A PROJECT REPORT ON

IMPROVING AMAZON EC2 SPOT INSTANCES PRICE PREDICTION USING MACHINE LEARNING ALGORITHM

Submitted in partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE & ENGINEERING

Submitted By

G. CHISHMA	**20MQ1A0561**
P. D V K PADMAVATHI	**20MQ1A0528**
K. REETHIKA	**20MQ1A0569**
A. CHANDRA SEKHAR	**20MQ1A0534**

Under the Esteemed Supervision of

M. Prasanthi M.Tech.,

Assistant Professor

... Empowering Minds

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRI VASAVI INSTITUTE OFENGINEERING &TECHNOLOGY

Accredited by NAAC with 'A' Grade & NBA (CSE, ECE & ME)

Approved by AICTE-New Delhi & Affiliated to JNTUK, Kakinada

An ISO 9001:2015 Certified Institute

Nandamuru, Pedana(M)-521369, Krishna Dist., Andhra Pradesh.

2020-2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRI VASAVI INSTITUTE OFENGINEERING &TECHNOLOGY

Accredited by NAAC with 'A' Grade & NBA (CSE, ECE & ME)

Approved by AICTE-New Delhi & Affiliated to JNTUK Kakinada.

An ISO 9001:2015 Certified Institute

Nandamuru, Pedana(M)-521369, Krishna Dist., Andhra Pradesh.

2020-2024

... Empowering Minds

CERTIFICATE

This is to certify that the project report entitled **"IMPROVING AMAZON EC2 SPOT INSTANCES PRICE PREDICTION USING MACHINE LEARNING ALGORITHM"** is a bonafide work carried out by **G.CHISHMA (20MQ1A0561), P.D.V.K.PADMAVATHI (20MQ1A0528) , K.REETHIKA (20MQ1A0569) , A. CHANDRA SEKHAR (20MQ1A0534)** . Under the guidance and supervision in partial fulfillment of the requirements for the award of degree of **B. Tech** in Computer Science & Engineering from **JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, KAKINADA.** The results embodied in this Project report have not been submitted to any other University or Institute for the award of any degree or diploma.

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
M. Prasanthi	Dr. G. Syam Prasad
M. Tech.,	B. Tech., M. Tech., Ph. D
Assistant Professor	Professor & HOD

EXTERNAL EXAMINER

DECLARATION

We certify that,

- The work contained in this report is original and has been done by us under the guidance of our supervisor.
- The work has not been submitted to any other institute for any degree or diploma. We have followed the guidelines provided by the institute in preparing the report.
- We confirm to the norms and guidelines given in the Ethical code of Conduct of the institute.
- Whenever we have used materials (data, theoretical analysis, figures and text) from the sources, we have given due credit to them by references.
- Further, we have taken permission from the copy right owners of the sources whenever necessary.

Signature(s) of the Student(s):

G. CHISHMA	**20MQ1A0561**
P. D V K PADMAVATHI	**20MQ1A0528**
K. REETHIKA	**20MQ1A0569**
A. CHANDRA SEKHAR	**20MQ1A0534**

ACKNOWLEDGEMENT

We take great pleasure to express our deep sense of gratitude to our project guide **M. Prasanthi M. Tech., Assistant professor**, for his valuable guidance during the course of our project work.

We would like to thank **Dr. G. Syam Prasad B.Tech., M. Tech., Ph. D**, Head of the Department of ComputerScience & Engineering for his encouragement.

We would like to express our heart-felt thanks to **Dr. B. R. S. Reddy M. Tech, Ph. D** Principal, Sri Vasavi Institute of Engineering & Technology, Nandamuru for providing all the facilities for our project.

Our utmost thanks to all the faculty members and Non-Teaching Staff of the Department of Computer Science & Engineering for their support throughout our project work.

Our Family Members and Friends receive our deepest gratitude and love for their support throughout our academic year.

Project Associates:

G. CHISHMA	**20MQ1A0561**
P. D V K PADMAVATHI	**20MQ1A0528**
K. REETHIKA	**20MQ1A0569**
A. CHANDRA SEKHAR	**20MQ1A0534**

ABSTRACT

Spot instances were introduced by Amazon EC2 in December 2009 to sell its spare capacity through auction based market mechanism. Despite its extremely low prices, cloud spot market has low utilization. Spot pricing being dynamic, spot instances are prone to outof bid failure. Bidding complexity is another reason why users today still fear using spot instances. This work aims to present Regression Random Forests (RRFs) model to predict one-week-ahead and one-day-ahead spot prices. The prediction would assist cloud users to plan in advance when to acquire spot instances, estimate execution costs, and also assist them in bid decision making to minimize execution costs and out-of-bid failure probability. Simulations with 12 months real Amazon EC2 spot history traces to forecast future spot prices show the effectiveness of the proposed technique. Comparison of RRFs based spot price forecasts with existing non-parametric machine learning models reveal that RRFs based forecast accuracy outperforms other models. We measure predictive accuracy using MAPE, MCPE, OOB Error and speed. Evaluation results show that MAPE <= 10% for 66 to 92% and MCPE <= 15% for 35 to 81% of one-day-ahead predictions with prediction time less than one second. MAPE<= 15% for 71 to 96% of one-week-ahead predictions. Index Terms— Amazon EC2, Compute instances, One-day-ahead prediction, One-week-ahead prediction, Regression Random Forests, Spot instances, Spot price prediction.

LIST OF CONTENT

LIST OF FIGURES

I

LIST OF ABBREVIATIONS

ABBREVIATION	EXPLANATION
DL	Deep Learning
ML	Machine Learning
RRF	Regression Random Forest
EC2	Elastic Cloud Computing
MAPE	Mean Absolute Percentage Error
MCPE	Mean Consequential Percentage Error
OOB Error	Out Of Bag Error

LIST OF TABLES

1.INTRODUCTION

1. INTRODUCTION:

1.HE on-demand scalability characteristic of cloud computing forces cloud service providers to overestimate their resources to meet peak load demand of its customers which happens at different time periods and may not overlap. Due to over-estimation, a large number of cloud resources are idle during off peak hours. Cloud providers also face the problem of allocating resources, keeping in view user‟s different job requirements and data center capacity. Different types of users, multiple types of requirements further alleviate the resource allocation problem. Also, demand for cloud resources fluctuate due to today‟s usage based pricing plans. In order to manage these demand fluctuations more flexible pricing plans are required to sell resources according to real time market demand. Spot pricing was introduced by Amazon EC2 in December 2009 to minimize operational cost, combat under utilization of its resources and make more profit. Similar to on-demand instances, spot instances offer several instance types comprising different combinations of CPU, memory, storage and networking capacity. Amazon Web Service (AWS) is not the only participant in the spot instance realm. Google Compute Engine launched its preemptible Virtual Machines on September 8, 2015 designed for such type of workloads that can be delayed and are fault tolerant at the same time. Users can bid for spot instances (SIs) where prices are charged at lowest bid price, whereas, pricing on Google Preemptible VMs is fixed at per hour rate. The distinguishing feature of Amazon Elastic Compute Cloud (EC2) spot instance is its dynamic pricing. From customer's perspective, spot instances offer prospects of low cost utility computing at a risk of out-of-bid failure at any time by Amazon EC2. Spot instance reliability depends on the market price and user's maximum bid (limited by their hourly budget). Spot prices vary dynamically with real-time based on demand (user's bid) and supply (resource availability) for spot instance capacity in the data centers across the globe. User's bids for spot instances and control the balance of reliability versus monetary cost. The price for spot instances sometimes can be as low as one eighth of the price of on-demand instances. On the other hand, it is also not uncommon that spot prices surpass on-demand prices in cloud data centers. When the demand is low, spot prices are low because less numbers of users are bidding for the same instance.

Therefore, a bidder‟s probability of incurring less monetary cost is higher. On the other hand, when the demand is high, users are willing to pay high to get access and hence spot prices increase.

Index Terms— Amazon EC2, Compute instances, One-day-ahead prediction, One-week-ahead prediction, Regression Random Forests, Spot instances, Spot price prediction

2.OBJECTIVE OF THE PROJECT

2.1. Existing System:

Existing System:

Spot pricing in particular is a pricing model targeted for divisible computing jobs that can shift the time of processing to when the computing resources are available at low cost The primary requirement is that the applications must be time flexible, do not have a steep completion deadline and should be interrupt tolerant. Spot instances are also required for executing certain sudden tasks which do not need reserved instances.

Disadvantages:

The ability to predict spot price lends itself to a variety execution and Dynamic pricing model is not followed by any other cloud service provider. However, we do not see any issues in using our approach for forecasting spot prices of service providers who follow dynamic pricing policy accuracy is very less.

2.2 Proposed system:

The objective of this work is to present and evaluate a predictive model for spot price prediction that can predict future prices with increased accuracy and speed, minimize forecasting errors and predict spot prices sufficiently far in advance to assist cloud spot users in bid decision making process with increased reliability. We compare prediction accuracy of the state of the art non-parametric supervised machine learning algorithms with Regression Random Forests (RRFs) model.

Advantages:

1. The purpose of proposed system is An analysis of the length of time epoch durations when spot price is less than on-demand price to raise users confidence level in opting for spot instances.

2.Spot price forecasting. We resort to machine learning based ensemble method namely RRFs for one-weekahead and one-day-ahead spot price prediction. The approach focuses on both prediction accuracy and speed is high.

2.3 Modules Description

Pandas: pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Numpy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

MatPlotLib:matplotlibPyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits

Scikit-learn: Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and ScitPy.

3.LITERATURE SURVEY

Layer Recurrent Neural Network Based Power System Load Forecasting**.**This paper presents a straight forward application of Layer Recurrent Neural Network (LRNN) to predict the load of a large distribution network. Short term load forecasting provides important information about the system's load pattern, which is a premier requirement in planning periodical operations and facility expansion. Approximation of data patterns for forecasting is not an easy task to perform. In past, various approaches have been applied for forecasting. In this work application of LRNN is explored. The results of proposed architecture are compared with other conventional topologies of neural networks on the basis of Root Mean Square of Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). It is observed that the results obtained from LRNN are comparatively more significant. Keywords: artificial neural network, electricity load forecasting, layer recurrent neural network, linear regression, short term load forecasting

EMOTION RECOGNITION IN HUMANS AND MACHINE

This paper presents a straight forward application of Layer Recurrent Neural Network (LRNN) to predict the load of a large distribution network. Short term load forecasting provides important information about the system's load pattern, which is a premier requirement in planning periodical operations and facility expansion. Approximation of data patterns for forecasting is not an easy task to perform. In past, various approaches have been applied for forecasting. In this work application of LRNN is explored. The results of proposed architecture are compared with other conventional topologies of neural networks on the basis of Root Mean Square of Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). It is observed that the results obtained from LRNN are comparatively more significant.

