

AN RELIABLE AND OPTIMIZED APPROACH FOR SPOT PRICE PREDICTION USING TIME SERIES

ANALYSIS

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

GUNTUPALLI SATISH BABU (Reg. No37110256)

GUNDA SUSHANTH (Reg. No:37110252)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119**

MARCH - 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **GUNTUPALLI SATISH BABU (37110256)** and **GUNDA SUSHANTH (37110252)** who carried out the project entitled "**An RELIABLE AND OPTIMIZED APPROACH FOR SPOT PRICE PREDICTION USING TIME SERIES ANALYSIS**" under my supervision from **August 2020 to March 2021**.

Internal Guide

Dr. A. Vijji Amutha Mary, M.E., Ph.D.,

Head of the Department

Dr. S. Vigneshwari, M.E., Ph.D.,
Dr. L. Lakshmanan, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **GUNTUPALLI SATISH BABU(37110256)** hereby declare that the Project Report entitled "**An RELIABLE AND OPTIMIZED APPROACH FOR SPOT PRICE PREDICTION**

USING TIME SERIES ANALYSIS" is done by me under the guidance of **DR. A. VIJJI AMUTHA MARY, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **DR. A. VIJJI AMUTHA MARY, M.E., Ph.D Assistant Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Public cloud providers, such as Amazon EC2, offer idle computing resources known as spot instances at a much cheaper rate compared to On-Demand instances. Spot instance prices are set dynamically according to market demand. Cloud user's request spot instances by submitting their bid, and if user's bid price exceeds current spot price then a spot instance is assigned to that user. The problem however is that while spot instances are executing their jobs, they can be revoked whenever the spot price rises above the current bid of the user. In such scenarios and to complete jobs reliably, we propose a set of improvements for the cloud spot market which benefits both the provider and users.

Based on an arccotangent-function-type system model, our novel bidding mechanism imitates fuzzy and intuitive human activities to refine and issue new bids according to previous errors. The validation is conducted by using Amazon's historical spot price trace to perform a set of simulations and comparisons. The result shows that the feedback control-based mechanism obtains a better trade-off between bidding rationality and success rate than the other five comparable strategies. Although this mechanism is only for black-box bidding (price prediction) at this current stage, it can be conveniently and gradually upgraded to take into account external constraints in the future.

TABLE OF CONTENTS

ABSTRACT	V
LIST OF FIGURES	Vii

CHAPTER No.	TITLE	PAGE No.
1	INTRODUCTION	1
	1.1 OVERVIEW	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 DOMAIN OVERVIEW	1
	1.3.1 MACHINE LEARNING	2
2	LITERATURE SURVEY	5
3	SYSTEM ANALYSIS AND REQUIREMENT SPECIFICATION	13
	3.1 EXISTING SYSTEM	17
	3.2 PROPOSED SYSTEM	18
	3.3 HARDWARE & SOFTWARE SPECIFICATION	19
	3.3.1 HARDWARE SPECIFICATION	19
	3.3.2 SOFTWARE SPECIFICATION	20
	3.4 SOFTWARE DESCRIPTION	20
	3.4.1 ABOUT PYTHON	20
	3.4.2 ABOUT IDE	25
4	SYSTEM DESIGN AND IMPLEMENTATION	28
	4.1 ARCHITECTURE DIAGRAM	28
	4.2 LIST OF MODULES	29
5	SOFTWARE TESTING	33
	5.1 GENERAL	38
	5.2 TEST DRIVEN DEVELOPMENT	39
	5.3 UNIT TESTING	41
	5.4 BLACKBOX TESTING	41
	5.5 INTEGRATION TESTING	42
	5.6 SYSTEM TESTING	43
	5.7 SANITY TESTING	44
	5.8 REGRESSION TESTING	46
6	RESULTS	50
7	CONCLUSION	57
	REFERENCES	58
	SOURCE CODE	59

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	WATERFALL MODEL	15
3.2	RAD MODEL	16
4.1	ARCHITECTURE DIAGRAM	31
5.1	GENERAL	41
5.2	INTEGRATION TESTING	45
5.3	SYSTEM TESTING	46
5.4	REGRESSION TESTING	50

CHAPTER 1

INTRODUCTI

ON

This proposed system reports the typical content of this method kit, including a group of central tendency analysis methods and two types of distribution modeling analysis methods. In a generic sense, we particularly argue that open methods act as a higher- level strategy over open-source tools and open-access data for scientific studies in any research domain.

