000
lifetime                  0.147623
vmdeleted                 5.652370
avgcpu                  754.147509
dtype: float64
```

Step2:
Make the predictions and calculate the metrics

4.4.2.5. RNN LSTM Model
This model is run by considering a vm cpu readings file from the vm traces and corresponding prediction is made for the cpu usage
Step1:



Load the input dataset generated for the VM

```
[ ] df = pd.read_csv("/content/drive/My Drive/vm_cpu_readingsV1.csv", header=None,index_col=False,names=headers,delimiter=',')
```

Since we require only the 'min cpu','max cpu','avg cpu' values, we takes these values and convert as numpy array

```
df = df[['min cpu','max cpu','avg cpu']]
dataset = df.values
dataset = dataset.astype('float32')
```

Neural networks are sensitive to input data, especiallly when we are using activation functions like sigmoid or tanh activation functions are used. So we rescale our data to the range of 0-to-1, using MinMaxScaler

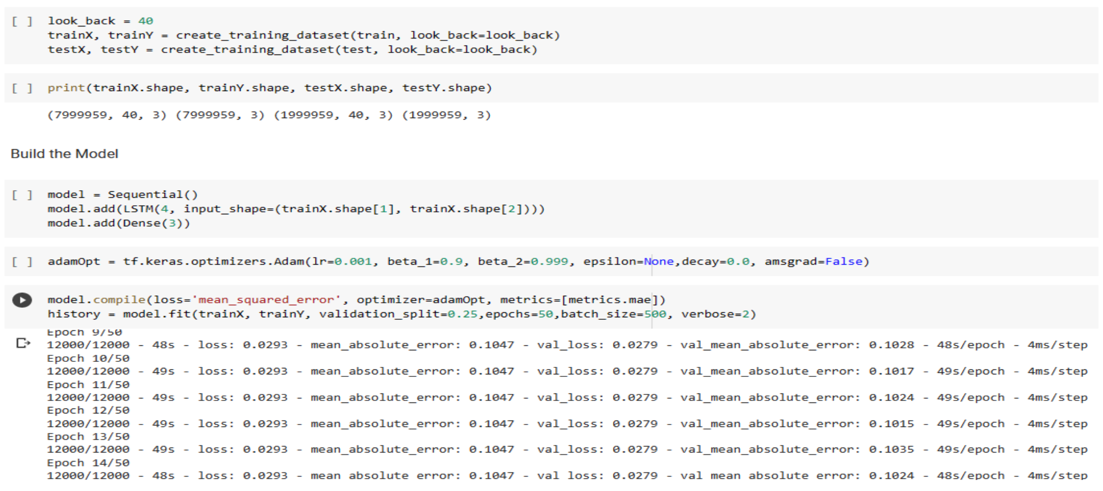
```
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
```

```
[ ] train_size = int(len(dataset) * 0.8)
    test_size = len(dataset) - train_size
    train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
    print(len(train), len(test))

    8000000 2000000
```

The create training dataset function below is used to convert an array of values into a dataset matrix. It takes two inputs: 1. dataset - numpy array to be converted into a dataset 2. look back - number of previous time steps to use as input variables to predict the next time period

```
[ ] def create_training_dataset(dataset, look_back=1):
      dataX, dataY = [], []
      for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), :3]
        dataX.append(a)
        dataY.append(dataset[i + look_back, :])
      return np.array(dataX), np.array(dataY)
```

Step2: Build the model



```
[ ] look_back = 40
    trainX, trainY = create_training_dataset(train, look_back=look_back)
    testX, testY = create_training_dataset(test, look_back=look_back)
```

```
[ ] print(trainX.shape, trainY.shape, testX.shape, testY.shape)

    (7999959, 40, 3) (7999959, 3) (1999959, 40, 3) (1999959, 3)
```

Build the Model

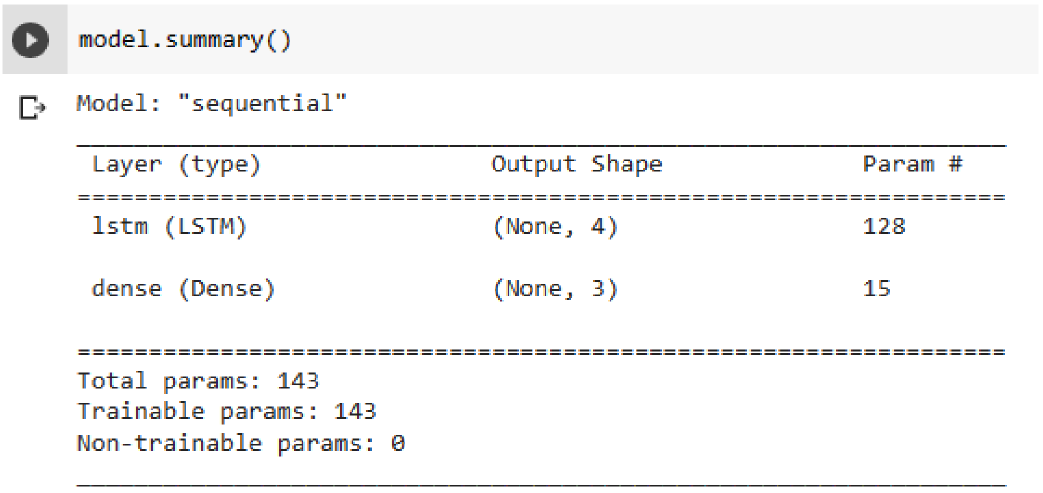
```
[ ] model = Sequential()
    model.add(LSTM(4, input_shape=(trainX.shape[1], trainX.shape[2])))
    model.add(Dense(3))
```

```
[ ] adamOpt = tf.keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,decay=0.0, amsgrad=False)
```

```
model.compile(loss='mean_squared_error', optimizer=adamOpt, metrics=[metrics.mae])
history = model.fit(trainX, trainY, validation_split=0.25,epochs=50,batch_size=500, verbose=2)
```

```
Epoch 9/50
12000/12000 - 48s - loss: 0.0293 - mean_absolute_error: 0.1047 - val_loss: 0.0279 - val_mean_absolute_error: 0.1028 - 48s/epoch - 4ms/step
Epoch 10/50
12000/12000 - 49s - loss: 0.0293 - mean_absolute_error: 0.1047 - val_loss: 0.0279 - val_mean_absolute_error: 0.1017 - 49s/epoch - 4ms/step
Epoch 11/50
12000/12000 - 49s - loss: 0.0293 - mean_absolute_error: 0.1047 - val_loss: 0.0279 - val_mean_absolute_error: 0.1024 - 49s/epoch - 4ms/step
Epoch 12/50
12000/12000 - 49s - loss: 0.0293 - mean_absolute_error: 0.1047 - val_loss: 0.0279 - val_mean_absolute_error: 0.1015 - 49s/epoch - 4ms/step
Epoch 13/50
12000/12000 - 49s - loss: 0.0293 - mean_absolute_error: 0.1047 - val_loss: 0.0279 - val_mean_absolute_error: 0.1035 - 49s/epoch - 4ms/step
Epoch 14/50
12000/12000 - 48s - loss: 0.0293 - mean absolute error: 0.1047 - val loss: 0.0279 - val mean absolute error: 0.1024 - 48s/epoch - 4ms/step
```

Step 3:
Model summary



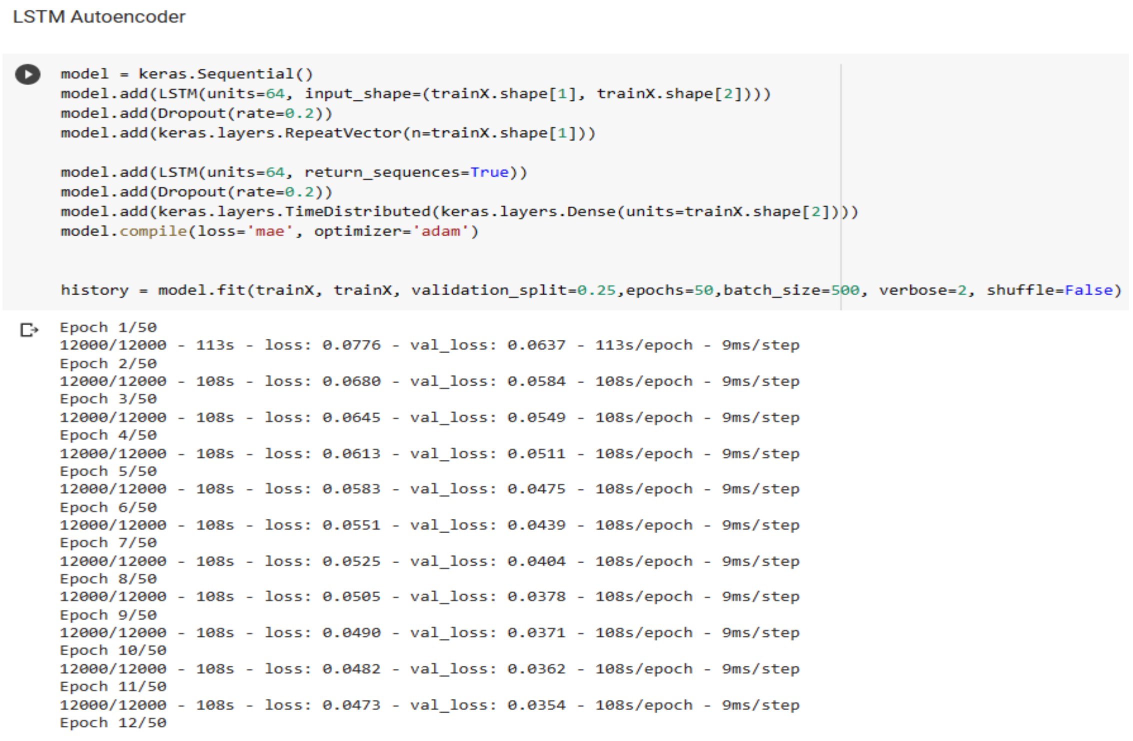
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 4)                 128

 dense (Dense)               (None, 3)                 15

=================================================================
Total params: 143
Trainable params: 143
Non-trainable params: 0
_________________________________________________________________
```

Step 4:
Make the predictions and calculate the metrics

4.4.2.6. Autoencoder LSTM Model

Step 1:

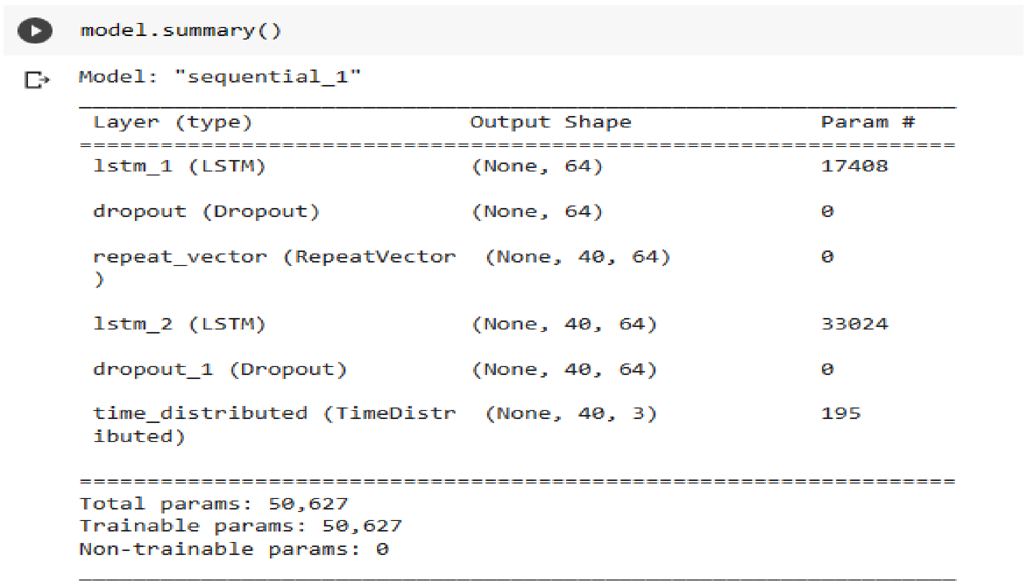


LSTM Autoencoder

```
model = keras.Sequential()
model.add(LSTM(units=64, input_shape=(trainX.shape[1], trainX.shape[2])))
model.add(Dropout(rate=0.2))
model.add(keras.layers.RepeatVector(n=trainX.shape[1]))

model.add(LSTM(units=64, return_sequences=True))
model.add(Dropout(rate=0.2))
model.add(keras.layers.TimeDistributed(keras.layers.Dense(units=trainX.shape[2])))
model.compile(loss='mae', optimizer='adam')


history = model.fit(trainX, trainX, validation_split=0.25,epochs=50,batch_size=500, verbose=2, shuffle=False)
```

```
Epoch 1/50
12000/12000 - 113s - loss: 0.0776 - val_loss: 0.0637 - 113s/epoch - 9ms/step
Epoch 2/50
12000/12000 - 108s - loss: 0.0680 - val_loss: 0.0584 - 108s/epoch - 9ms/step
Epoch 3/50
12000/12000 - 108s - loss: 0.0645 - val_loss: 0.0549 - 108s/epoch - 9ms/step
Epoch 4/50
12000/12000 - 108s - loss: 0.0613 - val_loss: 0.0511 - 108s/epoch - 9ms/step
Epoch 5/50
12000/12000 - 108s - loss: 0.0583 - val_loss: 0.0475 - 108s/epoch - 9ms/step
Epoch 6/50
12000/12000 - 108s - loss: 0.0551 - val_loss: 0.0439 - 108s/epoch - 9ms/step
Epoch 7/50
12000/12000 - 108s - loss: 0.0525 - val_loss: 0.0404 - 108s/epoch - 9ms/step
Epoch 8/50
12000/12000 - 108s - loss: 0.0505 - val_loss: 0.0378 - 108s/epoch - 9ms/step
Epoch 9/50
12000/12000 - 108s - loss: 0.0490 - val_loss: 0.0371 - 108s/epoch - 9ms/step
Epoch 10/50
12000/12000 - 108s - loss: 0.0482 - val_loss: 0.0362 - 108s/epoch - 9ms/step
Epoch 11/50
12000/12000 - 108s - loss: 0.0473 - val_loss: 0.0354 - 108s/epoch - 9ms/step
Epoch 12/50
```

Step 2:



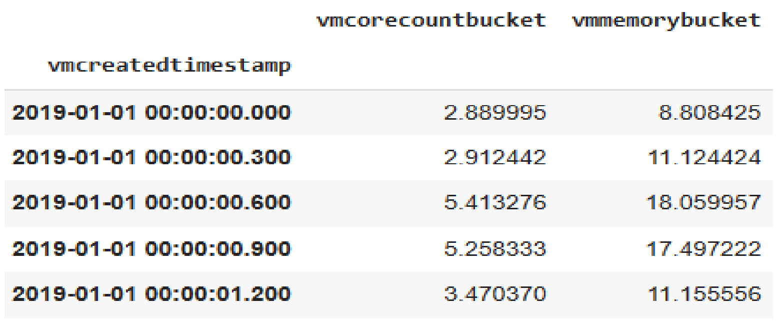
```
model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_1 (LSTM)               (None, 64)                17408

 dropout (Dropout)           (None, 64)                0

 repeat_vector (RepeatVector  (None, 40, 64)           0
 )

 lstm_2 (LSTM)               (None, 40, 64)            33024

 dropout_1 (Dropout)         (None, 40, 64)            0

 time_distributed (TimeDistr  (None, 40, 3)            195
 ibuted)

=================================================================
Total params: 50,627
Trainable params: 50,627
Non-trainable params: 0
_________________________________________________________________
```

4.4.3. Predictions for virtual machine memory and future workloads

For predictions based on the virtual machine memory I have taken the same vm traces dataset, but have bifurcated based on hours minutes and seconds and predicted the memory utilization of the vms as per the below models

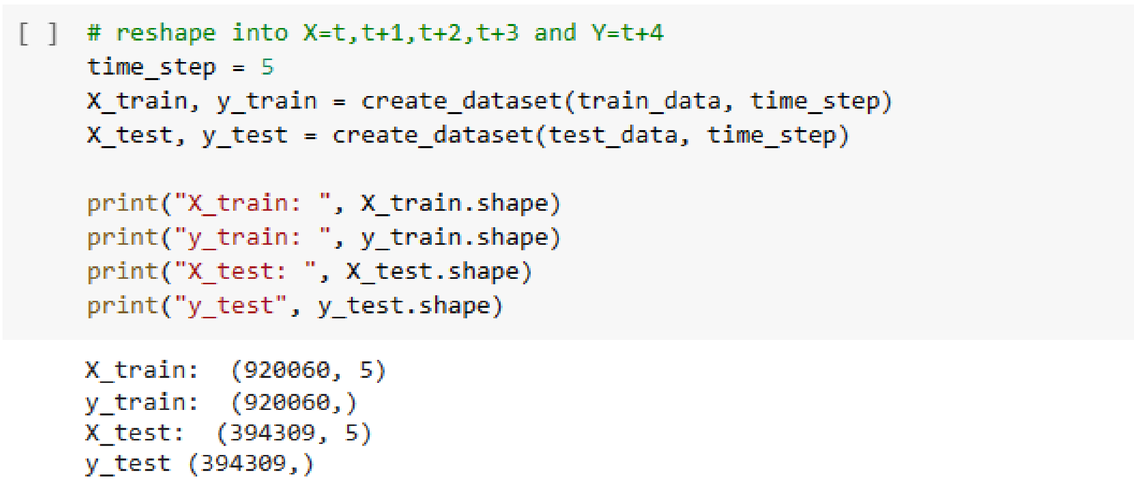
4.4.3.1. Random Forest Regressor

Step1:
Timestamp wise bifurcation of data:

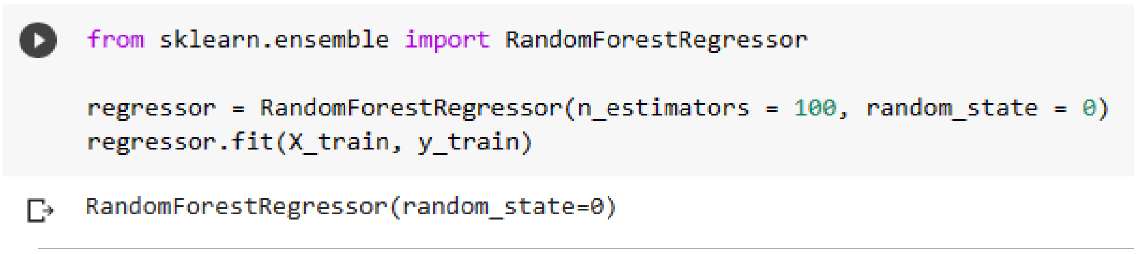


	vmcorecountbucket	vmmemorybucket
vmcreatedtimestamp		
2019-01-01 00:00:00.000	2.889995	8.808425
2019-01-01 00:00:00.300	2.912442	11.124424
2019-01-01 00:00:00.600	5.413276	18.059957
2019-01-01 00:00:00.900	5.258333	17.497222
2019-01-01 00:00:01.200	3.470370	11.155556

Step 2:
Run the random forest model by removing the timestamp so that we can predict future memory consumption



```
[ ] # reshape into X=t,t+1,t+2,t+3 and Y=t+4
    time_step = 5
    X_train, y_train = create_dataset(train_data, time_step)
    X_test, y_test = create_dataset(test_data, time_step)

    print("X_train: ", X_train.shape)
    print("y_train: ", y_train.shape)
    print("X_test: ", X_test.shape)
    print("y_test", y_test.shape)

    X_train:  (920060, 5)
    y_train:  (920060,)
    X_test:  (394309, 5)
    y_test (394309,)
```

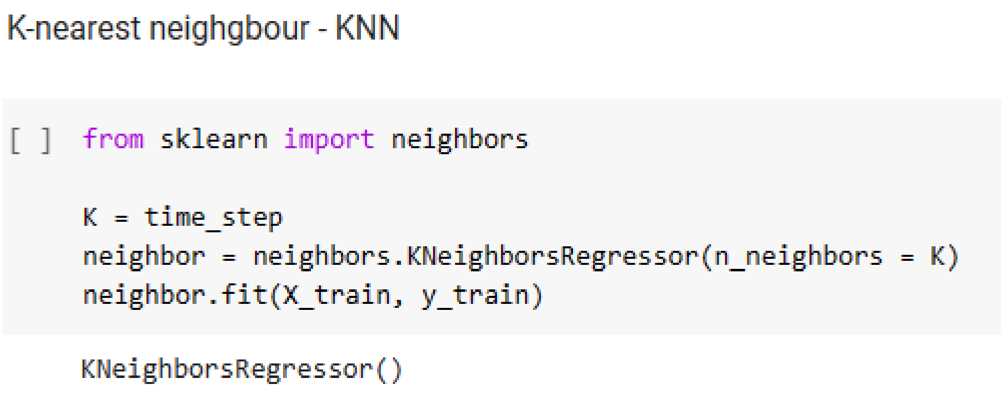


```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
regressor.fit(X_train, y_train)

RandomForestRegressor(random_state=0)
```

Step 3:
Make the prediction and publish the metrics

4.4.3.2. K-nearest neighbor – KNN

Step 1:
Run the model on the train data



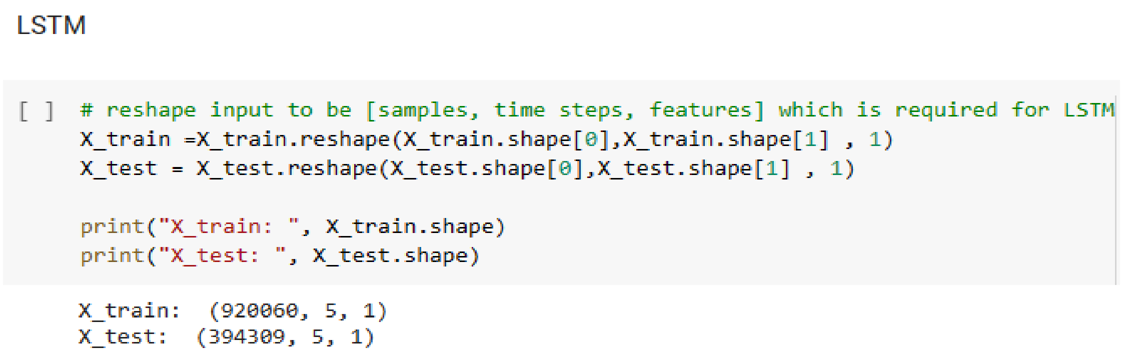
K-nearest neighgbour - KNN

```
[ ] from sklearn import neighbors

    K = time_step
    neighbor = neighbors.KNeighborsRegressor(n_neighbors = K)
    neighbor.fit(X_train, y_train)

    KNeighborsRegressor()
```

Step 2:
Make the predictions and publish the metrics

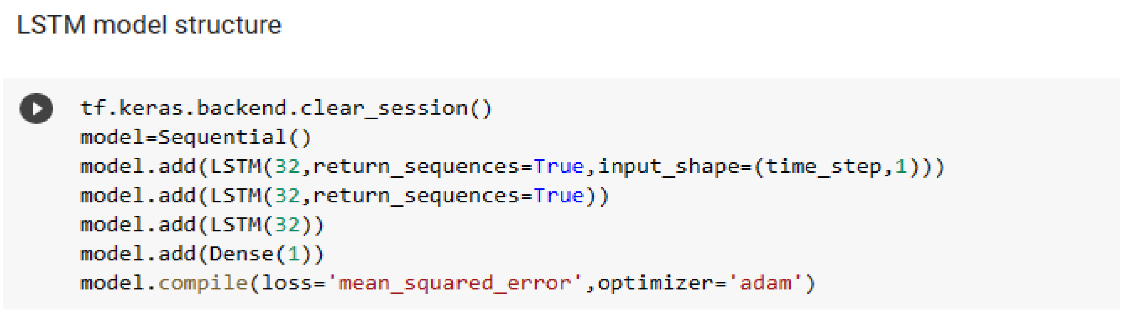
4.4.3.3. LSTM

Step 1:
Prepare the train, test data and add the parameters to the model



LSTM

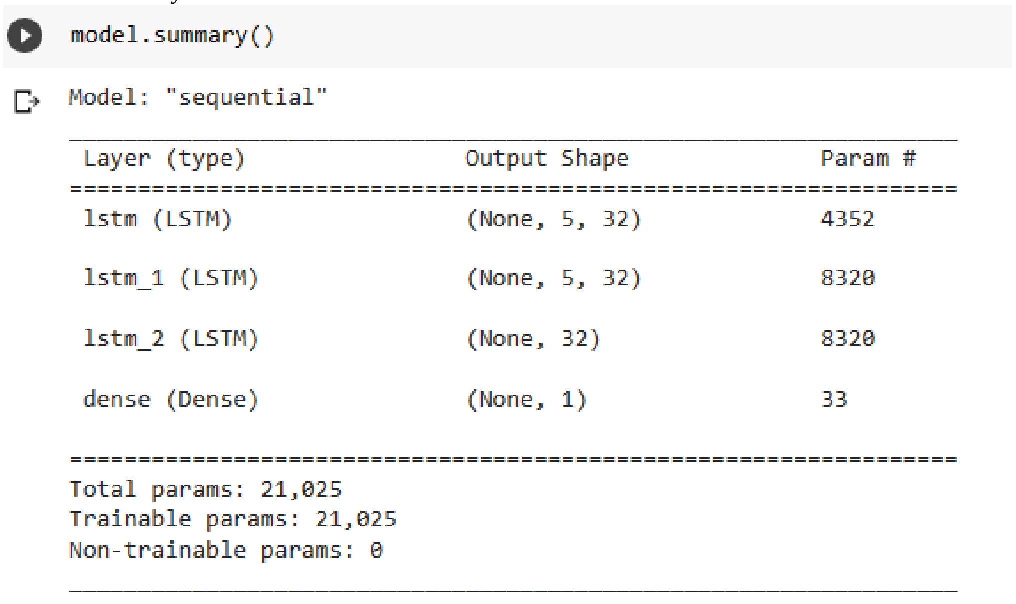
```
[ ]  # reshape input to be [samples, time steps, features] which is required for LSTM
     X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
     X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

     print("X_train: ", X_train.shape)
     print("X_test: ", X_test.shape)

     X_train:  (920060, 5, 1)
     X_test:  (394309, 5, 1)
```



LSTM model structure

```
tf.keras.backend.clear_session()
model=Sequential()
model.add(LSTM(32,return_sequences=True,input_shape=(time_step,1)))
model.add(LSTM(32,return_sequences=True))
model.add(LSTM(32))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

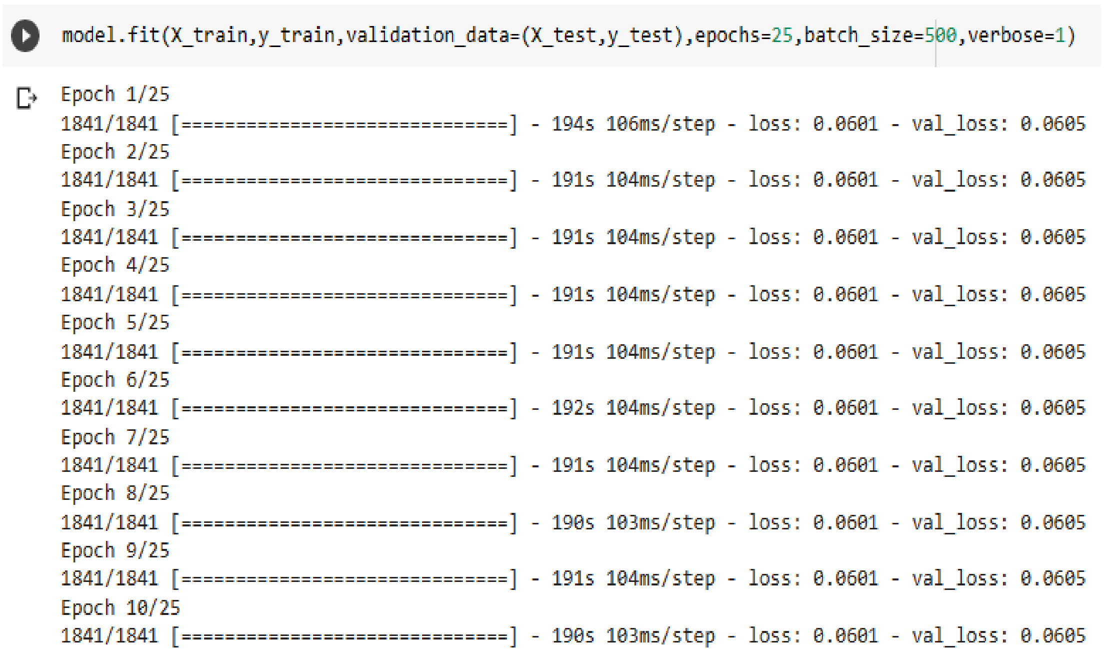
Step2:
Model Summary:



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 5, 32)             4352

 lstm_1 (LSTM)               (None, 5, 32)             8320

 lstm_2 (LSTM)               (None, 32)                8320

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 21,025
Trainable params: 21,025
Non-trainable params: 0
_________________________________________________________________
```

Step 3:
Run the fit to model



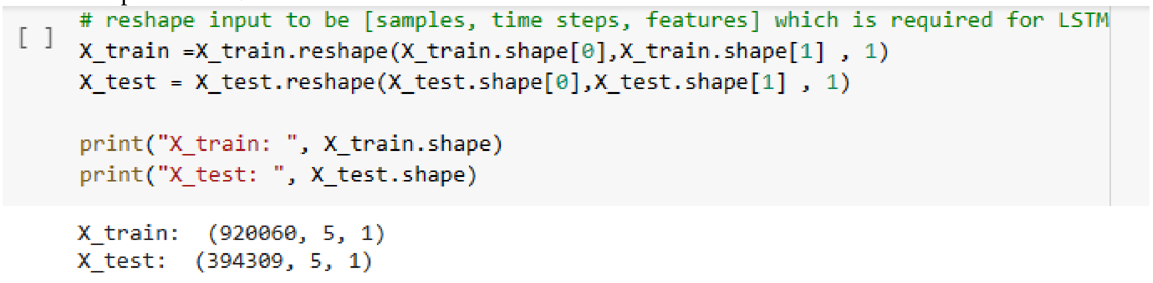
```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=25,batch_size=500,verbose=1)
```

```
Epoch 1/25
1841/1841 [==============================] - 194s 106ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 2/25
1841/1841 [==============================] - 191s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 3/25
1841/1841 [==============================] - 191s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 4/25
1841/1841 [==============================] - 191s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 5/25
1841/1841 [==============================] - 191s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 6/25
1841/1841 [==============================] - 192s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 7/25
1841/1841 [==============================] - 191s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 8/25
1841/1841 [==============================] - 190s 103ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 9/25
1841/1841 [==============================] - 191s 104ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 10/25
1841/1841 [==============================] - 190s 103ms/step - loss: 0.0601 - val_loss: 0.0605
```

Step 4:
Make the prediction on the test data and publish the metrics

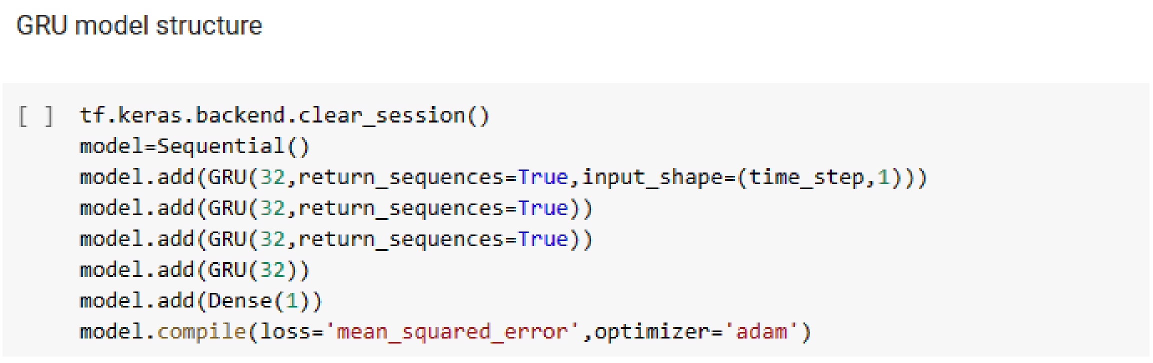
4.4.3.4. Gated Recurrent Unit (GRU)

Step1:
Define the parameters:



```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
```

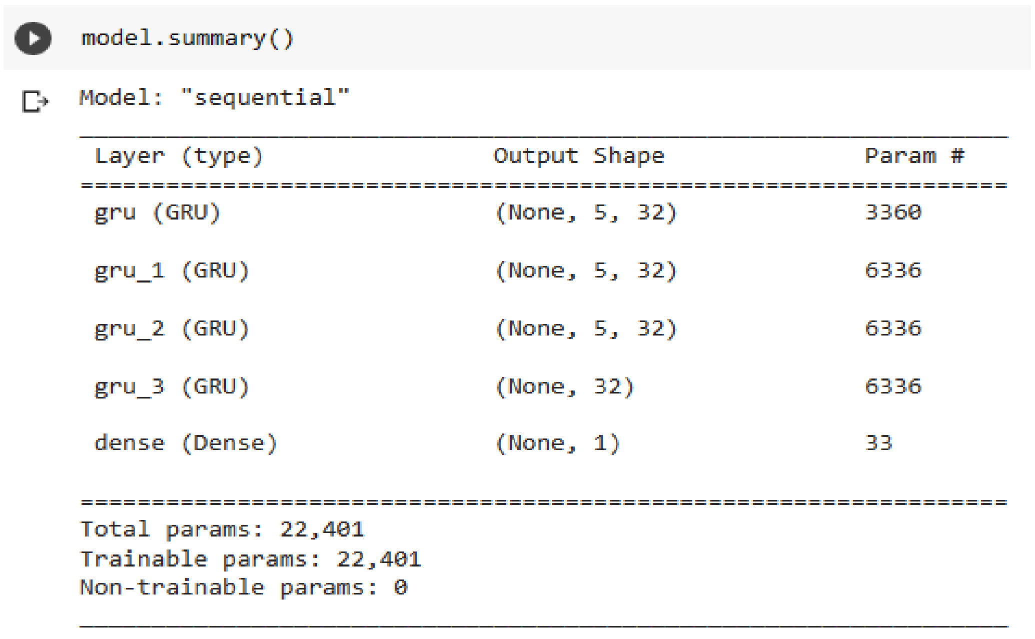
```
X_train:  (920060, 5, 1)
X_test:  (394309, 5, 1)
```