Keywords: artificial neural network, electricity load forecasting, layer recurrent neural network, linear regression, short term load forecasting

A Double Auction Mechanism to Bridge Users' Task Requirements and Providers' Resources in Two-Sided Cloud Market.Double auction-based pricing model is an efficient pricing model to balance users' and providers' benefits. Existing double auction mechanisms usually require both users and providers to bid with the unit price and the number of VMs. However, in practice users seldomknow the exact number of VMs that meets their task requirements, which leads to users' task requirements inconsistent with providers' resource. In this paper, we propose a truthful double

auction mechanism, including a matching process as well as a pricing and VM allocation scheme, to bridge users' task requirements and providers' resources in two-sided cloud markets. In the matching process, we design a cost-aware resource algorithm based on Lyapunov optimization techniques to precisely obtain the number of VMs that meets users' task requirements. In the pricing and VM allocation scheme, we apply the idea of second-price auction to determine the final price and the number of provisioned VMs in the double auction. We theoretically prove our proposed mechanism is individual-rational, truthful and budget-balanced, and analyze the optimality of proposed algorithm. Through simulation experiments, the results show that the individual profits achieved by our algorithm are 12.35 and 11.02 percent larger than that of scaleout and greedy scale-up algorithms respectively for 90 percent of users, and the social welfare of our mechanism is only 7.01 percent smaller than that of the optimum mechanism in the worst case.

Index Terms—Double auction mechanism, two-sided cloud markets, VM trading, Lyapunov optimization.

Dynamic and Heterogeneous Ensembles for Time Series Forecasting

This paper addresses the issue of learning time series forecasting models in changing environments by leveraging the predictive power of ensemble methods. Concept drift adaptation is performed in an active manner, by dynamically combining base learners according to their recent performance using a non-linear function. Diversity in the ensembles is encouraged with several strategies that include heterogeneity among learners, sampling techniques and computation of summary statistics as extra predictors. Heterogeneity is used with the goal of better coping with different dynamic regimes of the time series. The driving hypotheses of this work are that (i) heterogeneous ensembles should better fit different dynamic regimes and (ii) dynamic aggregation should allow for fast detection and adaptation to regime changes. We extend some strategies typically used in classification tasks to time series forecasting. The proposed methods are validated using Monte Carlo simulations on 16 realworld univariate time series with numerical outcome as well as an artificial series with clear regime shifts. The results provide strong empirical evidence for our hypotheses. To encourage reproducibility the proposed

method is publicly available as a software package. Keywords-dynamic ensembles; time series forecasting.

Bidding Strategies for Amazon EC2 Spot Instances A Comprehensive Review

Spot instance bid price is among one of the major issues in cloud computing. Bid price is aimed at complete execution of the work flow on spot instances while considering several job requirements such as hardware, latency, deadline and budget constraints. Several state-of-the-art spot bidding strategies have been proposed in the literature for executing jobs on Amazon EC2 spot instances. This paper presents different spot bidding strategies proposed by the authors. It highlights the objectives of each and provides the suitability of each of the proposed bidding strategies based on the type of application, its fault tolerance, job requirements and other constraints.

Keywords— Amazon EC2, Bidding Strategies, Spot Instances, Cost Minimization

4.SYSTEM ANALYSIS

4.1. SCOPE

Amazon Web Services (AWS) provides virtual computing environments via their EC2 service. You can launch instances with your favourite operating system, select pre-configured instance images or create your own.

Why this is revelant to data sciensits is because generally to run deep learning models you need a machine with a good GPU.

EC2 can be configured with a P2/P3 instance and can be configured with up to 8 or 16 GPUs respectively!

However, you can request Spot Instance Pricing.

Which basically charges you for the spot price that is in effect for the duration of your instance running time.

They are adjusted based on long-term trends in supply and demand for Spot instance capacity.

Spot instances can be discounted at up to 90% off compared to On-Demand pricing.

4.2 PURPOSE OF PROJECT

Propose a new metric called Mean Consequential Percentage Error (MCPE) adapted from Mean Consequential Error (used for classification problems) in order to measure prediction accuracy. 4 Spot price forecasting. We resort to machine learning based ensemble method namely RRFs for one-weekahead and one-day-ahead spot price prediction. The approach focuses on both prediction accuracy and speed.

4.3 KEYWORDS

Index Terms— Amazon EC2, Compute instances, One-day-ahead prediction, One-week-ahead prediction, Regression Random Forests, Spot instances, Spot price prediction

4.4 FEASIBILITY REPORT

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system

is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Economical Feasibility
- Operation Feasibility

4.5 TECHINICAL FEASIBILITY:

In the feasibility study first step is that the organization or company has to decide that what technologies are suitable to develop by considering existing system.

Here in this application used the technologies like Visual Studio 2008 and SQL Server 2005. These are free software that would be downloaded from web.

Visual Studio 2008 –it is tool or technology.

4.6 OPERATIONAL FEASIBILITY:

Not only must an application make economic and technical sense, it must also make operational sense.

Issues to consider when determining the operational feasibility of a project.

Operations Issues	**Support Issues**
What tools are needed to support operations? What skills will operators need to be trained in? What processes need to be created and/or updated? What documentation does operations need?	• What documentation will users be given? • • What training will users be given? • How will change requests be managed?

Very often you will need to improve the existing operations, maintenance, and support infrastructure to support the operation of the new application that you intend to develop. To determine what the impact will be you will need to understand both the current operations and support infrastructure of your organization and the operations and support characteristics of your new application.

To operate this application this system that the user no needs to require any technical knowledge that we are used to develop this project is. Asp.net, C#.net. that the application providing rich user interface by user can do the operation in flexible manner.

4.4 ECONOMIC FEASIBILITY:

It refers to the benefits or Outcomes we are deriving from the product as compared to the total cost we are spending for developing the product. If the benefits are more or less the same as the older system, then it is not feasible to develop the product.

In the present system, the development of new product greatly enhances the accuracy of the system and cuts short the delay in the processing this application. The errors can be greatly reduced and at the same time providing a great level of security. Here we don't need any additional equipment except memory of required capacity.No need for spending money on client for maintenance because the database used is web enabled database.

5.SYSTEM REQUIREMENT SPECIFICATIONS

SYSTEM REQUIREMENT SPECIFICATIONS

A **Software Requirements Specification** (**SRS**) – a requirements specification for a software system – is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development life cycle domain, typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- **Business requirements** describe in business terms *what* must be delivered or accomplished to provide value.
- **Product requirements** describe properties of a system or product (which could be one of

Several ways to accomplish a set of business requirements.)

- **Process requirements** describe activities performed by the developing organization. For instance, process requirements could specify specific methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement); a requirement that the product be maintainable (a Product requirement) often is addressed by imposing requirements to follow particular development styles

5.1 Functional Requirement

5.1.1 Software Requirements

OS : Windows

Python IDE : python 3.x and above
Jupyter Notebook,
Anaconda 3.5

Setup tools and pip to be installed for 3.6.x and above

5.1.2 Hardware Requirements

RAM	:	4GB and Higher
Processor	:	Intel i3 and above
Hard Disk	:	500GB: Minimum

5.2 NON FUNCTIONAL REQUIREMENTS

- Secure access of confidential data (user's details). SSL can be used.
- 24 X 7 availability.
- Better component design to get better performance at peak time
- Flexible service based architecture will be highly desirable for future extension

5.3 SDLC Methodologies

SDLC MODEL

The Software Development Lifecycle(SDLC) for small to medium database application development efforts.

This project uses iterative development lifecycle, where components of the application are developed through a series of tight iteration. The first iteration focus on very basic functionality, with subsequent iterations adding new functionality to the previous work and or correcting errors identified for the components in production.

The six stages of the SDLC are designed to build on one another, taking outputs from the previous stage, adding additional effort, and producing results that leverage the previous effort and are directly traceable to the previous stages. During each stage, additional information is gathered or developed, combined with the inputs, and used to produce the stage deliverables. It is important to not that the additional information is restricted in scope, new ideas that would take the project in directions not anticipated by the initial set of high-level requirements or features that are out-of-scope are preserved for later consideration.

Too many software development efforts go awry when development team and customer personnel get caught up in the possibilities of automation. Instead of focusing on high priority features, the team can become mired in a sea of nice to have features that are not essential to solve the problem, but in themselves are highly attractive. This is the root cause of large percentage of failed and or abandoned development efforts and is the primary reason the development team utilizes the iterative model.

Roles and Responsibilities of PDR AND PER

The iterative lifecycle specifies two critical roles that act together to clearly communicate project issues and concepts between the end-user community and the development team.

Primary End-user Representative (PER)

The PER is a person who acts as the primary point of contact and principal approver for the end-user community. The PER is also responsible for ensuring that appropriate subject matter experts conduct end-user reviews in a timely manner.

PER-PDR Relationship: The PER and PDR are the brain trust for the development effort. The PER has the skills and domain knowledge necessary to understand the issues associated with the business processes to the supported by the application and has a close working relationship with the other members of the end-user community. The PDR has the same advantages regarding the application development process and the other members of the development team together, they act as the concentration points for knowledge about the application to be developed.

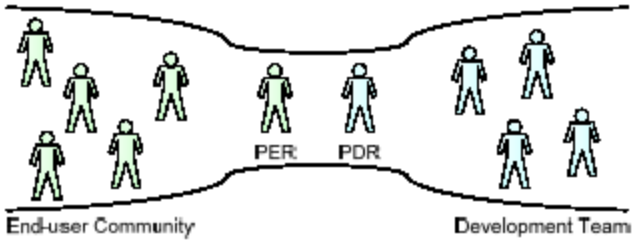

Fig:5.3.2 PER-PDR Relational

The objective of this approach is to create the close relationship that is characteristic of a software project with one developer and one end-user in essence, this approach the "pair programming" concept from Agile methodologies and extends it to the end-user community. While it is difficult to create close relationships between the diverse members of an end-user community and a software development team, it is much simpler to create a close relationship between the lead representatives for each group.

When multiple end-users are placed into relationship with multiple members of a development team, communication between the two groups degrades as the number of participants grows. In this model, members of end-user community may communicate with members of the development team as needed, but it is the responsibility of all participants to keep the PER and PDR apprised of the communications for example, this allows the PER and PDR to resolve conflicts that arise when two different end-users communicate different requirements for the same application feature to different members of the development team.