1.1 OVERVIEW

It measures the accuracy or quality of the algorithm. High bias means a poor match. The error due to bias is the difference between the expected (or average) prediction of any model and the correct value of the response variable which we are trying to predict. Variance of an algorithm is defined as the expected value of the squared difference between any particular hypotheses and the averaged hypothesis. Variance measures how predictions vary with respect to the average prediction. A high variance means a weak match.

1.2 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

1.3 DOMAIN OVERVIEW

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it

trains over the

analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future. Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

1.3.1 MACHINE LEARNING

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms. However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyper parameters to reach an optimum result. The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry. Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

Machine Learning Algorithms can be classified into 3 types as follows –

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

SUPERVISED LEARNING

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism’s genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?” and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x_1) but also number of bedrooms (x_2), number of bathrooms (x_3), number of floors (x_4), year built (x_5), zip code (x_6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a

single input value is used.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

REGRESSION

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

1. Linear Regression
2. Regression Trees
3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial Regression

CLASSIFICATION

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Tree

CHAPTER 2

LITERATURE SURVEY

2.1a

Literature Survey 1	
Title	Cost-Effective Cloud Server Provisioning for Predictable Performance of Big Data Analytics
Authors	Fei Xu, Haoyue Zheng, Huan Jiang, Wujie Shao, Haikun Liu
Published Year	2018
Efficiency	✓ Gleaned very interesting insights from Spot Price data ✓ Analyzed pricing volatility ✓ Provides temporal and spatial trend analysis
Drawbacks	□ Overall cost of deployment of a Spot Instance is high □ Disappearance of sudden spikes are not considered - Does not yield valuable insights into the evolution and structure of a given Spot market

Efficiency	✓ Gleaned very interesting insights from Spot Price data ✓ Analyzed pricing volatility ✓ Provides temporal and spatial trend analysis
Drawbacks	□ Overall cost of deployment of a Spot Instance is high □ Disappearance of sudden spikes are not considered - Does not yield valuable insights into the evolution and structure of a given Spot market

Description	This paper proposes iSpot, a cost-effective transient server provisioning framework for achieving predictable performance in the cloud, by focusing on Spark as a representative Directed Acyclic Graph (DAG)-style big data analytics workload. It first identifies the stable cloud transient servers during the job execution by devising an accurate Long Short-Term Memory (LSTM)-based price prediction method. Leveraging automatic job profiling and the acquired DAG information of stages, we further build an analytical performance model and present a lightweight critical data check pointing mechanism for Spark, to enable our design of iSpot provisioning strategy for guaranteeing the job performance on stable transient servers. Extensive prototype experiments on both EC2 spot instances and GCE preemptible instances demonstrate that, iSpot is able to guarantee the performance of big data analytics running on cloud transient servers while reducing the job budget by up to 83.8% in comparison to the state-of-the-art server provisioning strategies, yet with acceptable runtime overhead. Based on the job preruns on the sample input dataset, we collect the stage set (i.e., S) of the job, and the dependencies among stages (i.e., A_i) using the topological sort information of the stages in the Spark DAG. Specifically, each stage of the Spark job corresponds to a vertex, and the dependency between two stages corresponds to a directed edge. The vertices with the zero in-degree are considered as the stages that can be executed in parallel (i.e., P_i). This paper then delete these vertices with the zero in-degree from the Spark DAG and put them into S. The sorting process above is iteratively executed, provided that the Spark DAG has at least one
-------------	--

	Stage. This paper calculate the maximum execution time of the stages running in parallel, and iteratively add the maximum value to the completion time of the Spark job. Leveraging the event log generated by job pre-runs, we extract several key information such as the sampled input data size, the stage execution time, the size of shuffle write data and that of shuffle read data for each stage in the job pre-runs. Using these job profiling information above, we are able to obtain the data processing parameters of our performance model. For each stage i, the data processing rate R_i can be calculated as the size of the sampled input dataset divided by the stage execution time. The data processing ratio r_{ip} is estimated as the ratio of the shuffle write data size to the shuffle read data size of the stage i. The garbage collection rate r_{igc} is estimated as the ratio of the stage input data size to the garbage collection time.
--	--

2.1b

Literature Survey 2	
Title	AutoBoT: Resilient and Cost-effective Scheduling of a Bag of Tasks on Spot VMs
Authors	Prateeksha Varshney, Yogesh Simmhan
Published Year	2017
Efficiency	✓ Effectively improve prediction accuracy. ✓ Reduces the consumption of hardware resources ✓ Computationally Efficient