GRU model structure

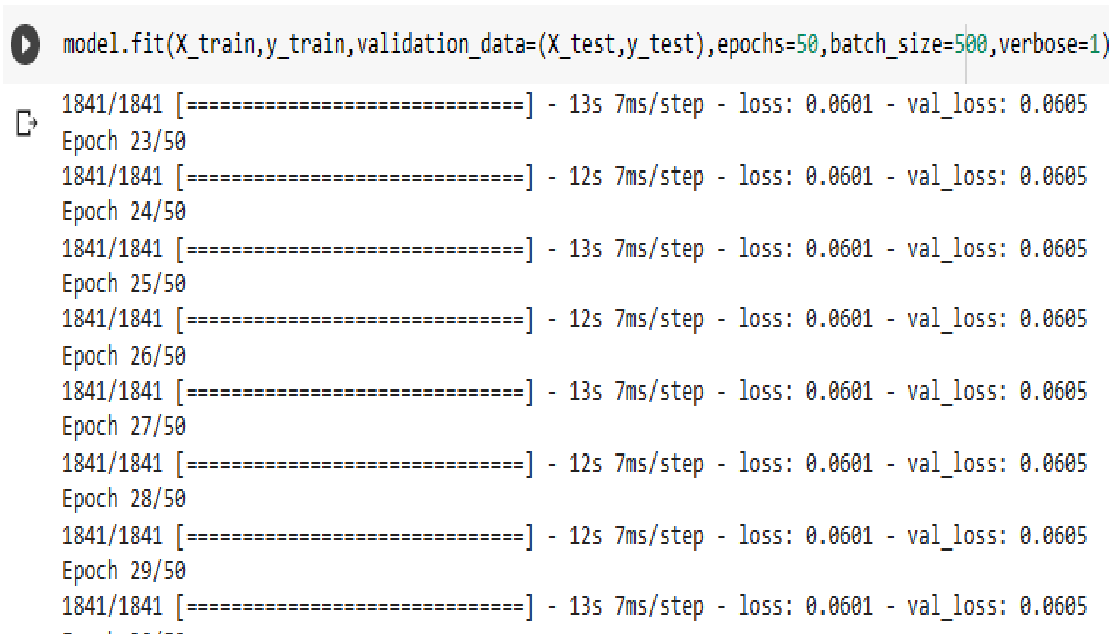
```
tf.keras.backend.clear_session()
model=Sequential()
model.add(GRU(32,return_sequences=True,input_shape=(time_step,1)))
model.add(GRU(32,return_sequences=True))
model.add(GRU(32,return_sequences=True))
model.add(GRU(32))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

Step 2:
Model summary



```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 5, 32)             3360

 gru_1 (GRU)                 (None, 5, 32)             6336

 gru_2 (GRU)                 (None, 5, 32)             6336

 gru_3 (GRU)                 (None, 32)                6336

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 22,401
Trainable params: 22,401
Non-trainable params: 0
_________________________________________________________________
```

Step 3:
Run the model fit

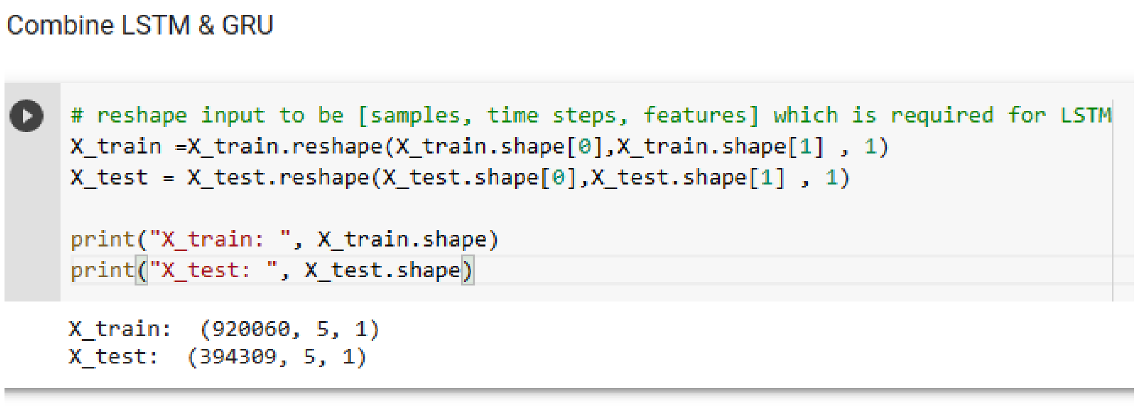


```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=50,batch_size=500,verbose=1)

1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 23/50
1841/1841 [==============================] - 12s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 24/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 25/50
1841/1841 [==============================] - 12s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 26/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 27/50
1841/1841 [==============================] - 12s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 28/50
1841/1841 [==============================] - 12s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 29/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
```

Step 4:
Make the prediction on the test data and publish the metrics

4.4.3.5. Combination of LSTM and GRU

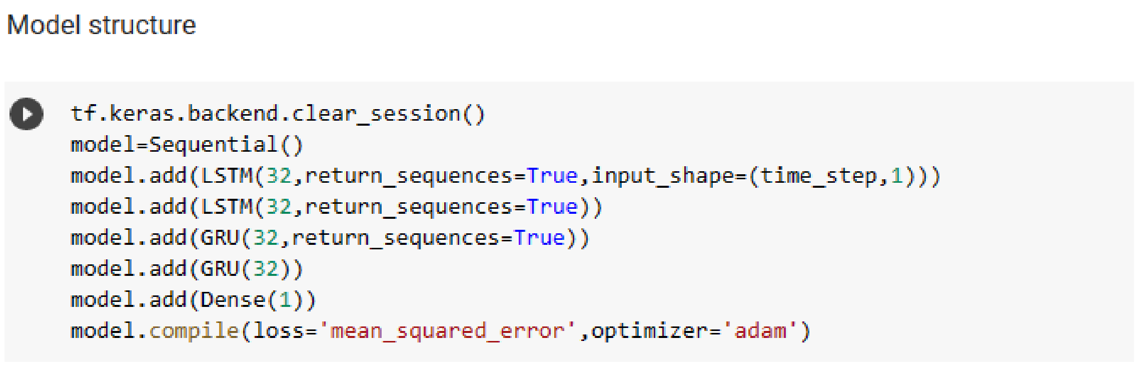
Step 1:
Run model structure and define train and test set



Combine LSTM & GRU

```
# reshape input to be [samples, time steps, features] which is required for LSTM
X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

print("X_train: ", X_train.shape)
print("X_test: ", X_test.shape)
```

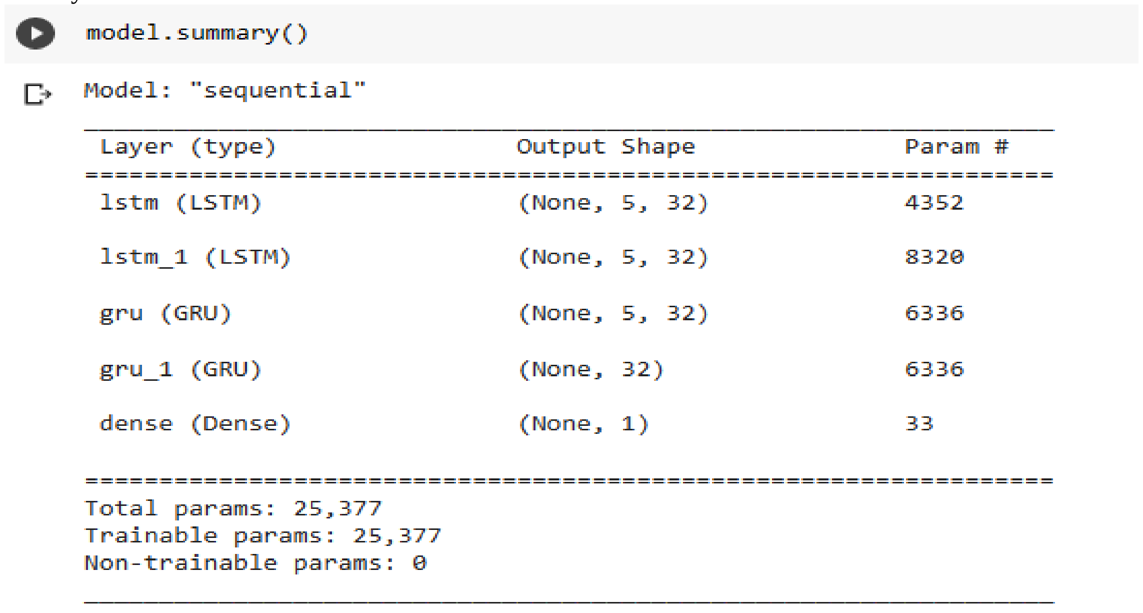
```
X_train:  (920060, 5, 1)
X_test:  (394309, 5, 1)
```



Model structure

```
tf.keras.backend.clear_session()
model=Sequential()
model.add(LSTM(32,return_sequences=True,input_shape=(time_step,1)))
model.add(LSTM(32,return_sequences=True))
model.add(GRU(32,return_sequences=True))
model.add(GRU(32))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam')
```

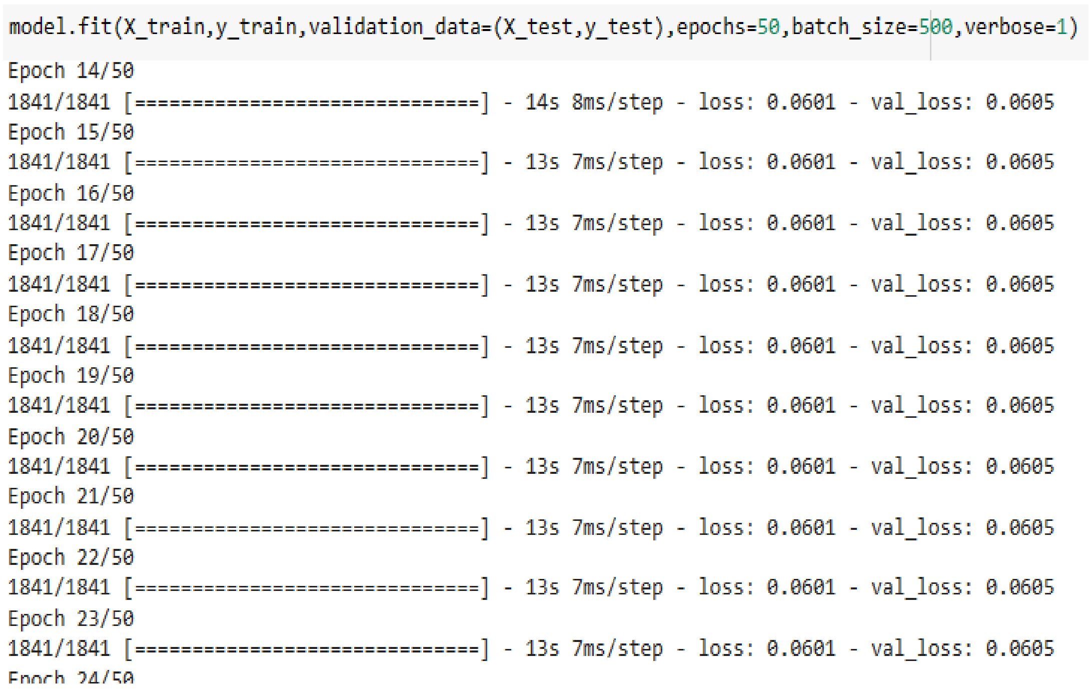
Step 2:
Model summary:



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 5, 32)             4352

 lstm_1 (LSTM)               (None, 5, 32)             8320

 gru (GRU)                   (None, 5, 32)             6336

 gru_1 (GRU)                 (None, 32)                6336

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 25,377
Trainable params: 25,377
Non-trainable params: 0
_________________________________________________________________
```

Step 3:
Run the model fit:



```
model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=50,batch_size=500,verbose=1)
Epoch 14/50
1841/1841 [==============================] - 14s 8ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 15/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 16/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 17/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 18/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 19/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 20/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 21/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 22/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 23/50
1841/1841 [==============================] - 13s 7ms/step - loss: 0.0601 - val_loss: 0.0605
Epoch 24/50
```

Step4:
Make the prediction on the test data and publish the metrics

4.5. Summary

The above section has described the detailed implementation procedure starting from the cost factors and then doing a deep dive into the parameters which are highly useful to predict the future workload behavior namely virtual machine CPU and Memory utilization based on vm types as well as based on cpu reading from the vms. The research work also has found that there is a relation between the cost and the scaling strategy that takes place into vms. The cost is found to be decrease if the scaling is done horizontally rather than vertically based on the first multiple linear regression model runs.

5. Results and Discussions

5.1. Introduction

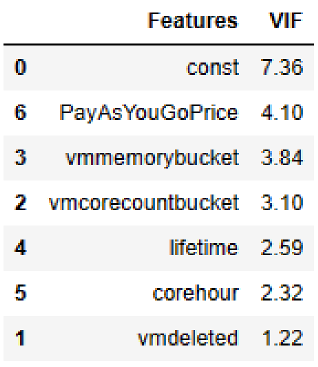
The results and discussions section describes in detail how the performance of the models run perform in terms of predicting the future workloads, resource requirements and cost optimization aspects. Just like the earlier section where I have bifurcated the models based on the virtual machine cpu and virtual machine memory requirement prediction, in this section also the same division would be seen, and the results would be grouped twofold: one for the vm cpu utilization and one for vm memory parameters utilization

5.2. Methods used and Evaluation Results

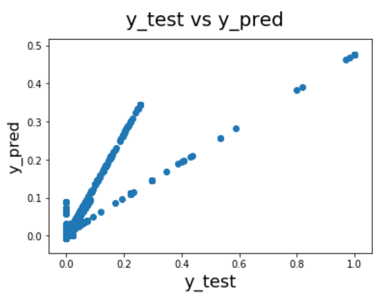
The methods and the models used have been explained in detail in Chapter 4 above. The results of the methods and Models are as below

5.2.1. Multiple Linear Regression results and conclusions

Top Cost contributing Features:



	Features	VIF
0	const	7.36
6	PayAsYouGoPrice	4.10
3	vmmemorybucket	3.84
2	vmcorecountbucket	3.10
4	lifetime	2.59
5	corehour	2.32
1	vmdeleted	1.22



From the above plot, it's evident that the model is doing well on the test set as well. Let's also check the R-squared and more importantly, the adjusted R-squared value for the test set.

Figure 5.2 1: Plot showing the results of the multiple regression

```
# r2_score for 6 variables
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
0.7860131055778703
```

Thus, for the model with 6 variables, the r-squared on training and test data is about 78.3% and 78.6% respectively. The adjusted r-squared on the train set is about is about 78.3%.

Checking the correlations between the final predictor variables

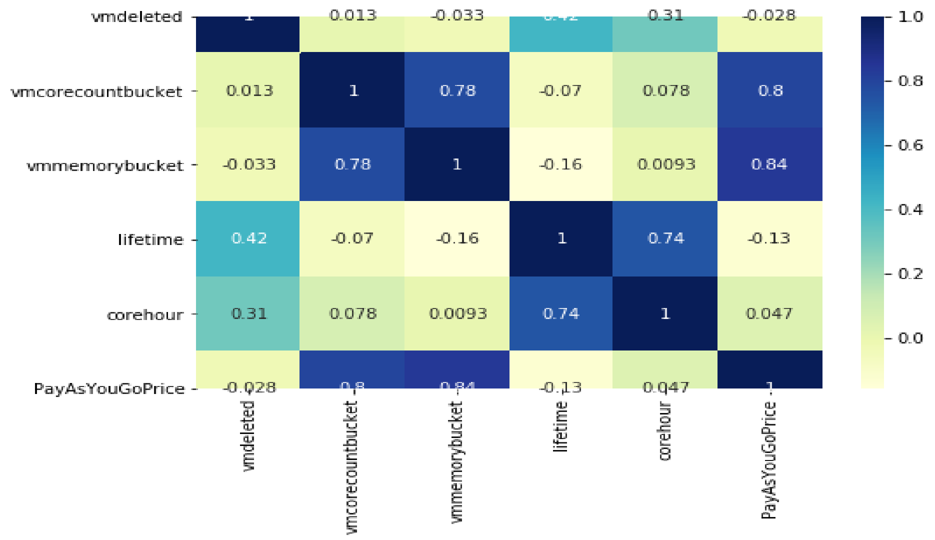


Figure 5.2 2: Plot showing the co-relations between the various resultant features

Though the model is simple, from the results it is clear

1) The Cost factor associated to vms is linearly related to vm memory bucket
2) The Cost factor associated to vms is linearly related to vm core count bucket. Hence horizontal scaling causes less cost compared to vertical scaling
3) Core count is again related to the cpu usage and hence the cost is also attached to the cpu usage and maximum cpu utilized.
4) The lifetime of vm is also corelated to the corehour

5.2.2. Results for virtual machine cpu prediction

5.2.2.1. Linear Regression Model

The comparison of the actual and predicted values are similar as per the below graph for Interactive vms

```
#Compare predicted and actual values of p95maxcpu
#Interactive
%pylab inline
pylab.rcParams['figure.figsize'] = (15, 6)
plt.plot(y_predict, label='Predicted')
plt.plot(y_test.values, label='Actual')
plt.ylabel('p95maxcpu')
plt.legend()
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

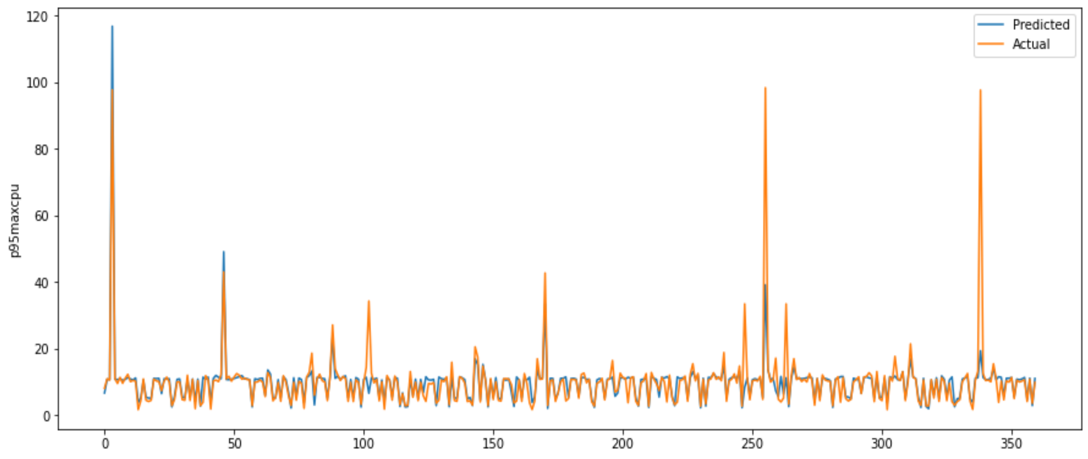


Figure 5.2.2.1: Plot showing the predicted vs actual for interactive vms

Delay-Insensitive VM:

```
#Delay-insensitive
%pylab inline
pylab.rcParams['figure.figsize'] = (15, 6)
plt.plot(y_predict_subid_batch, label='Predicted')
plt.plot(y_test_subid_batch.values, label='Actual')
plt.ylabel('p95maxcpu')
plt.legend()
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

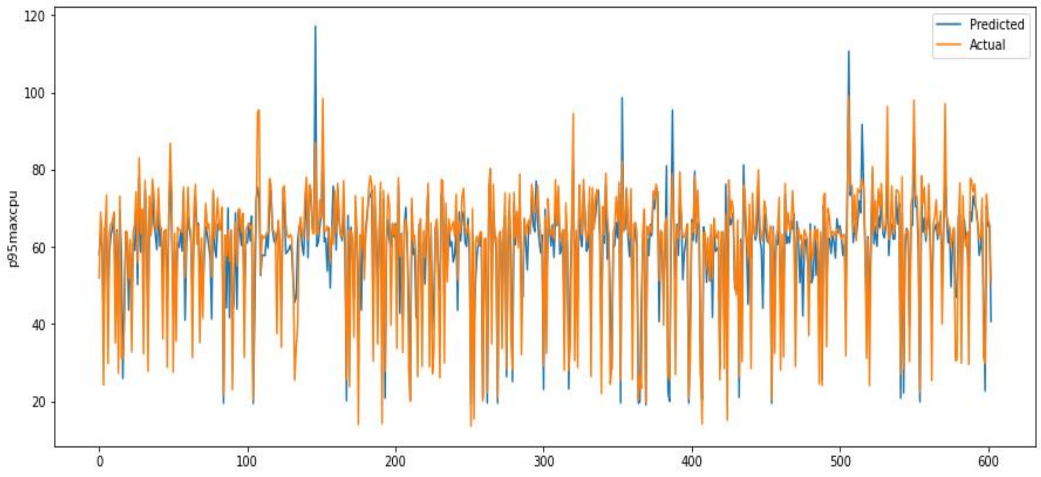


Figure 5.2.2.2: Plot showing the predicted vs actual for delay-insensitive vms

5.2.2.2. Lasso Regression Model

Compare predicted and actual values of p95maxcpu Interactive

```
%pylab inline
pylab.rcParams['figure.figsize'] = (15, 6)
plt.plot(y_predict, label='Predicted')
plt.plot(y_test.values, label='Actual')
plt.ylabel('p95maxcpu')
plt.legend()
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

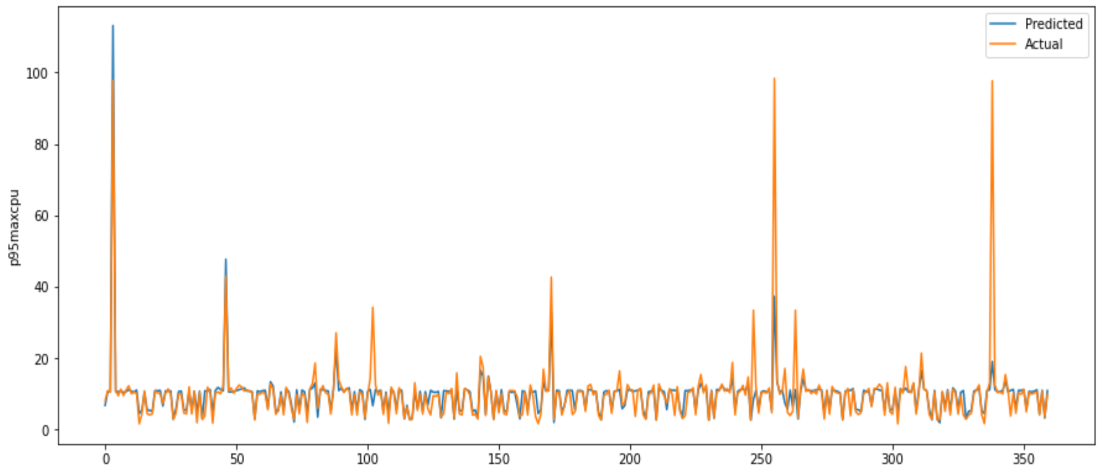
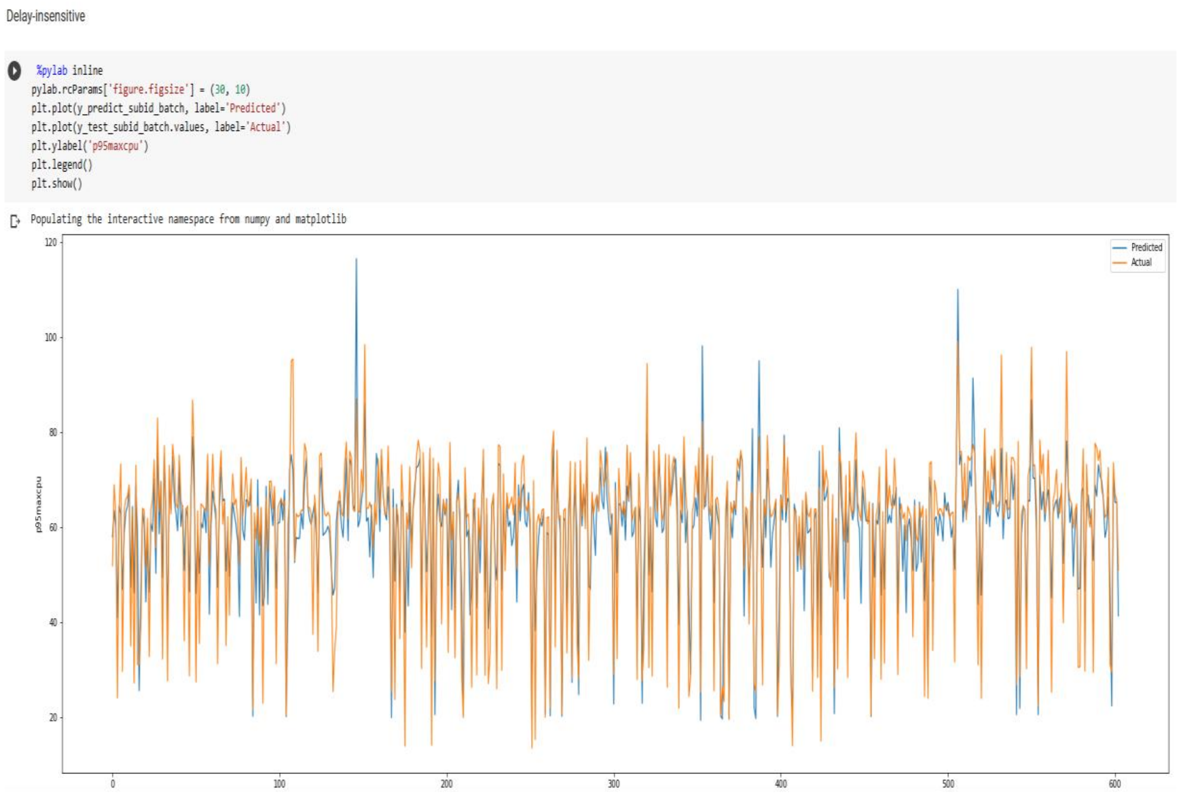


Figure 5.2.2.3: Plot showing the predicted vs actual for interactive vms for lasso



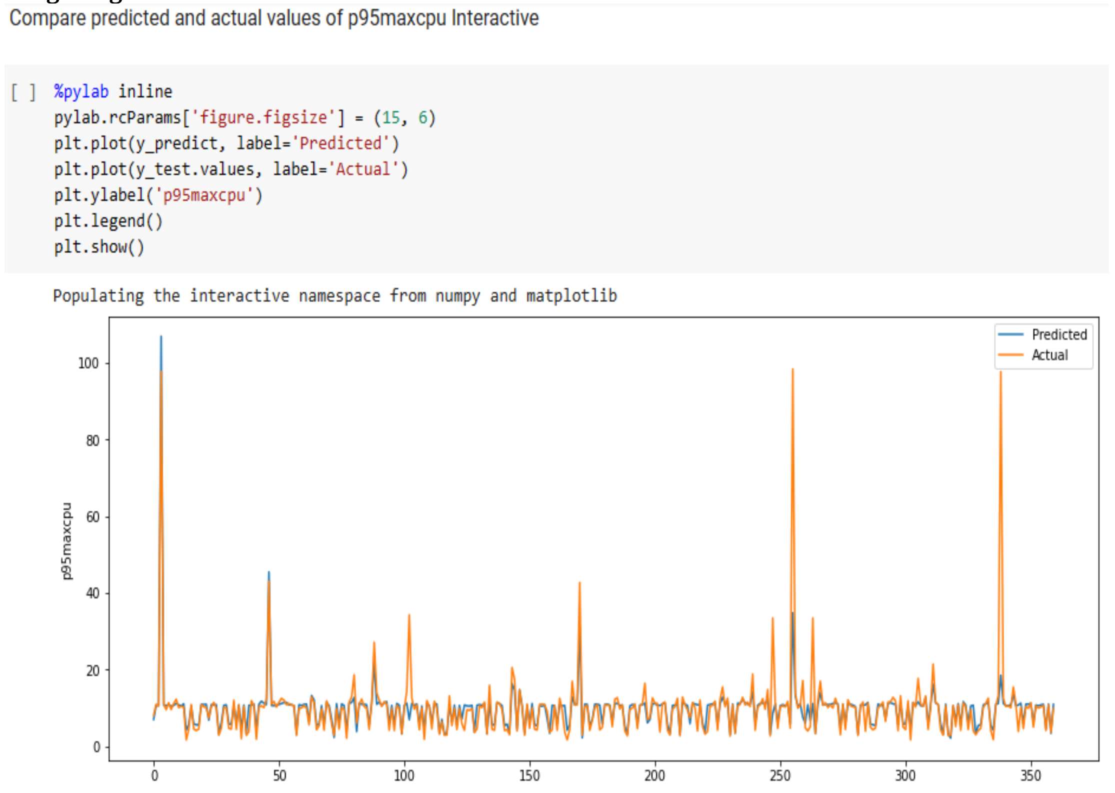
Delay-insensitive

```
%pylab inline
pylab.rcParams['figure.figsize'] = (30, 10)
plt.plot(y_predict_subid_batch, label='Predicted')
plt.plot(y_test_subid_batch.values, label='Actual')
plt.ylabel('p95maxcpu')
plt.legend()
plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

Figure 5.2.2.4: Plot showing the predicted vs actual for delay-insensitive vms for lasso

5.2.2.3. Ridge Regression Model



Compare predicted and actual values of p95maxcpu Interactive

```
[ ] %pylab inline
    pylab.rcParams['figure.figsize'] = (15, 6)
    plt.plot(y_predict, label='Predicted')
    plt.plot(y_test.values, label='Actual')
    plt.ylabel('p95maxcpu')
    plt.legend()
    plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

Figure 5.2.2.5: Plot showing the predicted vs actual for interactive vms for ridge

Delay-insensitive

```
[ ] %pylab inline
    pylab.rcParams['figure.figsize'] = (15, 6)
    plt.plot(y_predict_subid_batch, label='Predicted')
    plt.plot(y_test_subid_batch.values, label='Actual')
    plt.ylabel('p95maxcpu')
    plt.legend()
    plt.show()

    Populating the interactive namespace from numpy and matplotlib
```

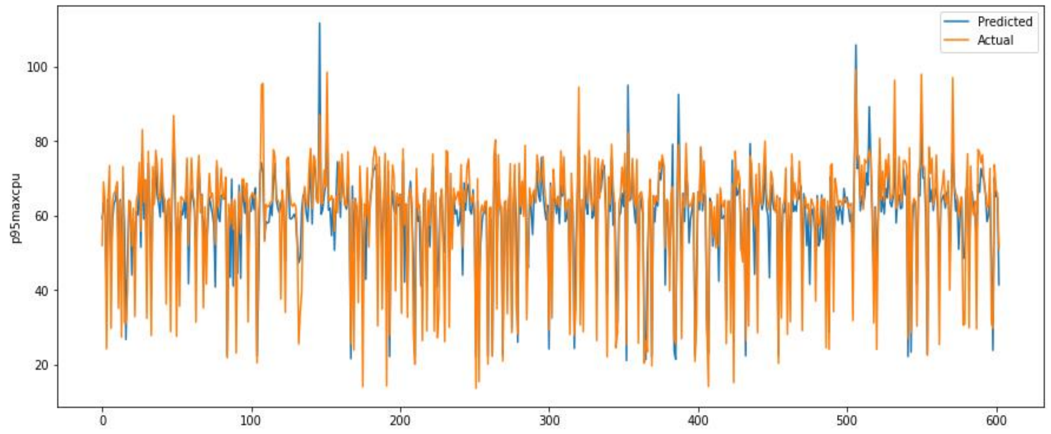


Figure 5.2.2. 6: Plot showing the predicted vs actual for delay-insensitive vms for ridge

5.2.2.4. Support Vector Regressor (SVR) Model

```
%pylab inline
pylab.rcParams['figure.figsize'] = (15, 6)
plt.plot(y_predict, label='Predicted')
plt.plot(y_test.values, label='Actual')
plt.ylabel('p95maxcpu')
plt.legend()
plt.show()

Populating the interactive namespace from numpy and matplotlib
```

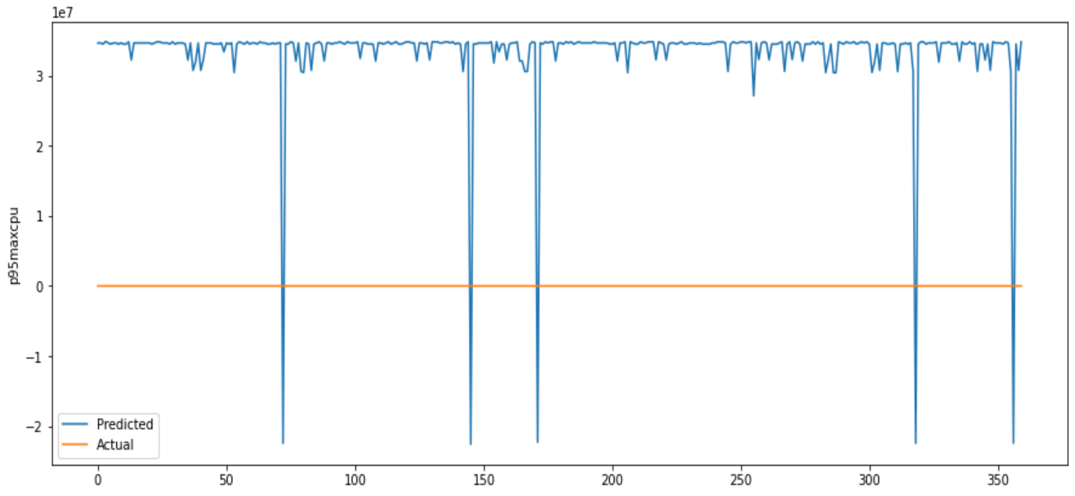
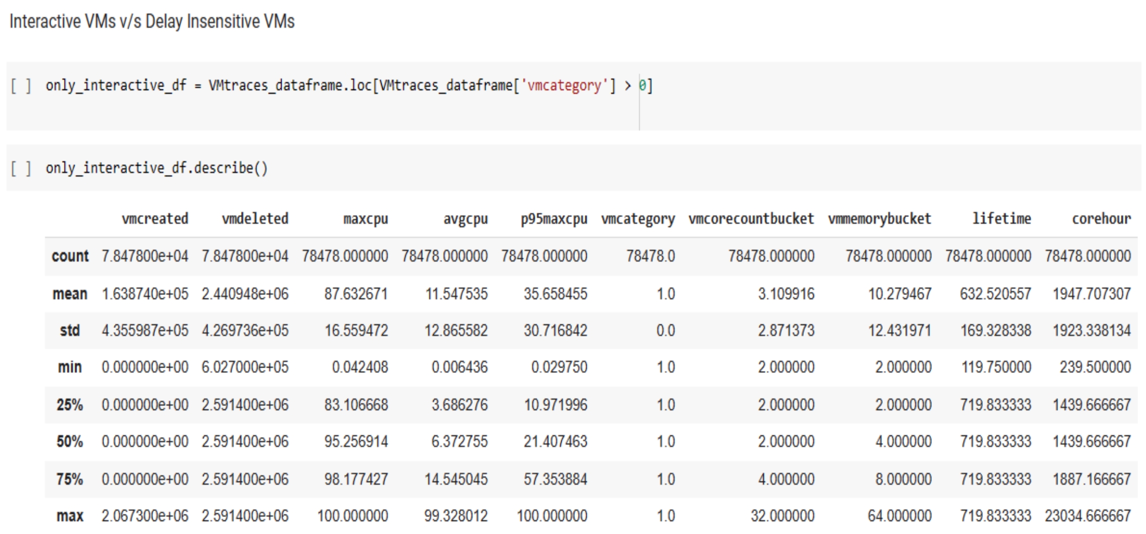


Figure 5.2.2.7: Plot showing the predicted vs actual for interactive vms for SVR


Interactive VMs v/s Delay Insensitive VMs