5.4 System architecture

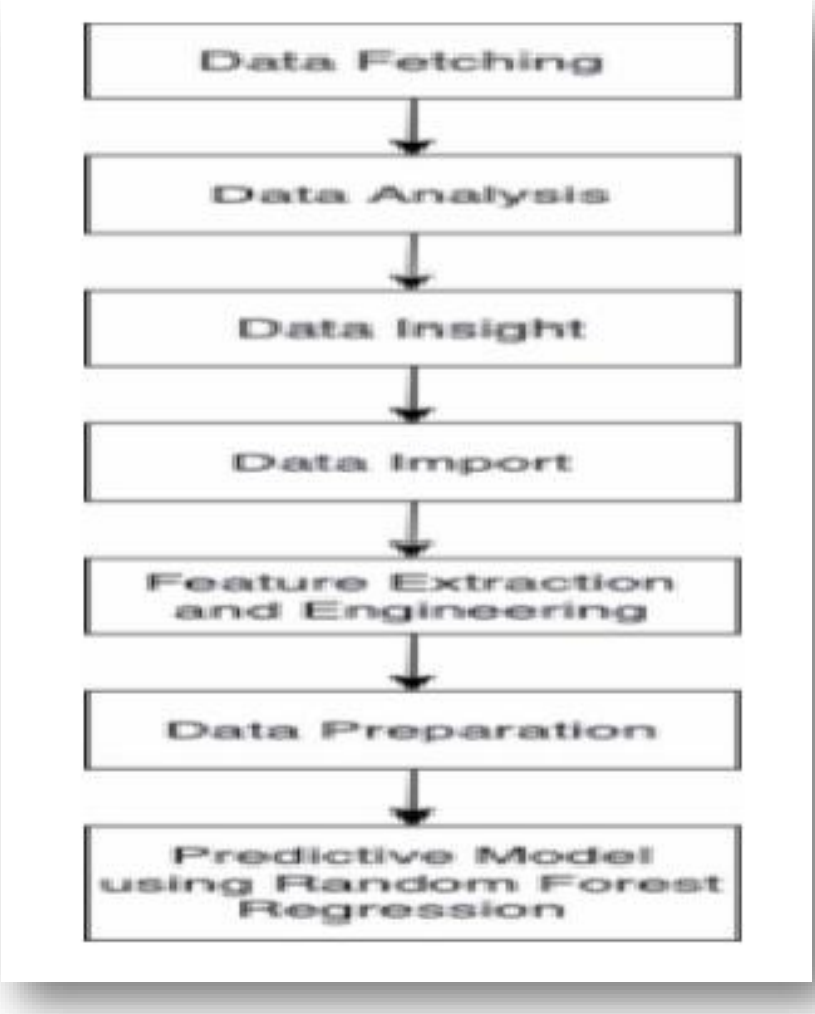

Fig:5.4.1System Architecture

6.SYSTEM DESIGN

6.1 UML DIAGRAMS

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

- actors
- business processes
- (logical) components
- activities
- programming language statements
- database schemas, and
- Reusable software components.

UML combines best techniques from data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modeling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modeling language. UML aims to be a standard modeling language which can model concurrent and distributed systems.

Use Case Diagram:

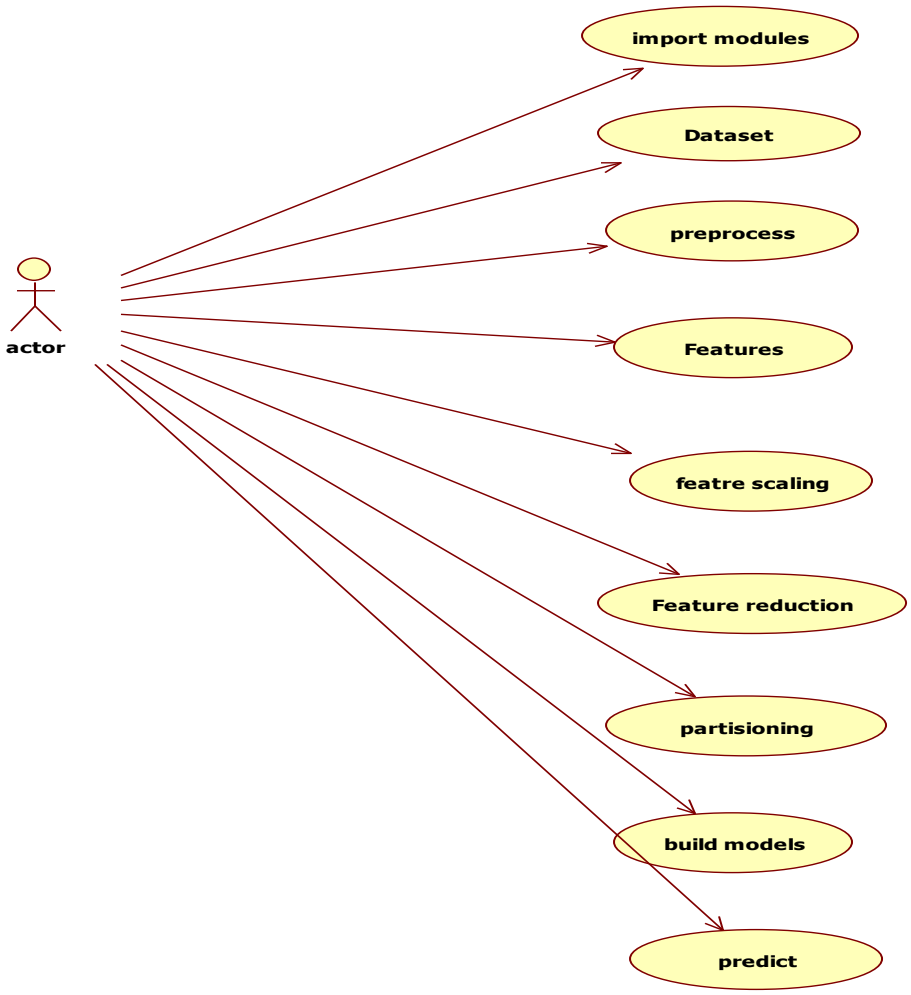

Fig:6.1.1 Use case diagram

Sequence diagram:

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

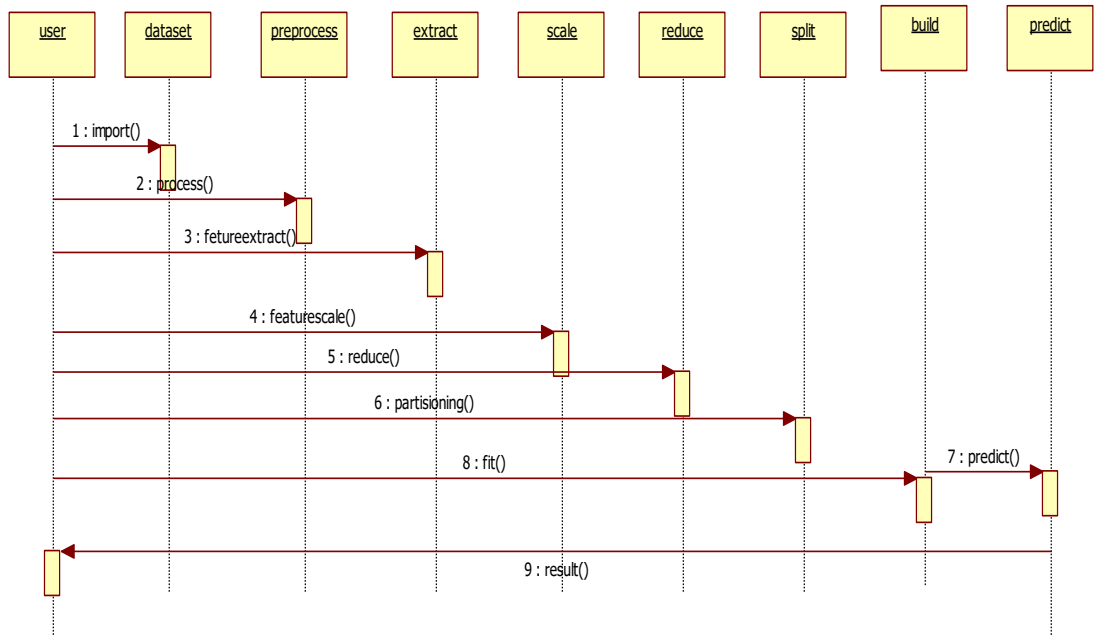

Fig:6.1.2 Sequence diagram

Collaboration Diagram:

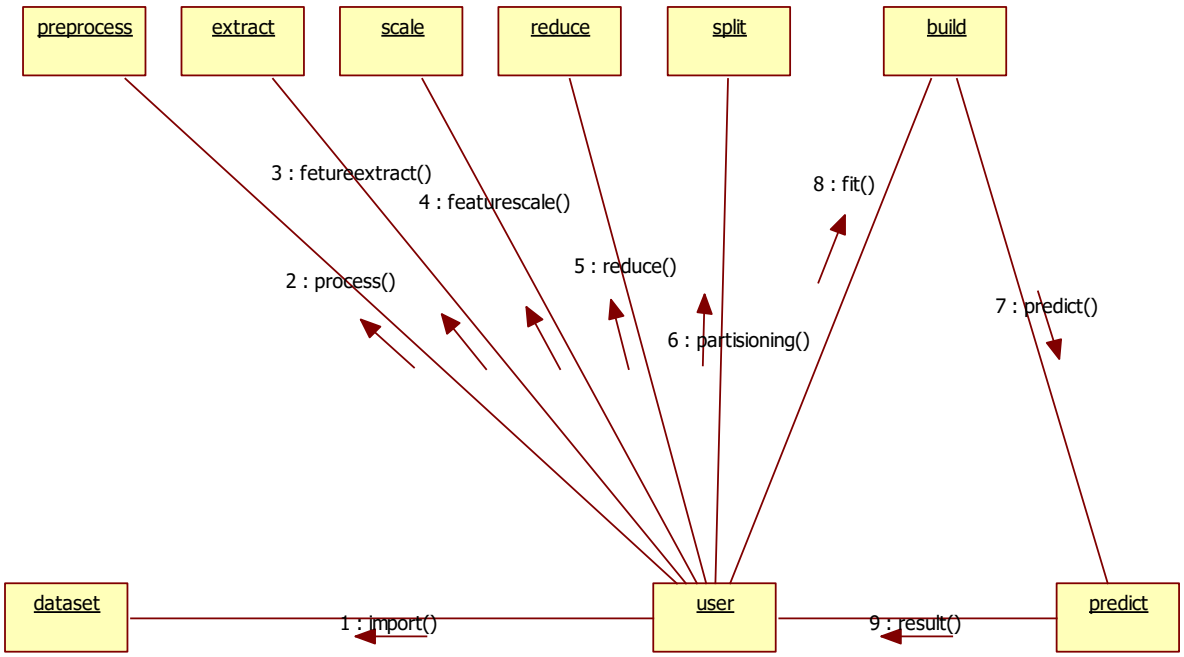

Fig:6.1.3 Collaboration diagram

Class Diagram:

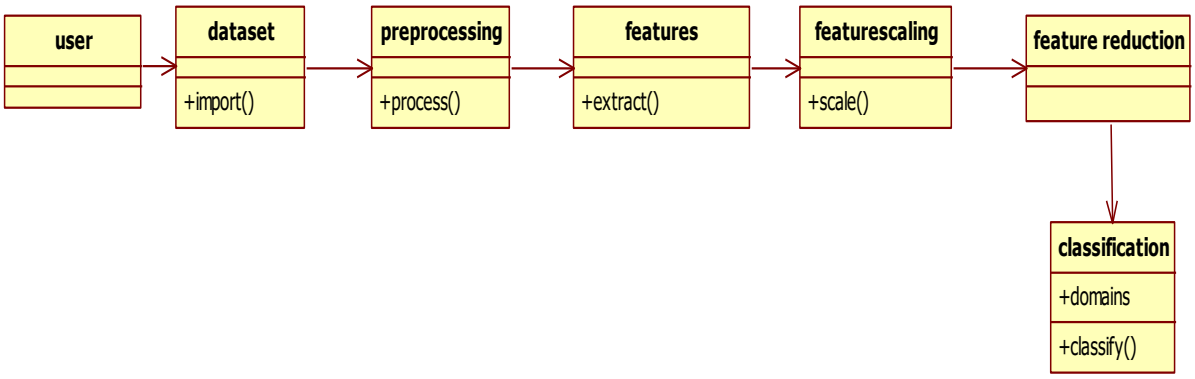

Fig:6.1.4 Class diagram

Component Diagram

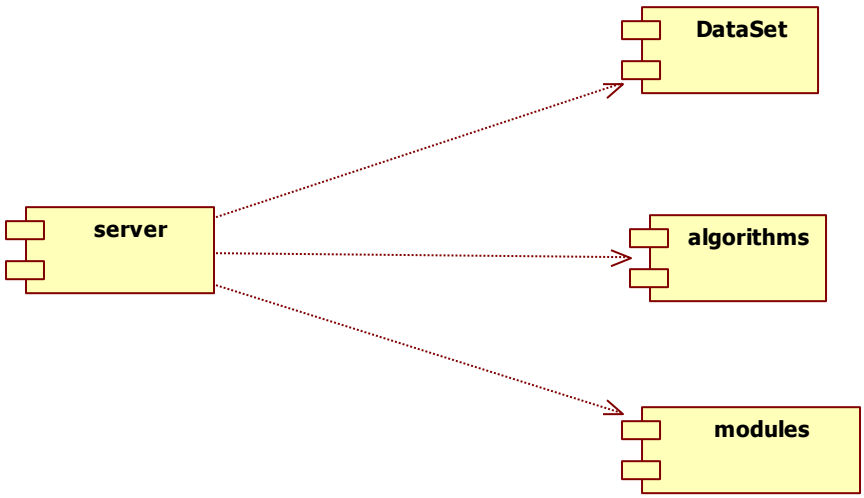

Fig:6.1.5 Component diagram

State Diagram:

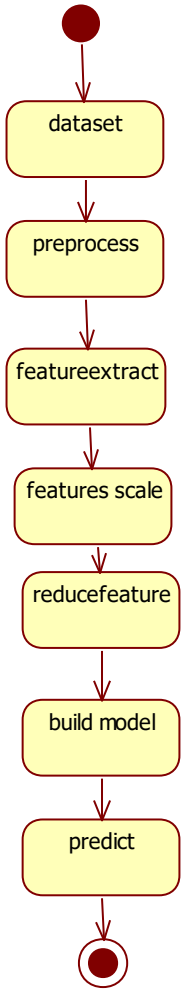

Fig:6.1.6 State diagram

Activity Diagram

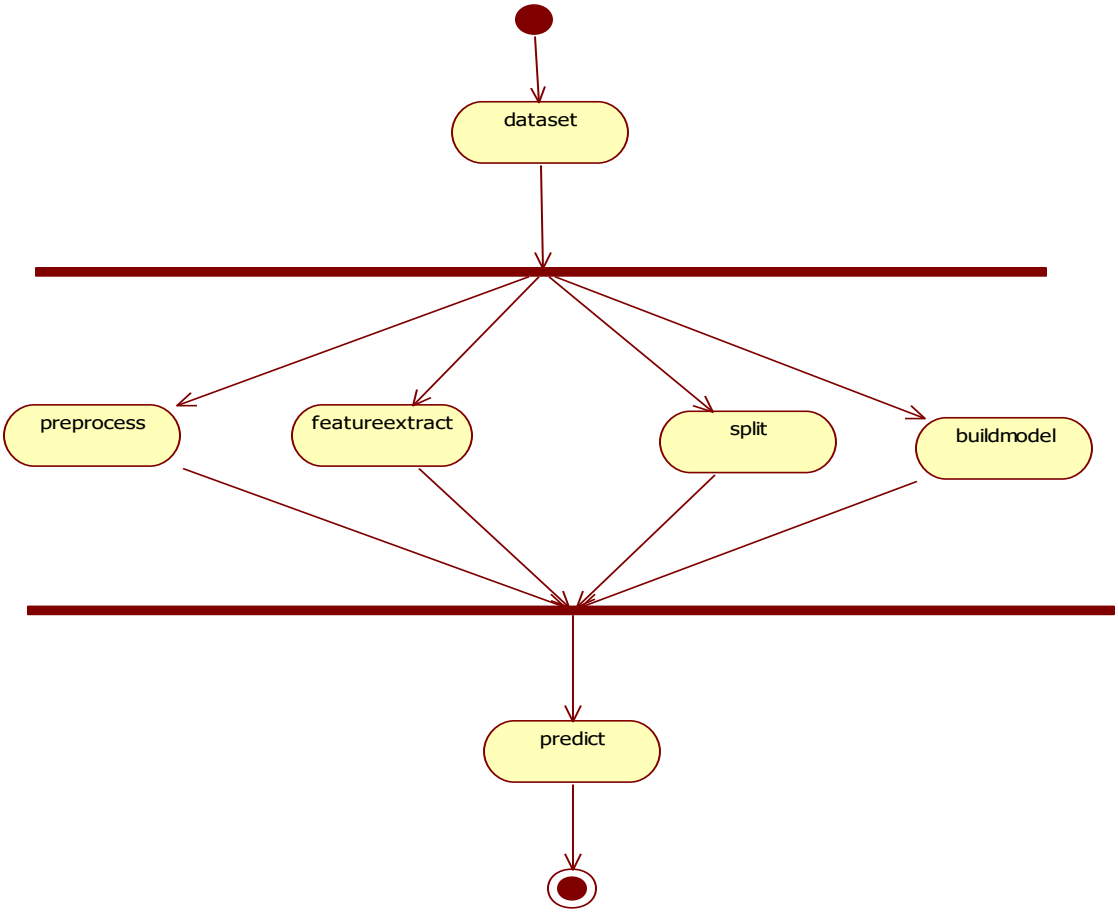

Fig:6.1.7 Activity diagram

7.TECHNOLOGY DESCRIPTION AND IMPLEMENTATION

Introduction To Python Framework : Introduction to Django This book is about Django, a Web development framework that saves you time and makes Web development a joy. Using Django, you can build and maintain high-quality Web applications with minimal fuss. At its best, Web development is an exciting, creative act; at its worst, it can be a repetitive, frustrating nuisance. Django lets you focus on the fun stuff — the crux of your Web application — while easing the pain of the repetitive bits. In doing so, it provides high-level abstractions of common Web development patterns, shortcuts for frequent programming tasks, and clear conventions for how to solve problems. At the same time, Django tries to stay out of your way, letting you work outside the scope of the framework as needed. The goal of this book is to make you a Django expert. The focus is twofold. First, we explain, in depth, what Django does and how to build Web applications with it. Second, we discuss higher-level concepts where appropriate, answering the question "How can I apply these tools effectively in my own projects?" By reading this book, you'll learn the skills needed to develop powerful Web sites quickly, with code that is clean and easy to maintain. What Is a Web Framework?

Django is a prominent member of a new generation of Web frameworks. So what exactly does that term mean? To answer that question, let's consider the design of a Web application written using the Common Gateway Interface (CGI) standard, a popular way to write Web applications circa 1998. In those days, when you wrote a CGI application, you did everything yourself — the equivalent of baking a cake from scratch. For example, here's a simple CGI script, written in Python, that displays the ten most recently published books from a database:

```
import MySQLdb

print "Content-Type: text/html"
print
print "<html><head><title>Books</title></head>"
print "<body>"
print "<h1>Books</h1>"
print "<ul>"

connection = MySQLdb.connect(user='me', passwd='letmein', db='my_db')
cursor = connection.cursor()
cursor.execute("SELECT name FROM books ORDER BY pub_date DESC LIMIT 10")
for row in cursor.fetchall():
    print "<li>%s</li>" % row[0]

print "</ul>"
print "</body></html>"

connection.close()
```

This code is straightforward. First, it prints a "Content-Type" line, followed by a blank line, as required by CGI. It prints some introductory HTML, connects to a database and executes a query that retrieves the latest ten books. Looping over those books, it generates an HTML unordered list. Finally, it prints the closing HTML and closes the database connection.

With a one-off dynamic page such as this one, the write-it-from-scratch approach isn't necessarily bad. For one thing, this code is simple to comprehend — even a novice developer can read these 16 lines of Python and understand all it does, from start to finish. There's nothing else to learn; no other code to read. It's also simple to deploy: just save this code in a file called latestbooks.cgi, upload that file to a Web server, and visit that page with a browser. But as a Web application grows beyond the trivial, this approach breaks down, and you face a number of problems:

Should a developer really have to worry about printing the "Content-Type" line and remembering to close the database connection? This sort of boilerplate reduces programmer productivity and introduces opportunities for mistakes. These setup- and teardown-related tasks would best be handled by some common infrastructure.

- ✓ What happens when this code is reused in multiple environments, each with a separate database and password? At this point, some environment-specific configuration becomes essential.
- ✓ What happens when a Web designer who has no experience coding Python wishes to redesign the page? Ideally, the logic of the page — the retrieval of books from the database — would be separate from the HTML display of the page, so that a designer could edit the latter without affecting the former.

- ✓ These problems are precisely what a Web framework intends to solve. A Web framework provides a programming infrastructure for your applications, so that you can focus on writing clean, maintainable code without having to reinvent the wheel. In a nutshell, that's what Django does.