```
[ ] only_interactive_df = VMtraces_dataframe.loc[VMtraces_dataframe['vmcategory'] > 0]
```

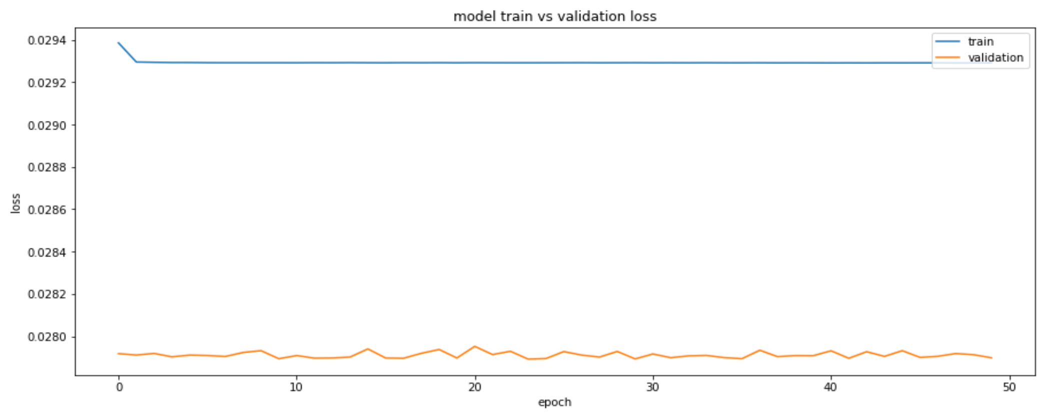
```
[ ] only_interactive_df.describe()
```

	vmcreated	vmdeleted	maxcpu	avgcpu	p95maxcpu	vmcategory	vmcorecountbucket	vmmemorybucket	lifetime	corehour
count	7.847800e+04	7.847800e+04	78478.000000	78478.000000	78478.000000	78478.0	78478.000000	78478.000000	78478.000000	78478.000000
mean	1.638740e+05	2.440948e+06	87.632671	11.547535	35.658455	1.0	3.109916	10.279467	632.520557	1947.707307
std	4.355987e+05	4.269736e+05	16.559472	12.865582	30.716842	0.0	2.871373	12.431971	169.328338	1923.338134
min	0.000000e+00	6.027000e+05	0.042408	0.006436	0.029750	1.0	2.000000	2.000000	119.750000	239.500000
25%	0.000000e+00	2.591400e+06	83.106668	3.686276	10.971996	1.0	2.000000	2.000000	719.833333	1439.666667
50%	0.000000e+00	2.591400e+06	95.256914	6.372755	21.407463	1.0	2.000000	4.000000	719.833333	1439.666667
75%	0.000000e+00	2.591400e+06	98.177427	14.545045	57.353884	1.0	4.000000	8.000000	719.833333	1887.166667
max	2.067300e+06	2.591400e+06	100.000000	99.328012	100.000000	1.0	32.000000	64.000000	719.833333	23034.666667

Figure 5.2.2.8: Plot showing the interactive versus delay-insensitive vms for SVR

5.2.2.5. RNN LSTM Model

```
# plot train and validation loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model train vs validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```



```
            Plot of LSTM model training v/s validation loss
```

Figure 5.2.2.9: Plot showing the LSTM vs Validation loss for RNN LSTM

Plot for minimum CPU utilization

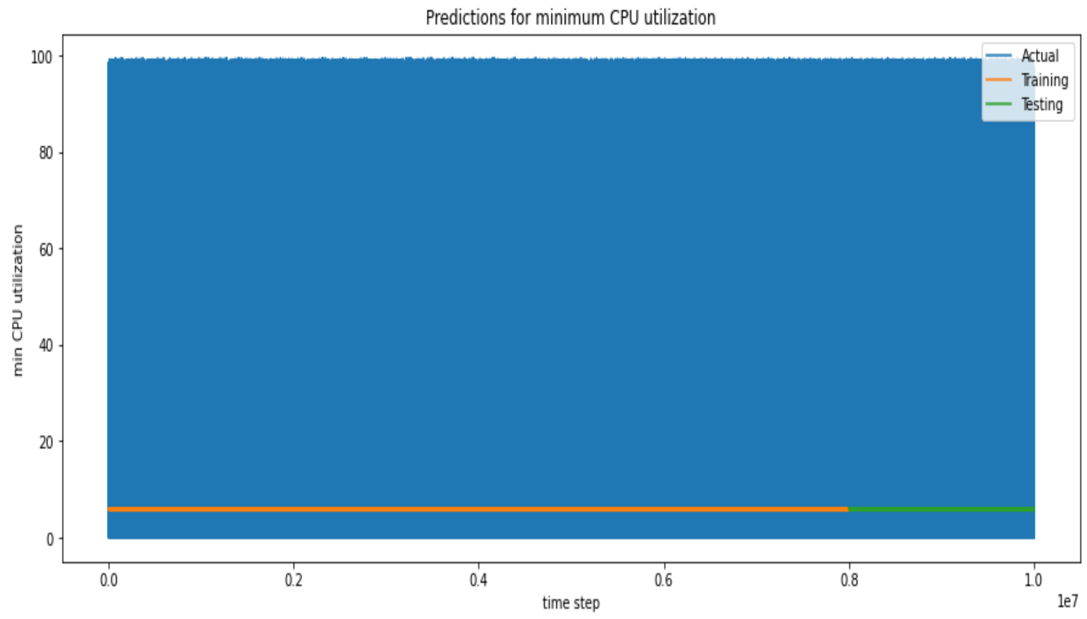


Figure 5.2.2.10: Plot showing minimum CPU utilization for RNN LSTM

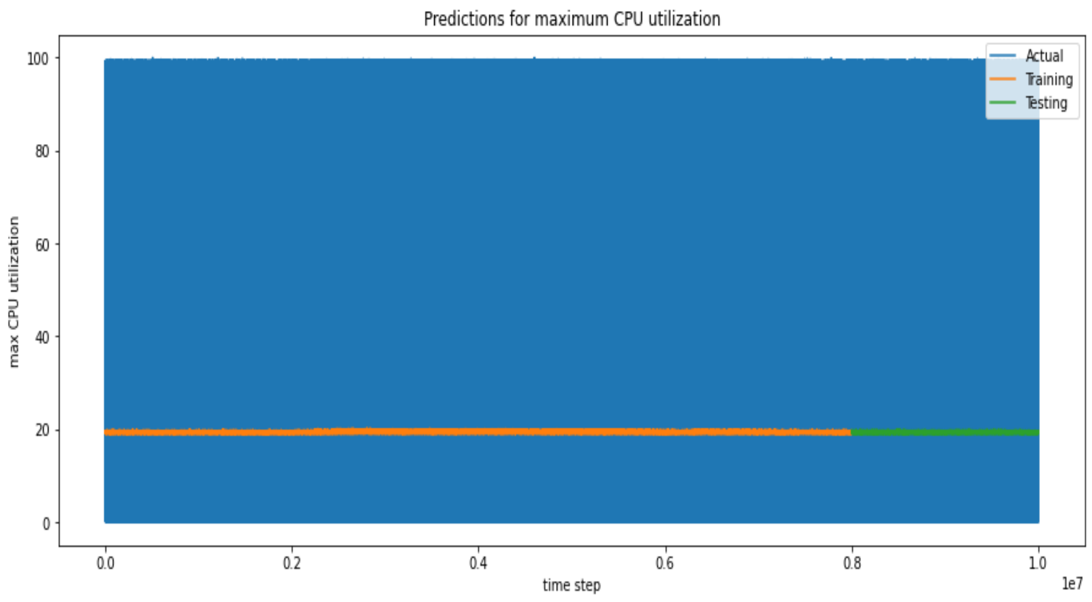


Figure 5.2.2.11: Plot showing maximum CPU utilization for RNN LSTM

5.2.2.6. Autoencoder LSTM Model

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model train vs validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```

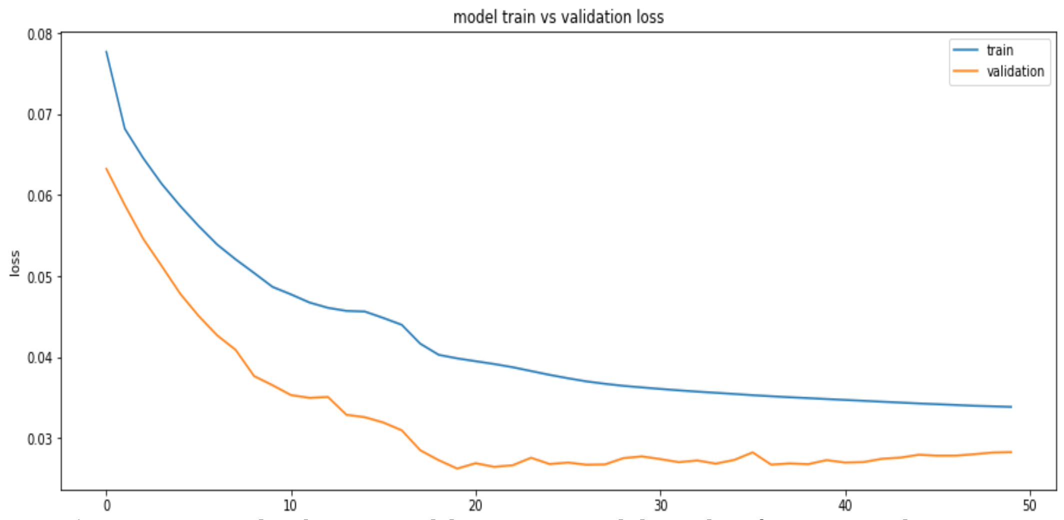


Figure 5.2.2.12: Plot showing model train versus validation loss for Autoencoder LSTM

5.2.3. Results for virtual machine memory prediction

As described in chapter 4 we have also considered the vm memory component and have run the models to predict the

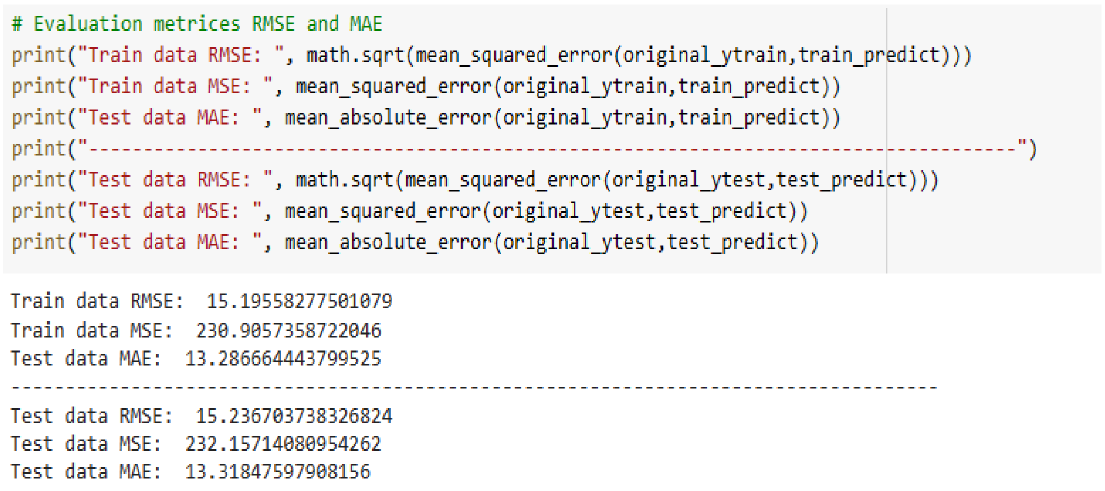
5.2.3.1. Random Forest Regressor

```
# Lets Do the prediction

train_predict=regressor.predict(X_train)
test_predict=regressor.predict(X_test)

train_predict = train_predict.reshape(-1,1)
test_predict = test_predict.reshape(-1,1)

print("Train data prediction:", train_predict.shape)
print("Test data prediction:", test_predict.shape)
```

```
Train data prediction: (1886877, 1)
Test data prediction: (808659, 1)
```

Transform:

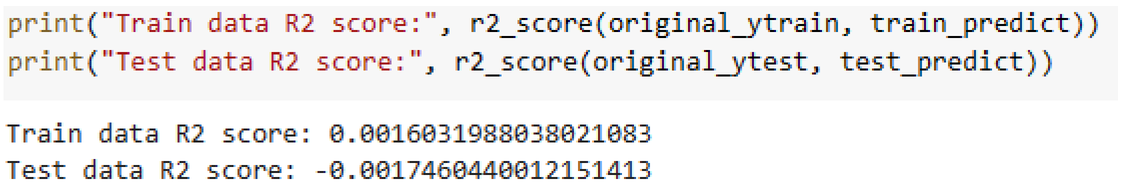
```
# Transform back to original form

train_predict = scaler.inverse_transform(train_predict)
test_predict = scaler.inverse_transform(test_predict)
original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1))
original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))
```



```
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------------------")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

```
Train data RMSE:  15.19558277501079
Train data MSE:  230.9057358722046
Test data MAE:  13.286664443799525
-------------------------------------------------------------------------------------
Test data RMSE:  15.236703738326824
Test data MSE:  232.1571408954262
Test data MAE:  13.31847597908156
```

Explained variance regression score
The explained variance score explains the dispersion of errors of a given dataset, and the formula is written as follows: Here, a nd Var(y) is the variance of prediction errors and actual values respectively. Scores close to 1.0 are highly desired, indicating better squares of standard deviations of errors.

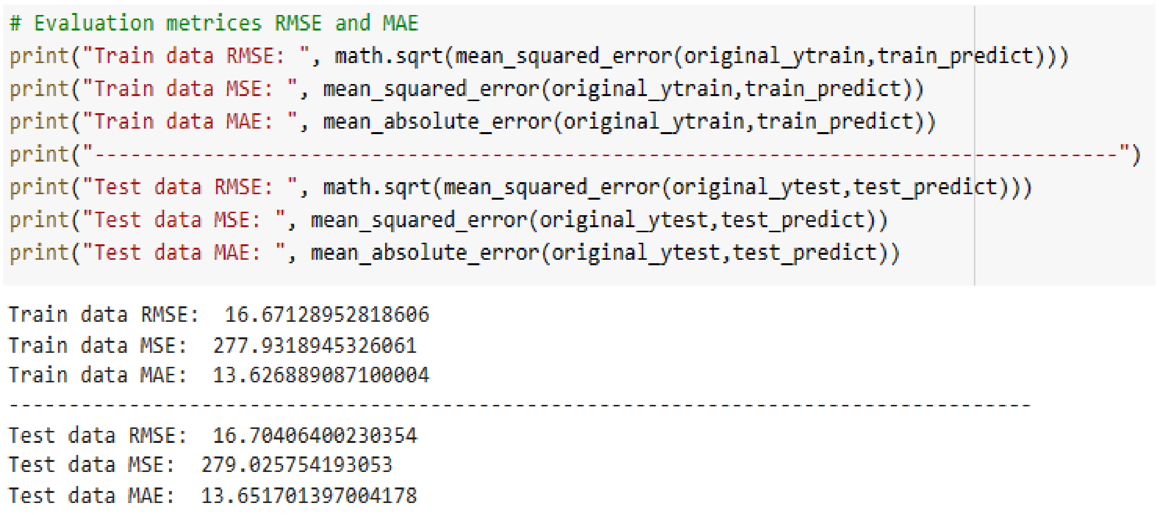


```
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

```
Train data R2 score: 0.0016031988038021083
Test data R2 score: -0.0017460440012151413
```

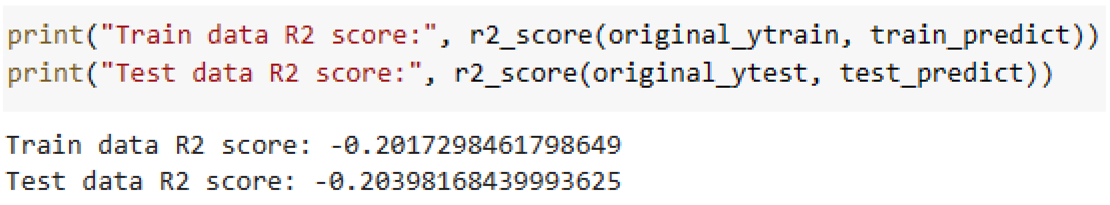
R2 score for regression
R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.
1 = Best
0 or < 0 = not good

5.2.3.2. KNN



```
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Train data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------------------")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