Python

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are reusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.

You can use *Microsoft Notepad*, *Microsoft WordPad*, *Microsoft Word*, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived(e.g., compiled)

Python

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don't know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to "big name" languages. Hopefully I can explain it for you.

Python concepts

If your not interested in the the hows and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it's a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

- Python's features include –**Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Dynamic vs Static

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of "thing" each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a "float" type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn't require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn't matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it's more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken

using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as **read-only** lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
Dictionaries are enclosed by curly braces ({}) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive. The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter. Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

Python libraries

1. Requests. The most famous http library written by kenneth reitz. It's a must have for every python developer.
2. Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won't use any other.
3. wxPython. A GUI toolkit for python. I have primarily used it in place of tkinter. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.
6. BeautifulSoup. I know it's slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.
8. NumPy. How can we leave this very important library? It provides some advance math functionalities to python.
9. SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
11. Pygame. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made
13. PyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI's for my python scripts.
14. PyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.
15. Scapy. A packet sniffer and analyzer for python made in python.
16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. Nltk. Natural Language Toolkit – I realize most people won't be using this one, but its generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. IPython. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

Numpy

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

• High quality plotting library.

Python class and objects

These are the building blocks of OOP. Class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently "the big thing" in most programming languages. Hence, there are several chapters dedicated to OOP later in the book. The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something. Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects. When I first learned C and C++, I

did great; functions just made sense for me. Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn. However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered. Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing. One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want. As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the codeObjects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: Class Name. Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components. As we've seen, an instance of a class inherits the attributes of that class. However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses. The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc. Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python's object model and the object interfaces are more consistent to Python's built-in objects, hence overloading is easier to learn and use. User-made classes can override nearly all of Python's built-in operation methods

Exceptions

I've talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program's flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. trying to open a file that doesn't exist, or when the program reaches a marker, such as the completion of a loop. Exceptions, by definition, don't occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python. Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

.• Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing datatypes without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files. This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions. Your program is usually short enough to not be hurt too much if an exception occurs. Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem. However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception. It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing. Because exceptions aren't supposed to happen very often, they aren't processed until they occur. Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions. However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time. Proper use of exceptions can help the performance of your program. The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing. Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won't spend too much time talking about this, but Python does allow for a programmer to create his own exceptions. You probably won't have to do this very often but it's nice to have the option when necessary. However, before making your own exceptions, make sure there isn't one of the built-in exceptions that will work for you. They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free. Making your own exceptions involves object-oriented programming, which will be covered in the next chapter. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class. To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a *module*; definitions from a module can be *imported* into other modules or into the *main* module.

Testing code

As indicated above, code is usually developed in a file using an editor.To test the code, import it into a Python session and try to run it. Usually there is an error, so you go back to the file, make a correction, and test again. This process is repeated until you are satisfied that the code works.

The entire process is known as the development cycle. There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or cannot return one or more values.

There are three types of functions in python:

Help (), min (), print ().

Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.

Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- **global names** of a module
- **local names** in a function or method invocation
- **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language. What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs The Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX : Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1.URL routing

2.HTML, XML, JSON, and other output format templating

3.Database manipulation

4.Security against Cross-site request forgery (CSRF) and other attacks

5.Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.

- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework.
- Django vs Flash vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- Web frameworks is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist

1.Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.

2.Work through a detailed tutorial found within the resources links on the framework's page.

8.SOURCE CODE

8. SOURCE CODE

```
import numpy as np
import pandas as pd
from subprocess import check_output

central_df = pd.read_csv("F:/NIT/2020-New_projects/Amazon-ec2-price_prediction/ca-central-1.csv")
central_df.columns = ['datetime','os','instance_type','price','region']
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
central_df['price'] = labelencoder.fit_transform(central_df['price'])
central_df.head()
X1 = central_df.drop(['price','datetime'],axis=1)
central_df2 = pd.get_dummies(X1)
X1 = central_df2.values
y1 = central_df['price'].values
from sklearn.model_selection import train_test_split
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2,random_state=42
from sklearn.linear_model import SGDRegressor
clf1 = SGDRegressor()
clf1.fit(X1_train,y1_train)
y1_rbf = clf1.predict(X1_test)
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor(max_depth=2, random_state=0)
```

9.TESTING AND TEST CASES

INTRODUCTION TO TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The increasing visibility of software as a system element and attendant costs associated with a software failure are motivating factors for we planned, through testing. Testing is the process of executing a program with the intent of finding an error. The design of tests for software and other engineered products can be as challenging as the initial design of the product itself.

There of basically **two types of testing** approaches.

One is ***Black-Box testing*** – the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operated.

The other is ***White-Box testing*** – knowing the internal workings of the product ,tests can be conducted to ensure that the internal operation of the product performs according to specifications and all internal components have been adequately exercised.

White box and Black box testing methods have been used to test this package. The entire loop constructs have been tested for their boundary and intermediate conditions. The test data was designed with a view to check for all the conditions and logical decisions. Error handling has been taken care of by the use of exception handlers.

TESTING STRATEGIES:

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.

Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is

accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.Testing is the only way to assure the quality of software and it is an umbrella activity rather than a separate phase. This is an activity to be preformed in parallel with the software effort and one that consists of its own phases of analysis, design, implementation, execution and maintenance.

UNIT TESTING:

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level. Here the module will be treated as a black box, which will take some input and generate output. Outputs for a given set of input combination are pre-calculated and are generated by the module.

SYSTEM TESTING:

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other. This testing methodology helps in making sure that all modules which are running perfectly when checked individually are also running in cohesion with other modules. For this testing we create test cases to check all modules once and then generated test combinations of test paths throughout the system to make sure that no path is making its way into chaos.

INTEGRATED TESTING

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests. To uncover errors that are associated with interfacing the objective is to make unit test modules and built a program structure that has been detected by design. In a non - incremental integration all the modules are combined in advance and the program is tested as a whole. Here errors will appear in an end less loop function. In incremental testing the program is constructed and tested in small segments where the errors.

Different incremental integration strategies are top – down integration, bottom – up integration, regression testing.

TOP-DOWN INTEGRATION TEST

Modules are integrated by moving downwards through the control hierarchy beginning with main program. The subordinate modules are incorporated into structure in either a breadth first manner or depth first manner.

This process is done in five steps:

- Main control module is used as a test driver and steps are substituted or all modules directly to main program.
- Depending on the integration approach selected subordinate is replaced at a time with actual modules.
- Tests are conducted.
- On completion of each set of tests another stub is replaced with the real module
- Regression testing may be conducted to ensure trha5t new errors have not been introduced.
- This process continuous from step 2 until entire program structure is reached. In top down integration strategy decision making occurs at upper levels in the hierarchy and is encountered first. If major control problems do exists early recognitions is essential.
- If depth first integration is selected a complete function of the software may be implemented and demonstrated.
- Some problems occur when processing at low levels in hierarchy is required to adequately test upper level steps to replace low-level modules at the beginning of the top down testing. So no data flows upward in the program structure.

BOTTOM-UP INTEGRATION TEST

Begins construction and testing with atomic modules. As modules are integrated from the bottom up, processing requirement for modules subordinate to a given level is always available and need for stubs is eliminated. The following steps implements this strategy.

- Low-level modules are combined in to clusters that perform a specific software sub function.
- A driver is written to coordinate test case input and output.
- Cluster is tested.
- Drivers are removed and moving upward in program structure combines clusters. Integration moves upward, the need for separate test driver's lesions.

If the top levels of program structures are integrated top down, the number of drivers can be reduced substantially and integration of clusters is greatly simplified.

REGRESSION TESTING

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

Regression testing maybe conducted manually by executing a subset of all test cases or using automated capture play back tools enables the software engineer to capture the test case and results for subsequent playback and compression. The regression suit contains different classes of test cases.

A representative sample to tests that will exercise all software functions.

Additional tests that focus on software functions that are likely to be affected by the change.

IMPLEMENTATION

Implementation is the process of converting a new or revised system design into operational one. There are three types of Implementation:

- Implementation of a computer system to replace a manual system. The problems encountered are converting files, training users, and verifying printouts for integrity.
- Implementation of a new computer system to replace an existing one. This is usually a difficult conversion. If not properly planned there can be many problems.
- Implementation of a modified application to replace an existing one using the same computer. This type of conversion is relatively easy to handle, provided there are no major changes in the files.

Implementation in Generic tool project is done in all modules. In the first module User level identification is done. In this module every user is identified whether they are genuine one or not to access the database and also generates the session for the user. Illegal use of any form is strictly avoided.

In the Table creation module, the tables are created with user specified fields and user can create many table at a time. They may specify conditions, constraints and calculations in creation of tables. The Generic code maintain the user requirements through out the project.

In Updating module user can update or delete or Insert the new record into the database. This is very important module in Generic code project. User has to specify the filed value in the form then the Generic tool automatically gives whole filed values for that particular record.

In Reporting module user can get the reports from the database in 2Dimentional or 3Dimensional view. User has to select the table and specify the condition then the report will be generated for the user.

TEST CASES

SNO	Test case Title	Pre-requisites	Action	Expected Result	Test Result(pass/fail)
Test case 1	Software requirements	Python version 3.6.5	python --version (Checking the version)	Python 3.6.5 present in your system	Pass
Test case 2	Idle requirements	Jupyter notebook	CMD--(jupyter notebook)	jupyter file should run on the local host	Pass
Test case 3	packages need	pandas, numpy,sklearn,matplotlib,nltk	ls(list of packages)	all packages should import	Pass
Test case 4	Import the datset	Import the dataset by using pandas	Datset found (uploaded into jupyter)	show the dataset in the ipynb file	Pass
Test case 5	Data cleaning	All data should be read from dataset	Apply stemming, lemetization on the data	It will give clean text suitable for machine learning algorithms.	Pass
Test case 6	Prediction	Model should be trained with training dataset	Use trainded model on test dataset for prediction	It will display the predicted results	Pass

Table:9.1 Test cases

10.INPUT AND OUTPUT DESIGN

10. INPUT AND OUTPUT

The following some are the projects inputs and outputs.

Inputs:

- Importing the all required packages like numpy, pandas, matplotlib, scikit – learn and required machine learning algorithms packages.
- Setting the dimensions of visualization graph.
- Downloading and importing the dataset and convert to data frame.

Outputs:

- Preprocessing the importing data frame for imputing nulls with the related information.
- All are displaying cleaned outputs.
- After applying machine learning algorithms it will give good results and visualization plots.

INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization,
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.