```
Train data RMSE:  16.67128952818606
Train data MSE:  277.9318945326061
Train data MAE:  13.626889087100004
-------------------------------------------------------------------------------------
Test data RMSE:  16.70406400230354
Test data MSE:  279.025754193053
Test data MAE:  13.651701397004178
```



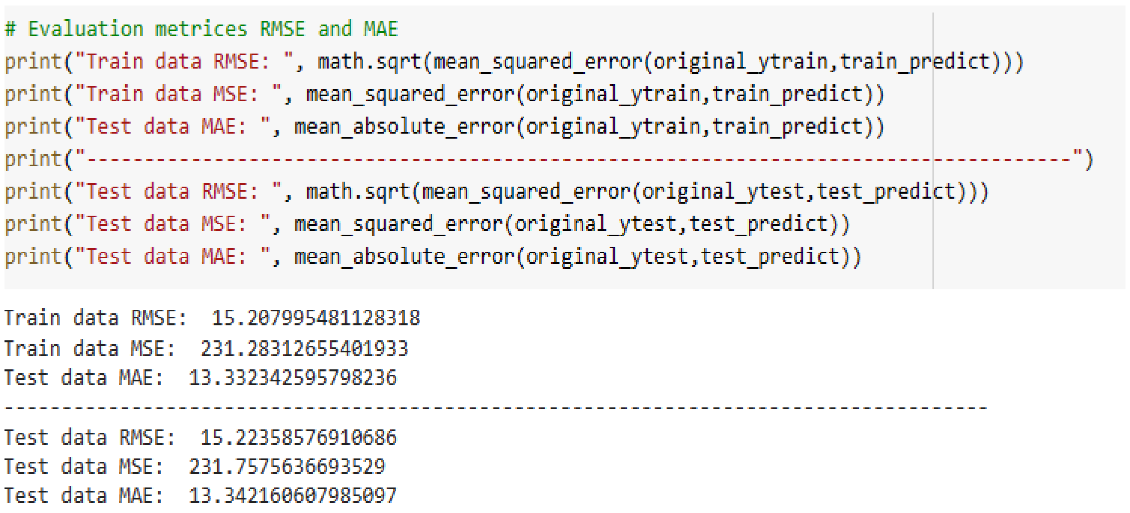
```
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

```
Train data R2 score: -0.2017298461798649
Test data R2 score: -0.20398168439993625
```

R2 score for regression
R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

1 = Best
0 or < 0 = not good

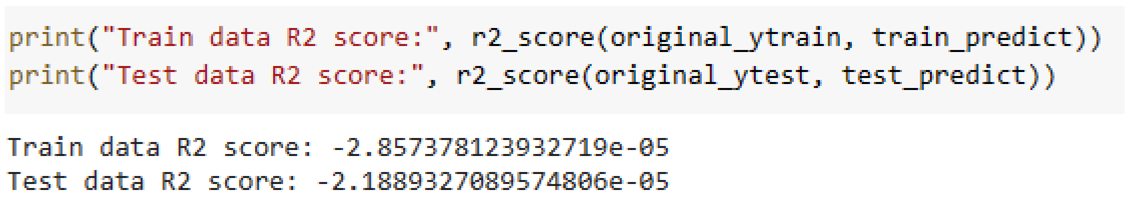
5.2.3.3. LSTIM



```
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------------------")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

```
Train data RMSE:  15.207995481128318
Train data MSE:  231.28312655401933
Test data MAE:  13.332342595798236
-------------------------------------------------------------------------------------
Test data RMSE:  15.22358576910686
Test data MSE:  231.7575636693529
Test data MAE:  13.342160607985097
```

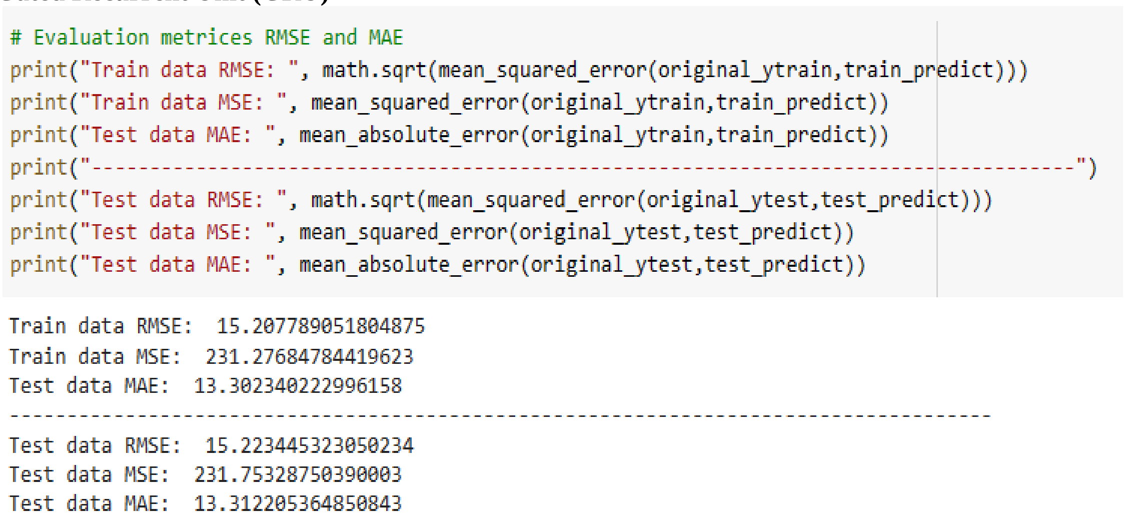
Explained variance regression score
The explained variance score explains the dispersion of errors of a given dataset, and the formula is written as follows: Here, and Var(y) is the variance of prediction errors and actual values respectively. Scores close to 1.0 are highly desired, indicating better squares of standard deviations of errors.



```
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

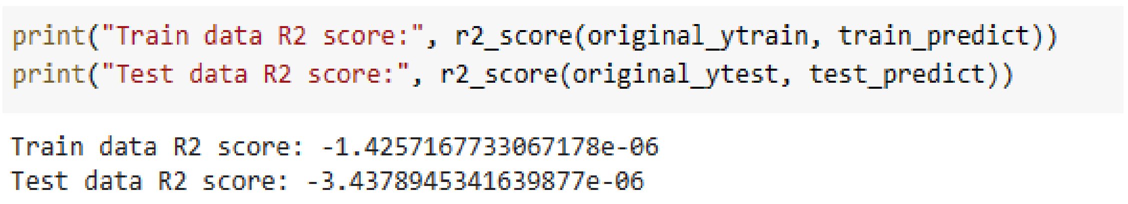
```
Train data R2 score: -2.857378123932719e-05
Test data R2 score: -2.1889327089574806e-05
```

5.2.3.4. Gated Recurrent Unit (GRU)



```
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------------------")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

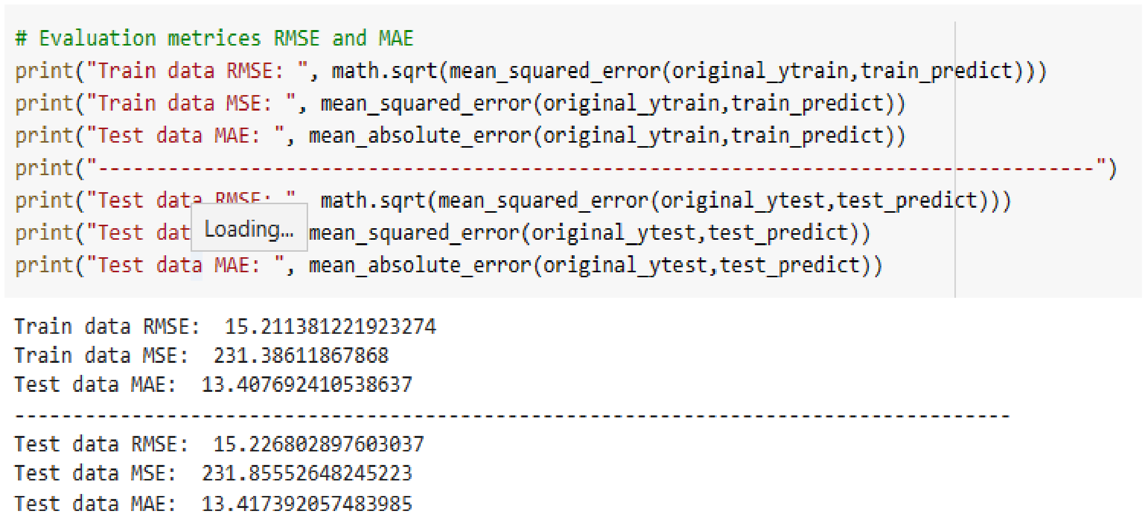
```
Train data RMSE:  15.207789051804875
Train data MSE:  231.27684784419623
Test data MAE:  13.302340222996158
-------------------------------------------------------------------------------------
Test data RMSE:  15.223445323050234
Test data MSE:  231.75328750390003
Test data MAE:  13.312205364850843
```



```
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

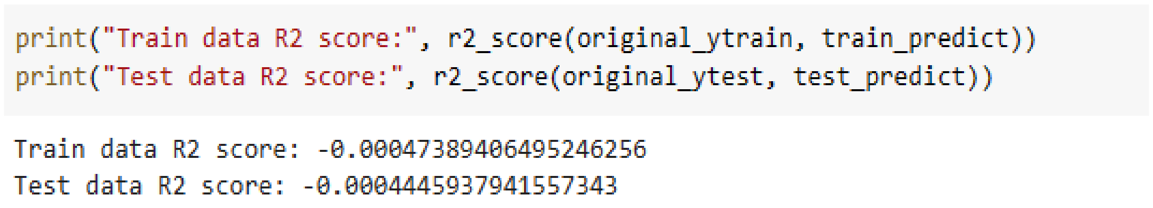
```
Train data R2 score: -1.4257167733067178e-06
Test data R2 score: -3.4378945341639877e-06
```

5.2.3.5. Combination of LSTM & GRU



```
# Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-------------------------------------------------------------------------------------")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test dat Loading… mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

```
Train data RMSE:  15.211381221923274
Train data MSE:  231.38611867868
Test data MAE:  13.407692410538637
-------------------------------------------------------------------------------------
Test data RMSE:  15.226802897603037
Test data MSE:  231.85552648245223
Test data MAE:  13.417392057483985
```

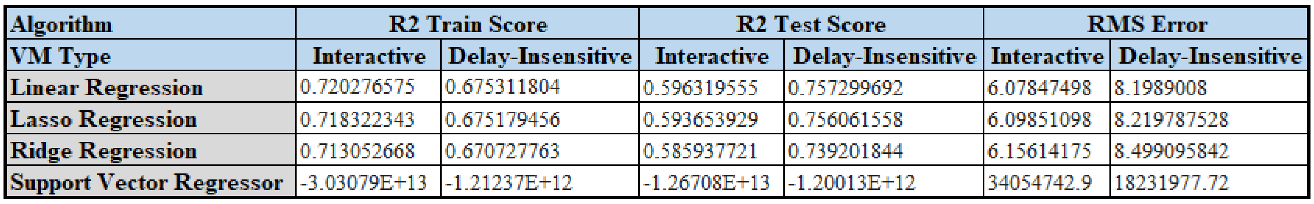


```
print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

```
Train data R2 score: -0.00047389406495246256
Test data R2 score: -0.0004445937941557343
```

5.3. Summary of model results and comparisons

VM CPU: For predicting the **vm cpu usage behavior** I had run the below algorithms and the R squared values for them along with the RMS Error values can be seen as per the table below.



Algorithm	R2 Train Score		R2 Test Score		RMS Error	
VM Type	Interactive	Delay-Insensitive	Interactive	Delay-Insensitive	Interactive	Delay-Insensitive
Linear Regression	0.720276575	0.675311804	0.596319555	0.757299692	6.07847498	8.1989008
Lasso Regression	0.718322343	0.675179456	0.593653929	0.756061558	6.09851098	8.219787528
Ridge Regression	0.713052668	0.670727763	0.585937721	0.739201844	6.15614175	8.499095842
Support Vector Regressor	-3.03079E+13	-1.21237E+12	-1.26708E+13	-1.20013E+12	34054742.9	18231977.72

Table 5.3 1: Summary of results for vm cpu prediction models

RNN LSTIM Model Results:
Additionally, I had also ran the RNN LSTM model with the below
RNN configuration: Lookback = 40-time steps
Trainable parameters: 143
4 LSTM units
3 output units

Optimizer used: Adam optimizer
Learning rate: 0.001
Minimizing mean square error loss
Train Score: 16.84 RMSE
Test Score: 16.50 RMSE

Autoencoder with LSTM Model Results:
The comparison on the model train graph versus validation graph can be seen as below

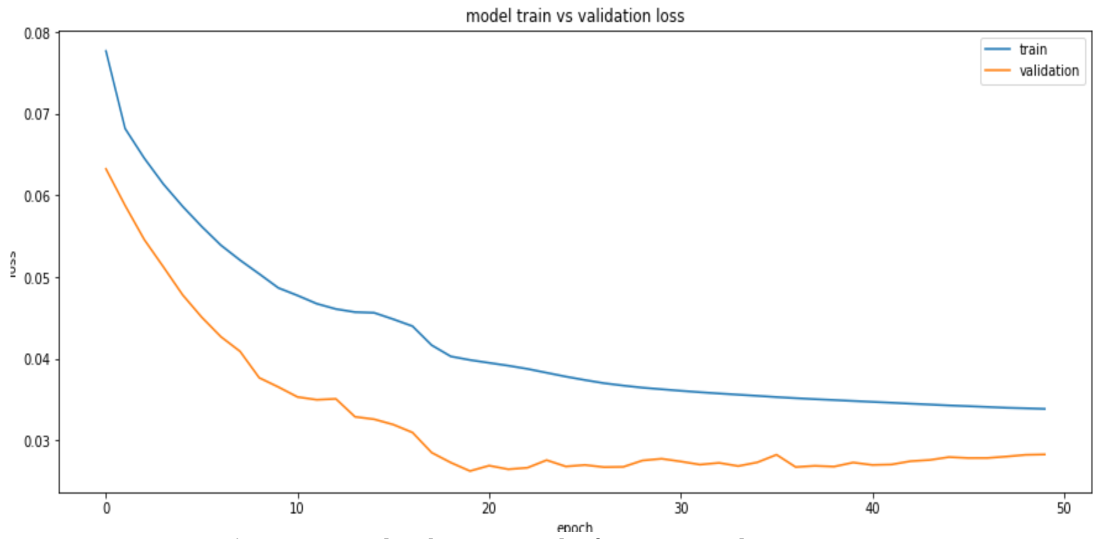


Figure 5.2.2: Plot showing results for Autoencoder using LSTM

VM Memory Bucket: For predicting the **vm memory usage** I had run the below algorithms and the R squared values for them along with the RMS Error values, MSE, MAE, Variance Regression Score, Mean Gamma deviation and Mean Poisson Deviance can be seen as per the table below.

Table 5.3 2: Summary of results for vm memory prediction models

Algorithm	RMSE		MSE		MAE		Variance Regression Score		R2 Score		Mean Gamma Deviation		Mean Poisson Deviance	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Random Forest	15.2	15.2	230.91	232.2	13.29	13.3	0.001603	-0.00175	0.0016032	-0.00175	0.936	0.94	12.929	12.997
K-Nearest Neighbor	16.67	16.7	277.93	279	13.63	13.7	-0.20103	-0.20333	-0.20173	-0.20398	1.21	1.21	16.054	16.116
LSTM	15.21	15.2	231.28	231.8	13.33	13.3	3.71E-11	3.90E-11	-2.86E-05	2.19E-05	0.938	0.94	12.953	12.971
GRU	15.21	15.2	231.28	231.8	13.3	13.3	0	0	-1.43E-06	-3.44E-06	0.938	0.94	12.952	12.971
LSTM+GRU	15.21	15.2	231.39	231.9	13.41	13.4	0	0	-0.000474	-0.00044	0.938	0.94	12.959	12.977

6. Conclusions and Recommendations

6.1. Introduction

This chapter of the thesis work deals with the conclusions derived after running all the models and tabulating their results leading to effective comparison of the same. The results have been derived by bifurcating the vm cpu and vm memory and running algorithms separately for each prediction of cpu and memory. Finally based on the results obtained suitable models are to be selected which can be called the best models for predicting the vm cpu and vm memory for future workloads. The results of the models run also has helped me to predict the cost contributing factors and how cost can be controlled by controlling the scaling of the clusters.

6.2. Discussion & Conclusions derived

I chose to derive the cost related parameter as it is a critical parameter when considering cloud cost. As seen from the multiple linear regression model, I have seen that there is a linear relation between cost, vm memory bucket, lifetime and core count of the vCpu. Hence if I can control and predict the duration of usage of higher values of memory bucket and core count, I can clearly control the cost. This leads us to the concept of up scaling and down scaling of the resources leading to cost savings. It is seen that horizontal scaling of increasing the number of cores is cost effective rather than vertical scaling of a higher degree of vCpu used along with the memory component.

The second part of my prediction deals with vm cpu and the dataset leads me to bifurcate the vm's by their respective vm categories and thereby predicting the vm cpu usage by running a set of models. I have chosen the 95th percentile CPU usage data as I found this parameter to be more accurate in predicting cpu usage behavior. I have chosen 'n' number of models starting with the basic regression models to predict the vm cpu usage and utilization over a duration of time. Both ridge regression and RNN based LSTM model are found to provide good results to predict the future values of cpu usage which can be beneficial for resource provisioning and cost predictions when we extrapolate the two components in the future.

The last part of my predictive models for vm memory utilization leads me to derive the vm memory consumption based on 'n' number of models. These models helped me to predict effectively the memory consumption for the clusters spun by the virtual machines. The combination of LSTM and GRU model has been seen to perform well to predict the vm memory utilization for future workloads.This is again an important parameter to not only help resource scheduling in the future but also to predict costs in the future based on the memory consumption behavior.

As I have run several models to predict various parameters, the focus was not much on getting the best model possible using hyper-parameter tuning but to create a generic frameworkwhich can be extended to larger datasets with huge number of vm traces and of course being cost optimized at the same time.

6.3. Contribution to Knowledge

What I have tried to establish through the research work is to highlight the importance of cloud costing as we are adopting more and more cloud technologies and infrastructure. Cloud cost should be the driving factor which leads to optimization of other cloud parameters for reaching a cost optimized resource provisioned and efficient infrastructure model. In the current setup of the research work, the best model/method is determined on complete dataset.