- Interface outputs, which involve the user in communicating directly with

The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

11.RESULTS

```
In [25]:
1 import numpy as np
2 import pandas as pd
3 from subprocess import check_output
```

```
In [27]:
1 central_df.head()
```

Out[27]:

	2017-05-06 17:29:01	c4.large	Linux/UNIX	ca-central-1a	0.0139
0	2017-05-06 17:29:01	m4.4xlarge	Windows	ca-central-1b	0.8328
1	2017-05-06 17:29:00	m4.4xlarge	Linux/UNIX	ca-central-1b	0.1051
2	2017-05-06 17:29:00	m4.2xlarge	Windows	ca-central-1b	0.4152
3	2017-05-06 17:29:00	m4.2xlarge	Linux/UNIX	ca-central-1b	0.0532
4	2017-05-06 17:28:49	m4.4xlarge	Linux/UNIX	ca-central-1b	0.1060

```
In [28]:
1 central_df.columns = ['datetime','os','instance_type','price','region']
```

```
In [29]:
1 central_df.head()
```

Out[29]:

	datetime	os	instance_type	price	region
0	2017-05-06 17:29:01	m4.4xlarge	Windows	ca-central-1b	0.8328
1	2017-05-06 17:29:00	m4.4xlarge	Linux/UNIX	ca-central-1b	0.1051
2	2017-05-06 17:29:00	m4.2xlarge	Windows	ca-central-1b	0.4152
3	2017-05-06 17:29:00	m4.2xlarge	Linux/UNIX	ca-central-1b	0.0532
4	2017-05-06 17:28:49	m4.4xlarge	Linux/UNIX	ca-central-1b	0.1060

```
In [31]:
1 from sklearn.preprocessing import LabelEncoder
2 labelencoder = LabelEncoder()
3 central_df['price'] = labelencoder.fit_transform(central_df['price'])
4 central_df.head()
```

Out[31]:

	datetime	os	instance_type	price	region
0	2017-05-06 17:29:01	m4.4xlarge	Windows	1	0.8328
1	2017-05-06 17:29:00	m4.4xlarge	Linux/UNIX	1	0.1051
2	2017-05-06 17:29:00	m4.2xlarge	Windows	1	0.4152
3	2017-05-06 17:29:00	m4.2xlarge	Linux/UNIX	1	0.0532
4	2017-05-06 17:28:49	m4.4xlarge	Linux/UNIX	1	0.1060

```
In [35]:  1 central_df['os'].value_counts()
```

```
Out[35]: m4.large       188057
         c4.large       144251
         m4.2xlarge     130601
         c4.xlarge       90271
         m4.4xlarge      85187
         m4.xlarge       68481
         c4.2xlarge      26618
         c4.8xlarge      23293
         c4.4xlarge      17380
         r4.large        12308
         d2.xlarge       11657
         m4.10xlarge      9471
         m4.16xlarge      8644
         r4.4xlarge       7765
         r4.2xlarge       7141
         i3.8xlarge       6783
         r4.8xlarge       5986
         d2.2xlarge       5053
         r4.16xlarge      4325
         x1.16xlarge      4108
         i3.4xlarge       3824
         d2.8xlarge       3075
         i3.large         3007
         i3.xlarge        2942
         i3.2xlarge       2502
         r4.xlarge        2463
         d2.4xlarge       2293
         i3.16xlarge      2101
```

Random Forests

n [41]:
```
1 from sklearn.ensemble import RandomForestRegressor
2 clf = RandomForestRegressor(max_depth=2, random_state=0)
```

n [42]:
```
1 clf.fit(X1_train,y1_train)
2
3 Pred = clf.predict(X1_test)
```

58s
```
clf.fit(X1_train,y1_train)

Pred = clf.predict(X1_test)
```

0s
```
[21] print(Pred)
```

```
[3358.96139489  667.13528485 3358.96139489 ...  667.13528485  667.13528485
 3358.96139489]
```

12.CONCLUSION&FUTURE WORKS

Amazon EC2 spot historical price to predict the price of 1-day-ahead and 1-week-ahead prices for spot instance by establishing a random forests regression.

Experiments are performed and compared with neural network, support vector machine regression, regression tree and other methods.

In this case, if the distance between x[axis] and node[axis] is less than the maximum distance between k nearest nodes set and X ,

we should search neatest node in node 's right subtree (line [17][18][19]. This is a similar case when x[axis] is more than node[axis]

(line [21][22][23][24][25][26][27]. Finally, this algorithm will return the set of k nearest nodes.

Evaluated algorithms In this paper, we use 5 algorithms as comparison methods which are Linear Regression (LR) [22],

Support Vector Machine Regression (SVR) , Random Forest (RF) [25], Multi-layer Perception Regression.

Spot pricing encourages users to shift execution of flexible workloads from provider's peak hours to off-peak hours and thus obtain monetary incentives. Analysis of one year spot price history data shows that there are sufficient number of time epochs of duration ranging from 30 days to more than 100 days and even longer when spot prices are up to 6 to 8 times cheaper than on-demand prices. It is therefore reasonable for users to shift their workloads from on-demand to spot instances. This work presents application of RRFs for Amazon EC2 spot price prediction. We compare several non-parametric machine learning prediction algorithms for spot price prediction in terms of various forecasting accuracy measures and conclude that RRFs outperforms other methods. Evaluation results show that $MAPE$ <= 10% for 66 to 92% and $MCPE$ <= 15% for 35 to 81% of one-day-ahead predictions in different regions. $MAPE$ <= 15% for 71 to 96% for One-week-ahead predictions in different regions. Instance types with lower infrastructure show better prediction accuracy than those with higher infrastructure. The prediction results suggest that spot price predictions using RRFs are more accurate than other machine learning algorithms. Prediction time for all predictions is less than 1 second. This indicates that the RRFs based predictions are fast enough to predict spot prices. Furthermore, RRFs when used for spot price predictions have

only a few adjustable parameters namely number of regression trees and leaf size. One-week-ahead and one-day-ahead predictions along with feature importance can be effectively used by spot users to plan job executions in advance and bid effectively leading to significant cost savings and reducing out-of-bid failure probability of spot instances.

13.REFERENCES

1. Saha, Debashis. "How Small and Medium Enterprises (SMEs) Should Bid for Spot Instances of Amazon's EC2 Cloud." International Journal of Business Data Communications and Networking (IJBDCN) 10.4 (2014): 43-59.
2. M. Mazzucco and M. Dumas, “Achieving Performance and Availability Guarantees with Spot Instances,” 2011 IEEE International Conference on High Performance Computing and Communications, Sep. 2011.
3. H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, “Optimal Resource Rental Planning for Elastic Applications in Cloud Market,” 2012 IEEE 26th International Parallel and Distributed Processing

   Symposium, May 2012
4. B. Javadi, R. K. Thulasiram, and R. Buyya, “Characterizing spot price dynamics in public cloud environments,” Future Generation Computer Systems, vol. 29, no. 4, pp. 988–999, Jun. 2013.
5. B. Kamiński and P. Szufel, “On optimization of simulation execution on Amazon EC2 spot market,” Simulation Modelling Practice and Theory, vol. 58, pp. 172–187, Nov. 2015.
6. S. Karunakaran and R. P. Sundarraj, “Bidding Strategies for Spot Instances in Cloud Computing Markets,” IEEE Internet Computing, vol. 19, no. 3, May 2015, pp. 32–40.
7. M. Zafer, Y. Song, and K.W. Lee, “Optimal Bids for Spot VMs in a Cloud for Deadline Constrained Jobs,” Proceedings of the IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA 24-29 June 2012.
8. W. Guo, K. Chen, Y. Wu, and W. Zheng, “Bidding for Highly Available Services with Low Price in Spot Instance Market,” Proceedings of the 24th International Symposium on HighPerformance Parallel and Distributed Computing (HPDC 2015), Portland, Oregon, USA, Jun 15 - 19, 2015, pp. 191-202.
9. N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A.N. Tantawi, and C. Krintz, "See Spot Run: Using Spot Instances for MapReduce Workflows", Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud, 2010), Boston, MA, Jun 22 - 25, 2010. 10. O.Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, “Deconstructing Amazon EC2 Spot Instance Pricing,” ACM Transactions on Economics and Computation, vol. 1, no. 3, pp. 1–20, Sep. 2013.

11. S. Tang, J. Yuan, and X.-Y. Li, “Towards Optimal Bidding Strategy for Amazon EC2 Cloud Spot Instance,” 2012 IEEE Fifth International Conference on Cloud Computing, Jun. 2012.
12. Turchenko, Volodymyr, et al.,, "Spot price prediction for cloud computing using neural networks." International Journal of Computing 12.4 (2014): 348-359.
13. Wolski, Rich, and John Brevik “Providing Statistical Reliability Guarantees in the AWS Spot Tier,” 24th High Performance Computing Symposium, 2016.
14. V. K. Singh and K. Dutta, “Dynamic Price Prediction for Amazon Spot Instances,” 2015 48th Hawaii International Conference on System Sciences, Jan. 2015.
15. A. Marathe, R. Harris, D. Lowenthal, B. R. de Supinski, B. Rountree, and M. Schulz, “Exploiting redundancy for cost-effective, timeconstrained execution of HPC applications asaon amazon EC2,” Proceedings of the 23rd international symposium on Highperformance parallel and distributed computing - HPDC ’14, 2014.
16. D. Poola, K. Ramamohanarao, and R. Buyya, “Fault-tolerant Workflow Scheduling using Spot

Instances on Clouds,” Procedia Computer Science, vol. 29, pp. 523–533, 2014

International Journal of Scientific Research in Computer Science, Engineering and Information Technology

ISSN : 2456-3307

OPEN ACCESS

Available Online at : www.ijsrcseit.com

doi : https://doi.org/10.32628/IJSRCSEIT

Improving Amazon EC2 Spot Instances Price Prediction using Machine Learning Algorithm

[1]M. Prasanthi, [2]G.Chishma, [3]P. Padmavathi, [4]K.Reethika, [5]A.Chandra Sekhar

[1]Assistant Professor, [2,3,4,5] UG Students

Department of CSE, Sri Vasavi Institute of Engineering and Technology, Nandamuru, Andhra Pradesh, India

ARTICLEINFO

Article History:

Accepted: 10 April 2024

Published: 22 April 2024

Publication Issue

Volume 10, Issue 2

March-April-2024

Page Number

713-720

ABSTRACT

Spot instances were introduced by Amazon EC2 in December 2009 to sell its spare capacity through auction based market mechanism. Despite its extremely low prices, cloud spot market has low utilization. Spot pricing being dynamic, spot instances are prone to out-of bid failure. Bidding complexity is another reason why users today still fear using spot instances. This work aims to present Regression Random Forests (RRFs) model to predict one-week-ahead and one-day-ahead spot prices. The prediction would assist cloud users to plan in advance when to acquire spot instances, estimate execution costs, and also assist them in bid decision making to minimize execution costs and out-of-bid failure probability. Simulations with 12 months real Amazon EC2 spot history traces to forecast future spot prices show the effectiveness of the proposed technique. Comparison of RRFs based spot price forecasts with existing non-parametric machine learning models reveal that RRFs based forecast accuracy outperforms other models. We measure predictive accuracy using MAPE, MCPE, OOBError and speed. Evaluation results show that $MAPE$ <= 10% for 66 to 92% and $MCPE$ <= 15% for 35 to 81% of one-day-ahead predictions with prediction time less than one second. $MAPE$ <= 15% for 71 to 96% of one-week-ahead predictions. Index Terms— Amazon EC2, Compute instances, One-day-ahead prediction, One-week-ahead prediction, Regression Random Forests, Spot instances, Spot price prediction.