This research work primarily focuses on deriving a model which is got through a hybrid approach dealing with the cost, vm cpu and vm memory separately and applying different models to achieve the results on the same dataset with different feature engineering applied based on need.

6.4. Future Recommendations

While I have tried to deal with the prediction mechanism of various infrastructure parameters separately in my research work, I see several other options as well for future research scope.

A consolidated model which considers all the predictive parameters and effectively predicts all the 3 predictions accurately and efficiently, is something that can be pursued. Additionally, a larger dataset containing several years of vm traces over different geographical regions can also be used for predictions and can lead to different seasonality parameters and data to be included and analyzed based on region/country/geography. Of course, with larger dataset we would need to consider usage of GPUs and TPUs.While I have considered the Azure vm traces,similar research can also be taken into consideration for vm traces of other cloud providers and how that differs in terms of Azure workloads. While most of the research has been done on costing with respect to cloud features, research on real time vm traces is few and far between. Hence the need to analyze the real time vm traces is essential to know the workload behavior, cost and other associated parameters.

References

[1] Al-Faifi, A., Song, B., Hassan, M.M., Alamri, A. and Gumaei, A., (2019) A hybrid multi criteria decision method for cloud service selection from Smart data. *Future Generation Computer Systems*, [online] 93, pp.43–57. Available at: https://doi.org/10.1016/j.future.2018.10.023.

[2] Al-Faifi, A.M., Song, B., Hassan, M.M., Alamri, A. and Gumaei, A., (2018) Data on performance prediction for cloud service selection. *Data in Brief*, 20, pp.1039–1043.

[3] Aldossary, M. and Djemame, K., (2018) Performance and energy-based cost prediction of virtual machines auto-scaling in clouds. *Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018*, pp.502–509.

[4] Aldossary, M., Djemame, K., Alzamil, I., Kostopoulos, A., Dimakis, A. and Agiatzidou, E., (2019) Energy-aware cost prediction and pricing of virtual machines in cloud computing environments. *Future Generation Computer Systems*, [online] 93, pp.442–459. Available at: https://doi.org/10.1016/j.future.2018.10.027.

[5] Barnwal, A., Jangade, R. and Pugla, S., (2019) Analyzing and predicting the allocation and utilization of resources in cloud computing system. *Proceedings of the 9th International Conference On Cloud Computing, Data Science and Engineering, Confluence 2019*, pp.56–62.

[6] Biswas, A., Majumdar, S., Nandy, B. and El-Haraki, A., (2015) Automatic resource provisioning: A machine learning based proactive approach. *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, 2015-FebruFebruary, pp.168–173.

[7] Calzarossa, M.C., Della Vedova, M.L. and Tessera, D., (2019a) A methodological framework for cloud resource provisioning and scheduling of data parallel applications under uncertainty. *Future Generation Computer Systems*, [online] 93, pp.212–223. Available at: https://doi.org/10.1016/j.future.2018.10.037.

[8] Calzarossa, M.C., Vedova, M.L.D., Massari, L., Nebbione, G. and Tessera, D., (2019b) Modeling and predicting dynamics of heterogeneous workloads for cloud environments. *Proceedings - IEEE Symposium on Computers and Communications*, 2019-June, pp.14–20.

[9] Chaisiri, S., Lee, B.S. and Niyato, D., (2012) Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 52, pp.164–177.

[10] Cortez, E., Russinovich, M., Bonde, A., Fontoura, M., Muzio, A. and Bianchini, R., (2017) Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms? *SOSP 2017 - Proceedings of the 26th ACM Symposium on Operating Systems Principles*, pp.153–167.

[11] Di, S., Kondo, D. and Cirne, W., (2012) Host load prediction in a Google compute cloud with a Bayesian model. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*.

[12] Ding, Q., Tang, B., Manden, P. and Ren, J., (n.d.) A Learning-based Cost Management System for Cloud Computing. pp.362–367.

[13] Djemame, K., Bosch, R., Kavanagh, R., Alvarez, P., Ejarque, J., Guitart, J. and Blasi, L., (2017) PaaS-IaaS inter-layer adaptation in an energy-aware cloud environment. *IEEE Transactions on Sustainable Computing*, 22, pp.127–139.

[14] Farahnakian, F., Pahikkala, T., Liljeberg, P. and Plosila, J., (2013) Energy Aware Consolidation Algorithm based on K-nearest Neighbor Regression for Cloud Data Centers.

[15] Genaud, S. and Gossa, J., (2011) Cost-wait trade-offs in client-side resource provisioning with elastic clouds. *Proceedings - 2011 IEEE 4th International Conference*

on Cloud Computing, CLOUD 2011, pp.1–8.

[16] Humphrey, M., (2010) '. pp.41–48.

[17] Jagannatha, S., Shravan, N.S. and Kavya, S., (2017) Cost performance analysis: Usage of resources in cloud using Markov-chain model. *2017 4th International Conference on Advanced Computing and Communication Systems, ICACCS 2017*.

[18] Lansky, J.A.N., Ali, S., Mohammadi, M., Majeed, M.K. and Rahmani, A.M., (2021) Deep Learning-Based Intrusion Detection Systems : A Systematic Review. *IEEE Access*, 9, pp.101574–101599.

[19] Ma, A., Zhang, C., Zhang, B. and Zhang, X., (2016) Cost optimization oriented dynamic resource allocation for service-based system in the cloud environment. *Proceedings - 2016 IEEE International Conference on Web Services, ICWS 2016*, pp.700–703.

[20] Mark, C.C.T., Niyato, D. and Chen-Khong, T., (2011) Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp.348–355.

[21] Minarolli, D. and Freisleben, B., (2014) Cross-Correlation Prediction of Resource Demand for Virtual Machine Resource Allocation in Clouds.

[22] Qu, J., (2018) An Intelligent Fault Detection Method based on Sparse Auto-Encoder for Industrial Process Systems : A Case Study on Tennessee Eastman Process Chemical System.

[23] Rak, M., Cuomo, A. and Villano, U., (2013) Cost/performance evaluation for cloud applications using simulation. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, pp.152–157.

[24] Shen, Y., Chen, H., Shen, L., Mei, C. and Pu, X., (2014) Cost-optimized resource provision for cloud applications. *Proceedings - 16th IEEE International Conference on High Performance Computing and Communications, HPCC 2014, 11th IEEE International Conference on Embedded Software and Systems, ICESS 2014 and 6th International Symposium on Cyberspace Safety and Security*, pp.1060–1067.

[25] Singh, P., Gupta, P. and Jyoti, K., (2019) TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud. *Cluster Computing*, [online] 222, pp.619–633. Available at: https://doi.org/10.1007/s10586-018-2868-6.

[26] Strunk, A., (2013) A lightweight model for estimating energy cost of live migration of virtual machines. *IEEE International Conference on Cloud Computing, CLOUD*, pp.510–517.

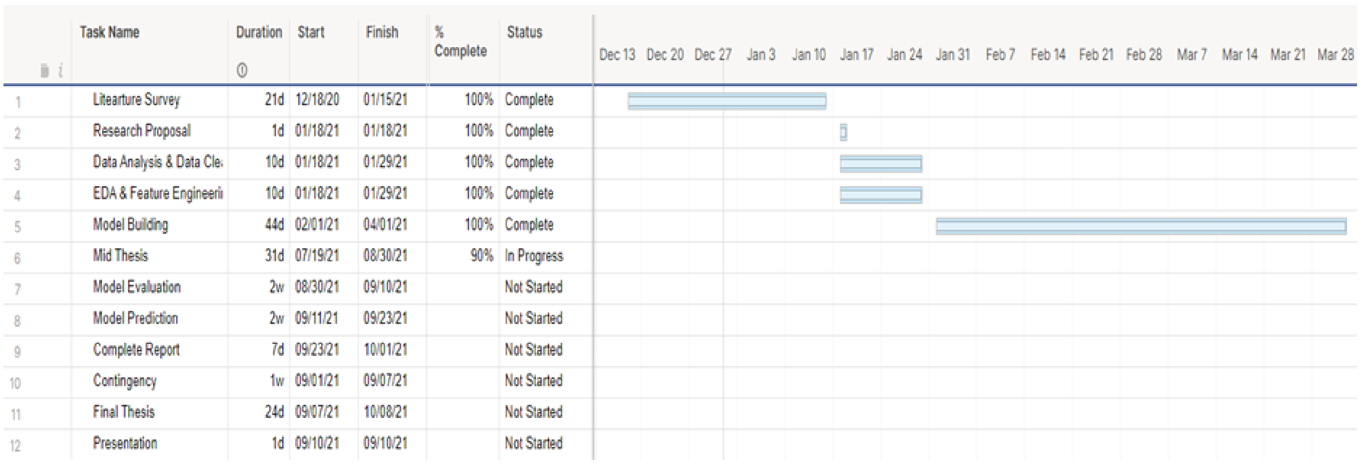
[27] Wan, B., Dang, J., Li, Z., Gong, H., Zhang, F. and Oh, S., (2020) Modeling Analysis and Cost-Performance Ratio Optimization of Virtual Machine Scheduling in Cloud Computing. 317, pp.1518–1532.

[28] Wang, W., Niu, D., Li, B. and Liang, B., (2013) Dynamic cloud resource reservation via cloud brokerage. *Proceedings - International Conference on Distributed Computing Systems*, pp.400–409.

[29] Yang, J., Liu, C., Shang, Y., Cheng, B., Mao, Z., Liu, C., Niu, L. and Chen, J., (2014) A cost-aware auto-scaling approach using the workload prediction in service clouds. *Information Systems Frontiers*, 161, pp.7–18.

[30] Yousefyan, S., Dastjerdi, A.V. and Salehnamadi, M.R., (2013) Cost effective cloud resource provisioning with imperialist competitive algorithm optimization. *IKT 2013 - 2013 5th Conference on Information and Knowledge Technology*, pp.55–60.

Appendix A: Research Plan



	Task Name	Duration	Start	Finish	% Complete	Status
1	Litearture Survey	21d	12/18/20	01/15/21	100%	Complete
2	Research Proposal	1d	01/18/21	01/18/21	100%	Complete
3	Data Analysis & Data Cle:	10d	01/18/21	01/29/21	100%	Complete
4	EDA & Feature Engineeri	10d	01/18/21	01/29/21	100%	Complete
5	Model Building	44d	02/01/21	04/01/21	100%	Complete
6	Mid Thesis	31d	07/19/21	08/30/21	90%	In Progress
7	Model Evaluation	2w	08/30/21	09/10/21		Not Started
8	Model Prediction	2w	09/11/21	09/23/21		Not Started
9	Complete Report	7d	09/23/21	10/01/21		Not Started
10	Contingency	1w	09/01/21	09/07/21		Not Started
11	Final Thesis	24d	09/07/21	10/08/21		Not Started
12	Presentation	1d	09/10/21	09/10/21		Not Started

Appendix B: Research Proposal

Cloud Cost Resource Optimized Model based on Predictive ML Approach

Apurba Kumar Roy

(Student Id: 944585)

Master of Science in Data Science, Liverpool John Moore's University

Supervisor: Kaustubh Vaman Sakhare
Research Proposal

Abstract: ***There has been tremendous growth and development across the globe with respect to cloud adoption and usage. The traditional on-premise servers have given way to global cloud infrastructure with most of the vendors like Amazon, Microsoft and Google setting up dedicated cloud infrastructure and management provisions. These developments have allowed companies and organizations to concentrate more on building software applications and systems while the infrastructure part is totally taken care by the cloud providers themselves. While there has been rapid cloud growth and adoption across the industry, it has come with its share of cons too. Organizations worldwide started moving rapidly to cloud systems, migrating data and application and paying only for what they used in terms of VMs, Clusters, CPUs, Memory and components with zero maintenance cost associated with the infrastructure. But slowly few organizations began to realize that the cost that they started to pay for the usage of the Cloud infrastructure was becoming even more than what they used to pay for the traditional on-premises systems. Cloud adoption comes with numerous advantages, but when the resources and cost is not managed properly, the cost incurred can become huge and uncontrollable. This phenomenon requires an in-depth understanding and implementation of optimized resource management which would produce an optimized cloud cost resourcing model exploiting the workload characteristics and machine learning to produce substantial results. Using the real time Microsoft Azure workloads, we can predict the accurate optimized costing for future workloads which would help organizations to take informed decisions towards their cloud resource management and costing. This proposal intends to take a closer look at the various aspects/Characteristics of Microsoft Azure Cloud implementation through the production Virtual Machine workloads and how we can take into consideration the cost perspective to take corrective steps in order to maintain a balance between the cost towards infrastructure and performance, at the same time reap the benefits of the cloud infrastructure in a more efficient and cost-effective way.***

1. Introduction

1.1 Background

Cloud computing has been gaining popularity all around the world in the last couple of years. There has been rapid strides of improvement and growth in terms of the services and features that cloud vendors now provide. The catalyst for these rapid changes has been the competition in the marketplace among these vendors like Amazon, Microsoft and Google to capture the cloud computing marketplace. The biggest beneficiary of these technological advances due to competition have been the organizations who have realized the huge benefit that cloud computing provides at the same time reduces the overhead of maintenance of servers and datacenters of their own. This has led organizations to quickly incorporate cloud computing as part of their organization strategy towards implementation and adoption moving away from the traditional on-premises servers. While the benefits of moving into cloud computing from the traditional server-based architecture are many, cost savings being one of the top 4 benefits as organizations only pay for the duration, they use the cloud infrastructure. However, if not properly implemented, it can also have adverse effects in terms of quickly becoming costlier than even the traditional servers.

This research work intends to present a predictive analytics approach to the resource and cost management aspects by considering real time production workload traces containing virtual machine and component information gathered over a period of time for spanning different regions and resource groups. Large vendors have been providing information of cost to customers based on current usage, but that information is quite limited. What is novel in this research work is that this is the first time both optimized resource management and cost are combined together, and predictive analytics used on real time workloads to predict the optimized cost and resources to be used and factors effecting the cost in the long run. Also, this research work intends to do an in-depth analysis on the cloud component factors along with different combinations of configurations which can reduce the cost and empowers the organizations to take informed decisions on the capacity and configuration planning of the resources.

1.2Dataset

The dataset for the study is taken from public releases of Microsoft Azure which contains traces from Virtual Machine workloads and also Azure function invocations. Details also include VM requests made and their lifetime, CPU Utilization, Number of cores used and the duration of the workload along with utilization readings. As part of this research work, we also intend to have a closer look at the optimized cost and hence we would be mapping the Azure VM pricing information from Microsoft Azure Pricing details.

1.3 Related Work

Research on cloud related virtual machine parameters and its relation to cost management and optimization field are few in real time which could give us deeper insights into Resource and Cost Management together and how each one relates to the other. For Example, there has been prior work done with respect to optimization engines for resource management like:In 2020, M.Shahrad et al.(Shahrad et al., 2020) deals with the function invocation that are available during workload run and which function invocation occurs at what time in order to optimize the functions through the variables of invocation time frequencies and patterns within. But even this research work does talk about the cost component that comes into picture when optimization w.r.t. cloud is taken into consideration

In 2019, M. Aldossary et al. (Aldossary et al., 2019) published a research work which talks about cost prediction based on the energy consumption of VMs and thereby predict the cost only. The one parameter that is the focus here is that it only talks about overall VM cost based on power consumption.

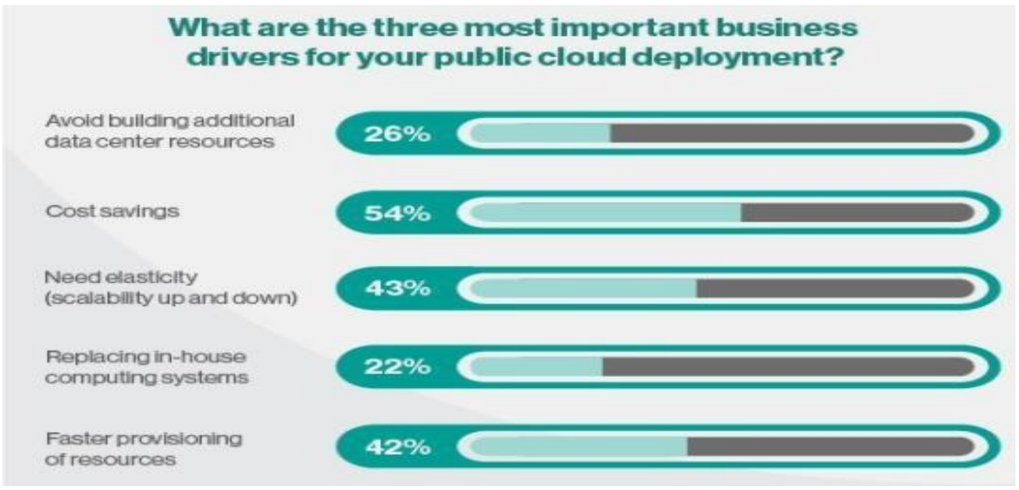
In 2019, P. Singh et al. (Singh et al., 2019)talks about workload prediction required for auto-scaling of cloud environment

In 2017, K. Djemame et al. (Djemame et al., 2017) talks about the energy efficiency analysis of cloud computing environments.

In 2009, Lori Moore-Merell et al.(Moore-Merrell, 2009) takes into account Microsoft Resource Central engine which automatically predicts the type of resource management that workloads would require based on virtual machine traces, but in no way has any co-relation to the optimized cost factor associated with Cloud computing for various workloads.