Keywords : Amazon EC2, Compute instances, One-day-ahead prediction, One-week-ahead prediction, Regression Random Forests, Spot instances, Spot price prediction

I. INTRODUCTION

The on-demand scalability characteristic of cloud computing forces cloud service providers to overestimate their resources to meet peak load demand of its customers which happens at different time periods and may not overlap. Due to over-estimation, a large number of cloud resources are idle during off peak hours. Cloud providers also face the problem of allocating resources, keeping in view user‟s different job requirements and data center capacity. Different types of users, multiple types of requirements further alleviate the resource allocation problem. Also, demand for cloud resources fluctuate due to today‟s usage based pricing plans. In order to manage these demand fluctuations more flexible pricing plans are required to sell resources according to real time market demand. Spot pricing was introduced by Amazon EC2 in December 2009 to minimize operational cost, combat under utilization of its resources and make more profit. Similar to on-demand instances, spot instances offer several instance types comprising different combinations of CPU, memory, storage and networking capacity. Amazon Web Service (AWS) is not the only participant in the spot instance realm. Google Compute Engine launched its preemptible Virtual Machines on September 8, 2015 designed for such type of workloads that can be delayed and are fault tolerant at the same time. Users can bid for spot instances (SIs) where prices are charged at lowest bid price, whereas, pricing on Google Preemptible VMs is fixed at per hour rate. The distinguishing feature of Amazon Elastic Compute Cloud (EC2) spot instance is its dynamic pricing. From customer's perspective, spot instances offer prospects of low cost utility computing at a risk of out-of-bid failure at any time by Amazon EC2. Spot instance reliability depends on the market price and user's maximum bid (limited by their hourly budget). Spot prices vary dynamically with real-time based on demand (user's bid) and supply (resource availability) for spot instance capacity in the data centers across the globe. User's bids for spot instances and control the balance of reliability versus monetary cost. The price for spot instances sometimes can be as low as one eighth of the price of on-demand instances. On the other hand, it is also not uncommon that spot prices surpass on-demand prices in cloud data centers. When the demand is low, spot prices are low because less numbers of users are bidding for the same instance. Therefore, a bidder‟s probability of incurring less monetary cost is higher. On the other hand, when the demand is high, users are willing to pay high to get access and hence spot prices increase.

Amazon Web Services (AWS) provides virtual computing environments via their EC2 service. You can launch instances with your favourite operating system, select pre-configured instance images or create your own.Why this is revelant to data sciensits is because generally to run deep learning models you need a machine with a good GPU. EC2 can be configured with a P2/P3 instance and can be configured with up to 8 or 16 GPUs respectively!However, you can request Spot Instance Pricing. Which basically charges you for the spot price that is in effect for the duration of your instance running time.They are adjusted based on long-term trends in supply and demand for Spot instance capacity. Spot instances can be discounted at up to 90% off compared to On-Demand pricing.

II.RELATED WORK

Title: Layer Recurrent Neural Network Based Power System Load Forecasting

This paper presents a straight forward application of Layer Recurrent Neural Network (LRNN) to predict the load of a large distribution network. Short term load forecasting provides important information about the system’s load pattern, which is a premier requirement in planning periodical operations and facility expansion. Approximation of data patterns for forecasting is not an easy task to perform. In past, various approaches have been applied for forecasting. In this work application of LRNN is explored. The results of proposed architecture are compared with other conventional topologies of neural networks on

the basis of Root Mean Square of Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). It is observed that the results obtained from LRNN are comparatively more significant.

Title-EMOTION RECOGNITION IN HUMANS AND MACHINE

This paper presents a straight forward application of Layer Recurrent Neural Network (LRNN) to predict the load of a large distribution network. Short term load forecasting provides important information about the system's load pattern, which is a premier requirement in planning periodical operations and facility expansion. Approximation of data patterns for forecasting is not an easy task to perform. In past, various approaches have been applied for forecasting. In this work application of LRNN is explored. The results of proposed architecture are compared with other conventional topologies of neural networks on the basis of Root Mean Square of Error (RMSE), Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE). It is observed that the results obtained from LRNN are comparatively more significant.

Title-A Double Auction Mechanism to Bridge Users' Task Requirements and Providers' Resources in Two-Sided Cloud Markets

Double auction-based pricing model is an efficient pricing model to balance users' and providers' benefits. Existing double auction mechanisms usually require both users and providers to bid with the unit price and the number of VMs. However, in practice users seldom know the exact number of VMs that meets their task requirements, which leads to users' task requirements inconsistent with providers' resource. In this paper, we propose a truthful double auction mechanism, including a matching process as well as a pricing and VM allocation scheme, to bridge users' task requirements and providers' resources in two-sided cloud markets. In the matching process, we design a cost-aware resource algorithm based on Lyapunov optimization techniques to precisely obtain the number of VMs that meets users' task requirements. In the pricing and VM allocation scheme, we apply the idea of second-price auction to determine the final price and the number of provisioned VMs in the double auction. We theoretically prove our proposed mechanism is individual-rational, truthful and budget-balanced, and analyze the optimality of proposed algorithm. Through simulation experiments, the results show that the individual profits achieved by our algorithm are 12.35 and 11.02 percent larger than that of scaleout and greedy scale-up algorithms respectively for 90 percent of users, and the social welfare of our mechanism is only 7.01 percent smaller than that of the optimum mechanism in the worst case.

Title-Dynamic and Heterogeneous Ensembles for Time Series Forecasting

This paper addresses the issue of learning time series forecasting models in changing environments by leveraging the predictive power of ensemble methods. Concept drift adaptation is performed in an active manner, by dynamically combining base learners according to their recent performance using a non-linear function. Diversity in the ensembles is encouraged with several strategies that include heterogeneity among learners, sampling techniques and computation of summary statistics as extra predictors. Heterogeneity is used with the goal of better coping with different dynamic regimes of the time series. The driving hypotheses of this work are that (i) heterogeneous ensembles should better fit different dynamic regimes and (ii) dynamic aggregation should allow for fast detection and adaptation to regime changes. We extend some strategies typically used in classification tasks to time series forecasting. The proposed methods are validated using Monte Carlo simulations on 16 realworld univariate time series with numerical outcome as well as an artificial series with clear regime shifts. The results provide strong empirical evidence for our hypotheses. To encourage reproducibility the proposed method is publicly available as a software package. Keywords-dynamic ensembles; time series forecasting.

Title-Bidding Strategies for Amazon EC2 Spot InstancesA Comprehensive Review

Spot instance bid price is among one of the major issues in cloud computing. Bid price is aimed at complete execution of the work flow on spot instances while considering several job requirements such as hardware, latency, deadline and budget constraints. Several state-of-the-art spot bidding strategies have been proposed in the literature for executing jobs on Amazon EC2 spot instances. This paper presents different spot bidding strategies proposed by the authors. It highlights the objectives of each and provides the suitability of each of the proposed bidding strategies based on the type of application, its fault tolerance, job requirements and other constraints.

III.PROPOSED SYSTEM

The objective of this work is to present and evaluate a predictive model for spot price prediction that can predict future prices with increased accuracy and speed, minimize forecasting errors and predict spot prices sufficiently far in advance to assist cloud spot users in bid decision making process with increased reliability. We compare prediction accuracy of the state of the art non-parametric supervised machine learning algorithms with Regression Random Forests (RRFs) model.

Advantages:

1. The purpose of proposed system is An analysis of the length of time epoch durations when spot price is less than on-demand price to raise users confidence level in opting for spot instances.
2. Spot price forecasting. We resort to machine learning based ensemble method namely RRFs for one-week ahead and one-day-ahead spot price prediction. The approach focuses on both prediction accuracy and speed is high.

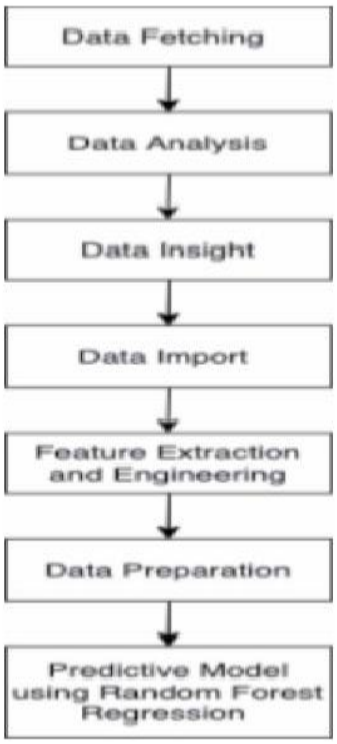

Fig 1 : Proposed work flow

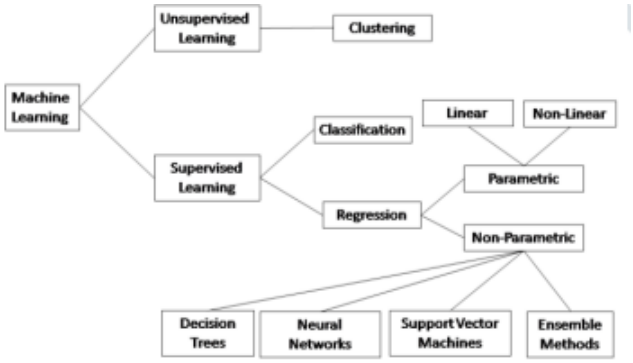

Fig 2 : Classification

Classification predicts categorical (discrete, unordered) labels, whereas prediction or regression is most often used for numeric prediction. Algorithms that do not make strong assumptions or make fewer assumptions about the form of the mapping function are called nonparametric algorithms. By not making assumptions, they are free to learn any functional form from the training data and automatically adapt easily to changes in the underlying time dynamics with varying characteristics. Popular non-parametric regression machine learning algorithms are:

- Support Vector Machines
- Multilayer Feed forward Neural Networks
- Decision Trees (Classification/Regression)
- Regression Tree Ensembles
- Random Forests

"Wisdom of crowds" refers to the phenomenon in which aggregated predictions from a large group of people can be more accurate than most individual judgments and can rival or even beat the accuracy of subject matter experts. Ensemble methods are learning models that achieve performance by combining the opinions of multiple learners. In doing so one can often get away with using much simpler learners and still achieve great performance in terms of increased robustness and accuracy.