2. Problem Statement

With the advent of Cloud computing, there has been rapid expansion and migration to cloud infrastructure from the traditional on-premise structure to large cloud providers such as Microsoft Azure, Amazon Web Services and Google Cloud Platform (GCP). There have been rapid innovation and new features being introduced by cloud providers very frequently owning to increased market place competition. All these advanced features are exposed and are being made available to the customers. However, on one hand these features are attractive and solves a lot of problems, on the other hand there is a price that is associated with these features as well. One of the primary reasons of cloud adoption is the cost savings that it brings with it as shown in the diagram below (Image adopted from internet). Hence efficient handling of the resources along with the optimized cost model is the need of the hour without which organizations would have to pay a hefty price due to incorporations of unmanaged infrastructure spanning all geographical locations.



Practical and efficient resource and cost management for customers with deeper understanding of the Infrastructure components key characteristics is something that should be implemented across all organizations in order to reap the benefits of cloud computing. Moreover, if these can characteristics and cost associated can be accurately predicted on time, customers can be highly benefited and may adjust their infrastructure in a way that suits their budget as well as satisfies their requirements in a more efficient manner. Additionally, the current prediction of Cloud computing cost which is being provided as a side service by large cloud computing vendors is not comprehensive enough to take into account all the parameters specifically VMs size, CPU Utilization and its relation to CPU cores, VM Utilization, cold starts in VMs, vertical scaling of cluster configurations and horizontal scaling of cluster configuration. This research work intends to also explore the insights into the tradeoff between cloud resource component utilization, its cost and the overall performance gained in terms of job runs holistically in order to enable the organizations take informed decisions accordingly. In this research work I intend to look into all factors and not a single parameter alone predicting the optimized cost.

3. Research Questions (If Any)

How can we provide an efficient predictive analysis for cloud implementation of resource and cost management to customers by looking at the workloads running in an organization so that customers can adjust their configurations to suit the budget requirements and at the same time be efficient in managing the cloud resources?

4. Aim and Objectives

The main aim of this research is to propose a cloud resource managed and cost optimized prediction model which would take into account all the workloads running in an organization including the key characteristic components within the cloud infrastructure like VMs, Clusters, Memory, Cores etc.

The research objectives are formulated based on the aim of this study which are as follows:

- To suggest a suitable predictive model for cloud resource and cost management for customers for their workloads running with different cloud components
- Detailed analysis on the workload behavior including the VMs, Cores, Clusters, Resource Managers, CPU utilization, Memory and Costing associated with each. Subsequent prediction by taking into account all these factors.
- Identify the trends in workload behavior across different resource managers
- To compare, evaluate between the predictive models and select the best model which would produce optimized results

5. Significance of the study

The study proposed in this research work would help in formulating a ML based predictive analytics approach to solving the optimized cloud cost and resource management for Microsoft Azure workloads spanning different geographic locations, the usage of various cloud infrastructure functions and how these real time workloads have effect on the overall cost of the organization across different resource groups. The dataset selected for this study contains VM traces containing valuable information on the Cloud component parameters required to study and predict the cost as well as the effective and efficient resource utilizations for the workloads so that organizations can take corrective and informed decisions on the capacity planning and resource planning of the cloud resources leading to savings. In order to provide good performance, availability and reliability under various workload conditions, can be an expensive affair without the existence of an effective cost and resource management. It has a huge significance in the overall success of cloud strategy of every organization and effective implementation of the same would greatly help in adding value to any organization.

6. Scope of the Study

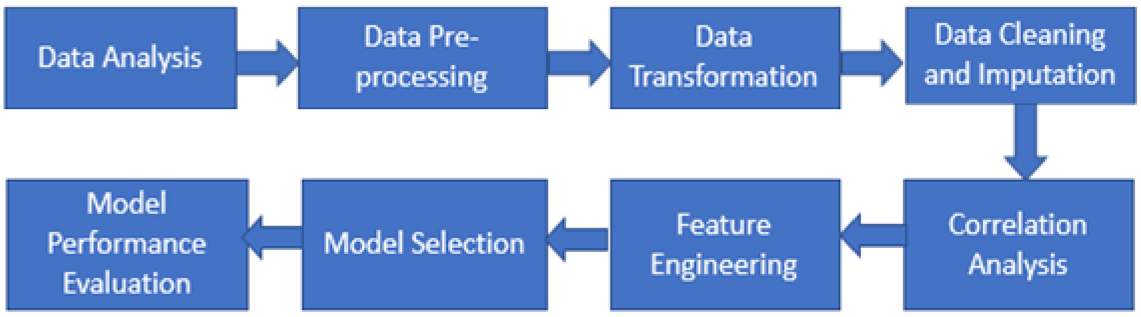
As part of the work presented in this research work, I intend to study the Microsoft Azure workloads spanning different geographic locations and containing valuable information on the different component parameters of cloud computing seen to be working when a particular job or jobs run in any region across Resource groups. The study would include characteristics of production Virtual Machine (VM) workloads and a thorough understanding of the servers, power consumption, CPU utilization, VM lifetime, Deployment size and its relation to the overall infrastructure cost. Additionally, this research work also takes a look at the Azure functions being invoked for the various production workloads and what is the impact of such invocation on the overall cost incurred to the organization.

7. Research Methodology

The research methodology to be employed during the study of the Cloud cost and resource management prediction involves key processes such as the selection of appropriate dataset, pre-processing the selected dataset, related transformation which are required on the dataset into comprehensible format, balancing the dataset, feature engineering to extract additional parameters and values, extracting the cost information from Microsoft provided pricing as per different VM configurations and implementing ML techniques and evaluating the algorithms performance using evaluation measures. These steps would form the back bone of the research work presented for the study.

7.1 Proposed Model Architecture

The Proposed Model Structure would contain the below steps:



1) **Data Analysis:** The dataset would be first analyzed to find out the parameters and variables across different workloads. Resource Manager groups and third-party VM information so that they can be bucketed

accordingly. Also, appropriate variables are identified and the costing information is also fetched from Microsoft pricing information to be store and analyzed against the workloads provided.

2) **Data Pre-Processing and Transformation:** The data from the Azure public dataset present in the form of workloads would be pre-processed to incorporate data which is generated each sec or minute from the VMs. These would be divided based on the Resource Groups for each VM and similar Resource Group VMs would be clubbed together for further analysis. Appropriate data transformation would be taken forward for the conversion of the categorical fields into numeric as well. Also, the VMs would be renamed and grouped together based on the similar configurations.
3) **Imputation of Data:** Appropriate imputation of data would be conducted so that the irrelevant data is cleansed. Wherever irrelevant data is found to be existing those would be treated appropriately.
4) **Diagrammatic Representation of the Co-related variables:** Appropriate code would be executed to check the co-relation of the variables and dependency analysis of the various VM parameters and components
5) **Feature Engineering:** Appropriate feature Engineering would be conducted in order to create specific variables which would be useful in the implementation of the algorithm. These would be in the form of VM and component parameters required to include
6) **Model Selection:** Multiple models would be utilized to predict the cost and resource optimized models for Azure workloads across different geographic locations and Availability zones. The models to be run for this prediction would be Regression, Decision tree, Time-series, XGBoost and Random Forest through the train and test data. As first steps the data set would be divided in the ratio of 70:30 with 70 being the training data set and 30 is the test data set.
7) **Performance Index:** Once the models have been run successfully, we need to compare the various performance indexes like R2, Accuracy, Mean Absolute Error (MAE), k-Cross validation
   - R2: Variance Explained by the Model/Total Variance, R2 is between 0-100.Higher value of R2 means better Regression Model fits the observations.
   - The value of K should also be suitably selected in order to achieve correct Results.
   - Accuracy: Prediction which was done correct/total testing done X100

The Prediction algorithm takes into account the VM behaviors like start time, end time, deployment parameter, workload segments and power consumption.

CPU Utilization: The average CPU utilization % is to be mapped to the performance as well as cost parameters while deploying a particular VM running similar workloads.

Right VM Sizing: VMs which are under the similar Resource groups must be sized based on the workload and time duration for which the cloud resources are required and utilized at least above 60% for efficiency in resource utilization as well as cost consciousness for the same.

Weekend and Weekday trend Analysis: Overall pattern to be observed for workloads which runs on weekdays compared to weekends and accordingly resource allocation to be predicted along with the right cost implication for the same.

Efficient Cluster Configuration: Based on the workload behavior efficient Cluster configuration with the right sized VM and CPU resource requirements can be predicted. Also, the cost can be brought down with the correct configuration parameters.

Server Maintenance: Appropriate server maintenance cycles can be predicted based on the workload dataset and cost can be adjusted accordingly for the maintenance time frame.

8. Resources Required

The following resources would be required for implementation of this research:

A GPU is required to perform the training for the following reasons:

- As the dataset contains huge data set containing data every second of the workloads.
- It is expected that the number of epochs might be high and for better performance.
- Complex operations to be performed and data size is going to inflate to a big extent

8.1 Software Required

- Python and associated libraries for Visualization

8.2 Technical Skills Required

- Python
- Regression Algorithm
- Time Series Algorithm
- Random Forest Algorithm
- XGBoost Algorithm

8.3 Domain Knowledge Required

- Cloud Computing Fundamentals
- Azure Cost Management Fundamentals
- Azure Cluster components knowledge
- Azure Cloud usage
- Virtual Machine fundamentals

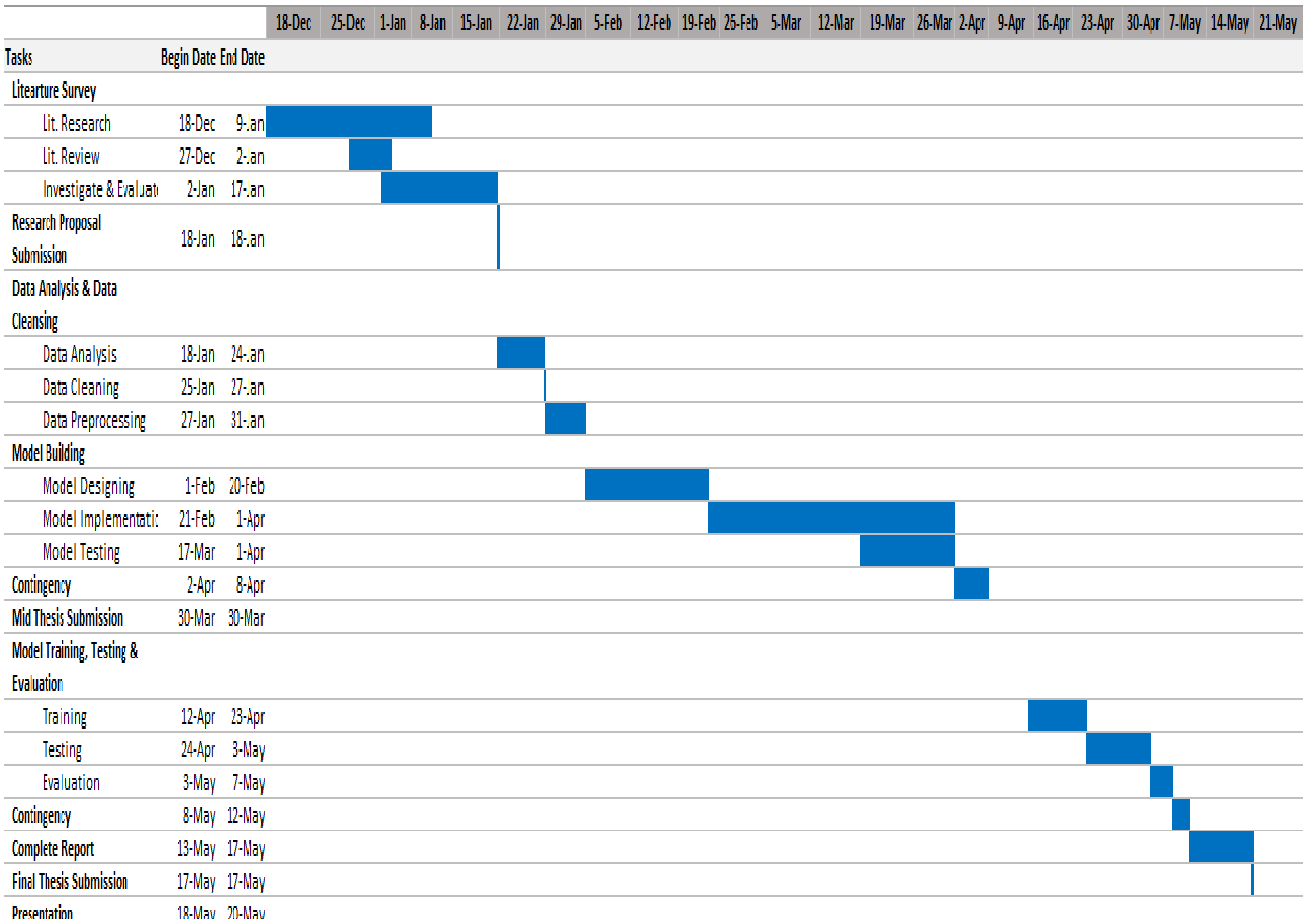
9. Expected Outcomes

In this research work, the prediction mechanism for the resource and cost optimized cloud infrastructure setup should be able to predict accurately the cost as well as efficient management of resource to produce optimum results.

This research work intends to analyze the different aspects of efficient Resource Management along with the cost implication, so as to provide an unbiased and correct picture to the organizations using the cloud infrastructure setup. This would help the organizations to take informed and correct decision regarding the future cloud strategy and utilization of resource at the same time keeping the cost factor in check.

10. Research Plan

The below figure shows the plan for this research represented in a Gantt chart format:



Tasks	Begin Date	End Date
Litearture Survey		
Lit. Research	18-Dec	9-Jan
Lit. Review	27-Dec	2-Jan
Investigate & Evaluat	2-Jan	17-Jan
Research Proposal Submission	18-Jan	18-Jan
Data Analysis & Data Cleansing		
Data Analysis	18-Jan	24-Jan
Data Cleaning	25-Jan	27-Jan
Data Preprocessing	27-Jan	31-Jan
Model Building		
Model Designing	1-Feb	20-Feb
Model Implementatic	21-Feb	1-Apr
Model Testing	17-Mar	1-Apr
Contingency	2-Apr	8-Apr
Mid Thesis Submission	30-Mar	30-Mar
Model Training, Testing & Evaluation		
Training	12-Apr	23-Apr
Testing	24-Apr	3-May
Evaluation	3-May	7-May
Contingency	8-May	12-May
Complete Report	13-May	17-May
Final Thesis Submission	17-May	17-May
Presentation	18-May	20-May

Acknowledgement

I would like to express my profound sense of gratitude and satisfaction to present the dissertation work on **Cloud Cost Resource Optimized Model Based on Predictive ML Approach for Infrastructure usage/workload Behavior**. This could not have been possible without the support, guidance and focused direction provided by my guide **Asst. Prof.Dr. Sahil Sharma**. I would like to express my gratitude and thankfulness for his valuable guidance and support through the entire period of the dissertation work. I would also like to thank my fellow students of master's in data science who kept me honest towards the various phases of the dissertation work. I would also like to thank our group mentor **Prof. Dr. Manoj Jayabalan,** Faculty of Engineering and Technology, LJMU, for all his sessions right from the starting of dissertation program which was the source of knowledge and guidance towards completion of thefinal dissertation work. Finally, I would like to thank my wife, my daughter and my family for their unwavering support throughout the entire thesis work.

References

[1] Aldossary, M., Djemame, K., Alzamil, I., Kostopoulos, A., Dimakis, A. and Agiatzidou, E., (2019) Energy-aware cost prediction and pricing of virtual machines in cloud computing environments. Future Generation Computer Systems, [online] 93, pp.442–459. Available at: https://doi.org/10.1016/j.future.2018.10.027.

[2] Djemame, K., Bosch, R., Kavanagh, R., Alvarez, P., Ejarque, J., Guitart, J. and Blasi, L., (2017) PaaS-IaaS inter-layer adaptation in an energy-aware cloud environment. IEEE Transactions on Sustainable Computing, 22, pp.127–139.

[3] Moore-Merrell, L., (2009) Resource centre. Fire Risk Management, JUN, pp.60–61.

[4] Shahrad, M., Fonseca, R., Goiri, Í., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Tresness, C., Russinovich, M. and Bianchini, R., (2020) Serverless in the Wild: Characterizing and optimizing the serverless workload at a large cloud provider. arXiv.

[5] Singh, P., Gupta, P. and Jyoti, K., (2019) TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud. Cluster Computing, [online] 222, pp.619–633. Available at: https://doi.org/10.1007/s10586-018-2868-6.

Appendix C: Ethics Forms

Appendix D: Resource Required

Hardware requirements for the study and to conduct work towards the in-depth research study are illustrated as below (this List is not a final one and may require changes when implementation starts)

1) Hardware Requirements

Some of the deep learning algorithms require GPUs to process the data sets and hence in certain conditions GPU access might also be required to run the models. However, the following are the hardware requirements that would be required for the methods to be run with the bare minimum configuration as below:

a) Processor: 10th Generation Intel® Core™ i7-10750H
b) Laptop OS: Windows 10
c) RAM: 16GB and above
d) Video card: NVIDIA® GeForce®

Appendix E: Risk and Contingency Plan

Appendix E 1: Risk and contingency plan

Serial No	Risk Seen	Plan towards Contingency
1	Unable to meet deadlines due to other factors affecting the study/research	Connect with Student Mentor and reach out for suggestions and act accordingly
2	Not able to create and accomplish the research work of building Models	Reach out to the Batch Supervisor and suggest corrective actions
3	The data set is not appropriate for the research to be completed with desired results	More investigation on appropriate data sets would be required and suitable data sets selected accordingly
4	Local computer resources not enough to run models	Promptly ask for GPU access by reaching out to student mentor