IV. RESULTS AND DISCUSSION

```
In [*]:
import numpy as np
import pandas as pd
from subprocess import check_output

In [26]:
central_df = pd.read_csv("ca-central-1.csv");

In [27]:
central_df.head()
```

Out[27]:

	2017-05-06 17:29:01	c4.large	Linux/UNIX	ca-central-1a	0.0139
0	2017-05-06 17:29:01	m4.4xlarge	Windows	ca-central-1b	0.8328
1	2017-05-06 17:29:00	m4.4xlarge	Linux/UNIX	ca-central-1b	0.1051
2	2017-05-06 17:29:00	m4.2xlarge	Windows	ca-central-1b	0.4152
3	2017-05-06 17:29:00	m4.2xlarge	Linux/UNIX	ca-central-1b	0.0532
4	2017-05-06 17:28:49	m4.4xlarge	Linux/UNIX	ca-central-1b	0.1060

Fig 3. Results screenshot

```
In [21]:
from sklearn.model_selection import train_test_split

In [22]:

In [*]:
from sklearn.linear_model import SGDRegressor
clf1 = SGDRegressor()
clf1.fit(X1_train,y1_train)

y1_rbf = clf1.predict(X1_test)
```

Fig 4. Results screenshot

Random Forests

```
In [24]:
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor(max_depth=2, random_state=0)

In [*]:
clf.fit(X1_train,y1_train)

Pred = clf.predict(X1_test)
```

Fig 5. Results screenshot

V. CONCLUSION

Spot pricing encourages users to shift execution of flexible workloads from provider's peak hours to off-peak hours and thus obtain monetary incentives. Analysis of one year spot price history data shows that there are sufficient number of time epochs of duration ranging from 30 days to more than 100 days and even longer when spot prices are up to 6 to 8 times cheaper than on-demand prices. It is therefore reasonable for users to shift their workloads from on-demand to spot instances. This work presents application of RRFs for Amazon EC2 spot price prediction. We compare several non-parametric machine learning prediction algorithms for spot price prediction in terms of various

forecasting accuracy measures and conclude that RRFs outperforms other methods. Evaluation results show that $MAPE$ <= 10% for 66 to 92% and $MCPE$ <= 15% for 35 to 81% of one-day-ahead predictions in different regions. $MAPE$ <= 15% for 71 to 96% for One-week-ahead predictions in different regions. Instance types with lower infrastructure show better prediction accuracy than those with higher infrastructure. The prediction results suggest that spot price predictions using RRFs are more accurate than other machine learning algorithms. Prediction time for all predictions is less than 1 second. This indicates that the RRFs based predictions are fast enough to predict spot prices. Furthermore, RRFs when used for spot price predictions have only a few adjustable parameters namely number of regression trees and leaf size. One-week-ahead and one-day-ahead predictions along with feature importance can be effectively used by spot users to plan job executions in advance and bid effectively leading to significant cost savings and reducing out-of-bid failure probability of spot instances.

VI. FUTURE WORK

Amazon EC2 spot historical price to predict the price of 1-day-ahead and 1-week-ahead prices for spot instance by establishing a random forests regression. Experiments are performed and compared with neural network, support vector machine regression, regression tree and other methods. In this case, if the distance between x[axis] and node[axis] is less than the maximum distance between k nearest nodes set and X ,we should search neatest node in node 's right subtree (line [17][18][19]. This is a similar case when x[axis] is more than node[axis]
(line [21][22][23][24][25][26][27]. Finally, this algorithm will return the set of k nearest nodes. Evaluated algorithms In this paper, we use 5 algorithms as comparison methods which are Linear Regression (LR) [22],Support Vector Machine Regression (SVR) , Random Forest (RF) [25], Multi-layer Perception Regression.

REFERENCES

[1]. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. Deconstructing amazon ec2 spot instancepricing. ACM Transactions on Economics and Computation 1, 3 (2013), 16.

[2]. Amazon Web Services. AWS EC2 Spot Instance Price Histories. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html Accessed Jul-2018.

[3]. Amazon Web Services. New AWS EC2 Spot Instance Pricing. https://aws.amazon.com/about-aws/whats-new/2017/11/amazon-ec2-spot-introduces-new-pricing-model-and-the-ability-to-launch-new-spot-instances-via-runinstances-api/Accessed Jul-2018.

[4]. Amazon Web Services. Amazon EC2. https://aws.amazon.com/ec2/, 2018. [Online; accessed July-2018].

[5]. Amazon Web Services. Amazon ec2 service level agreement, 2018. https://aws.amazon.com/ec2/sla/ accessed July2018.

[6]. Amazon Web Services. Amazon ec2 spot instances, 2018. https://aws.amazon.com/ec2/spot/ accessed July 2018.

[7]. Amazon Web Services. Amazon instance pricing, 2018. https://aws.amazon.com/ec2/pricing/on-demand/ accessedJuly 2018.

[8]. Amazon Web Services. Amazon simple storage service, 2018. https://aws.amazon.com/s3 accessed July 2018.

[9]. Amazon Web Services. How spot instances work, July 2018.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/how-spot-instances-work.html accessed July 2018.

[10]. Amazon Web Services. New amazon ec2 spot pricing model, July 2018. https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/ accessed July 2018.

[11]. Amazon Web Services. New amazon ec2 spot pricing model: Simplified purchasing without bidding and fewer interrup tions, 2018. https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/ accessed August 2018.

[12]. Amazon Web Services. New amazon ec2 spot pricing model: Simplified purchas ing without bidding and fewer interruptions, 2018. https://aws.amazon.com/blogs/aws/amazon-ec2-update-streamlined-access-to-spot-capacity-smooth-price-changes-instance-hibernation/accessedAugust 2018.

[13]. Amazon Web Services. New ec2 spot instance termination notices, July 2018. https://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/ accessed July 2018.

[14]. Aristotle Cloud Federation. https://federatedcloud.org [Online; accessed Aug-2018].

[15]. Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., and Krintz, C. See Spot Run: Using SpotInstances for MapReduce Workflows. In Usenix HotCloud (2010).

[16]. Google Cloud Platform. Google preemptable virtual machines, July 2018. https://cloud.google.com/preemptible-vms/ accessed July 2018.

[17]. He, X., Shenoy, P., Sitaraman, R., and Irwin, D. Cutting the cost of hosting online services using cloud spot markets.In Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (New York,NY, USA, 2015), HPDC '15, ACM, pp. 207–218.

[18]. Javadi, B., Thulasiram, R. K., and Buyya, R. Characterizing spot price dynamics in public cloud environments. FutureGeneration Computer Systems 29, 4 (2013), 988–999.

[19]. Khodak, M., Zheng, L., Lan, A. S., Joe-Wong, C., and Chiang, M. Learning cloud dynamics to optimize spot instancebidding strategies.

[20]. 20] Nurmi, D., Brevik, J., and Wolski, R. Qbets: Queue bounds estimation from time series. In Job Scheduling Strategiesfor Parallel Processing (2008), Springer, pp. 76–101.

[21]. Nurmi, D., Wolski, R., and Brevik, J. Probabilistic advanced reservations for batch-scheduled parallel machines. InProceedings of the 13th ACM SIGPLAN symposium on principles and practice of parallel programming (2008), ACM,pp. 289–290.

[22]. Schwarz, G. Estimating the dimension of a model. Annals of Statistics 6, 2 (1978).

[23]. Song, J., and Guerin, R. Pricing and bidding strategies for cloud computing spot instances. In 2017 IEEE Conferenceon Computer Communications Workshops (INFOCOM WKSHPS) (May 2017), pp. 647–653.

[24]. Steve Fox. New aws spot pricing model: The good, the bad, and the ugly, July 2018. http://autoscalr.com/2018/01/04/new-aws-spot-pricing-model-good-bad-ugly/ accessed July 2018.

[25]. Subramanya, S., Guo, T., Sharma, P., Irwin, D., and Shenoy, P. Spoton: A batch computing service for the spotmarket. In Proceedings of the Sixth ACM Symposium on Cloud Computing (New York, NY, USA, 2015), SoCC '15, ACM,pp. 329–341.

[26]. Wallace, R. M., Turchenko, V., Sheikhalishahi, M., Turchenko, I., Shults, V., Vazquez-Poletti, J. L., andGrandinetti, L. Applications of neural-based spot market prediction for cloud computing. In Intelligent Data Acquisitionand Advanced Computing Systems (IDAACS), 2013

IEEE 7th International Conference on (2013), vol. 2, IEEE, pp. 710–716.

[27]. Willsky, A. S., and Jones, H. L. A generalized likelihood ratio approach to the detection and estimation of jumps inlinear systems. IEEE Transactions on Automatic Control 21, 1 (1976).

[28]. Wolski, R., Brevik, J., Chard, R., and Chard, K. Probabilistic guarantees of execution duration for amazon spotinstances. In Proceedings of the International Conference for High Performance Computing, Networking, Storage andAnalysis (2017), ACM, p. 18.

[29]. Yi, S., Kondo, D., and Andrzejak, A. Reducing costs of spot instances via checkpointing in the amazon elastic computecloud. In 2010 IEEE 3rd International Conference on Cloud Computing (July 2010), pp. 236–243.

[30]. Zheng, L., Joe-Wong, C., Tan, C. W., Chiang, M., and Wang, X. How to bid the cloud. In Proceedings of the 2015ACM Conference on Special Interest Group on Data Communication (New York, NY, USA, 2015), SIGCOMM '15, ACM,pp. 71–84.

CHAPTER 1
INTRODUCTION

CHAPTER 2
OBJECTIVE OF THE PROJECT

CHAPTER 3
LITERATURE SURVEY

CHAPTER 4
SYSTEM ANALYSIS

CHAPTER 5
SYSTEM REQUIREMENT SPECIFICATION

CHAPTER 6

SYSTEM DESIGN

CHAPTER 7
TECHNOLOGY DESCRIPTION AND IMPLEMENTATION

CHAPTER 8
SOURCE CODE

CHAPTER 9
TESTING AND TEST CASES

CHAPTER 10
INPUT AND OUTPUT

DESIGN

CHAPTER 11
RESULTS

CHAPTER 12
CONCLUSION & FUTURE WORK

CHAPTER 13
REFERENCES