Drawbacks	Big payloads □ A lot of pre-development education □ This system is Opportunistic and uncontrollable
Description	In this paper, AutoBoT uniquely reduces costs by including pre-emptible spot-priced VMs that are much cheaper, but are unreliable and have time-variant pricing. It guarantees timely completion by making active runtime decisions on pricing, number Of VMs to acquire/release, and on task placement, check pointing and migration. Our rigorous simulations of 7 Million BoT runs sampled from the Google cluster workload uses a realistic Cloud model and 6 months of Amazon EC2 pricing data to compare AutoBoT against two baseline algorithms. We analyze the impact of BoT size, data centers, time periods, deadline duration, loss budget and check pointing strategies. AutoBoT often gives $\approx 80\%$ profit and rare but bounded losses, compared to using only fixed-price VMs. Further, its 100% completion guarantee is 23 – 42% better than using only spot-priced VMs which offer a similar profit. In this paper, use a Bag of Tasks (BoT) application model that is commonly seen in both compute intensive/task-parallel, and data- intensive/data-parallel applications [1], [2]. The user provides a BoT having n independent tasks $\tau_j \in T$, along with the baseline execution time of each task on the baseline VM r_0, which is given by θ_j time units. This is a reasonable assumption since users typically have an estimate of the runtime of tasks on Cloud VMs since they pay real money for the resources used. Even otherwise, this is a one-time profiling cost, and makes AutoBoT well-suited for BoTs with recurrent tasks from a fixed set. We do not make assumptions about the homogeneity or

	internal logic of the tasks within a BoT. The user also provides a
--	--

	hard deadline of δ time units between the BoT submission time and the successful execution of all its tasks.
--	---

2.1c

Literature Survey 3	
Title	Exploring Fine-Grained Resource Rental Planning in Cloud Computing
Authors	Han Zhao, Miao Pan, Xinxin Liu, Xiaolin Li, Yuguang Fang
Published Year	2015
Efficiency	✓ Determine coefficient of correlation between hourly price pattern ✓ Medium to strong correlation between different compute optimized instance types within any given region ✓ Lowering out-of-bid failure and monetary cost for batch processing of compute optimized jobs
Drawbacks	□ Extreme spikes in the Spot price history are not considered - Difficulties to obtain better performance - Very calculation intensive while training the model

Description	An Application Service Provider (ASP) needs to carefully evaluate various resource rental options before finalizing the application deployment. We choose Amazon EC2 marketplace as a case of study, and analyze the economical trade-off for on- demand resource rental strategies. Given fixed resource pricing, we first develop a deterministic model, using a mixed integer linear program, to facilitate resource rental decision making. Evaluation results show that our planning optimization model reduces resource rental cost by as much as 50% compared with a baseline strategy. Next, we further investigate planning solutions to resource market featuring time-varying pricing (Amazon Spot Instance Market). We perform time- series analysis over the spot price trace and examine its predictability using Auto-Regressive Integrated Moving-Average (ARIMA). We also develop a stochastic planning model based on Multistage recourse. By comparing these two approaches, we discover that spot price forecasting does not provide our planning model with a crystal ball due to the weak correlation of past and future price, and the stochastic planning model better hedges against resource pricing uncertainty than resource rental planning using forecast prices
-------------	---

2.1d

Literature Survey 4	
Title	Monetary Cost Optimizations for Hosting Workflow-as-a-Service in IaaS Clouds
Authors	Amelie Chi Zhou, Bingsheng He and Cheng Liu
Published Year	2014
Efficiency	✓ Highly reduced prices ✓ Guaranteeing the job performance on stable transient servers ✓ Reducing the job budget by up to 83.8% in comparison to the state-of-the-art server provisioning strategies
Drawbacks	- loss budget and check pointing strategies □ Underutilized resources, in the form of cheap yet revocable transient servers □ High resource intensive processing
Description	This paper develops a scheduling system called Dyna to minimize the expected monetary cost given the user-specified probabilistic deadline guarantees. Dyna includes an A*-based instance configuration method for performance dynamics, and a hybrid instance configuration refinement for using spot instances. Experimental results with three scientific workflow applications on Amazon EC2 and a cloud simulator demonstrate (1) the ability of Dyna on satisfying the probabilistic deadline guarantees required by the users; (2) the effectiveness on reducing monetary cost in comparison with the existing approaches.

	When a user has specified the probabilistic deadline requirement for a workflow, WaaS providers schedule the workflow by choosing the cost-effective instance types for each task in the workflow. The overall functionality of the Dyna optimizations is to determine the suitable instance configuration for each task of a workflow so that the monetary cost is minimized while the probabilistic performance requirement is satisfied. We formulate the optimization process as a search problem, and develop a two-step approach to find the solution efficiently.
--	---

CHAPTER 3

SYSTEM ANALYSIS AND REQUIREMENT SPECIFICATION

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

WATERFALL MODEL

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

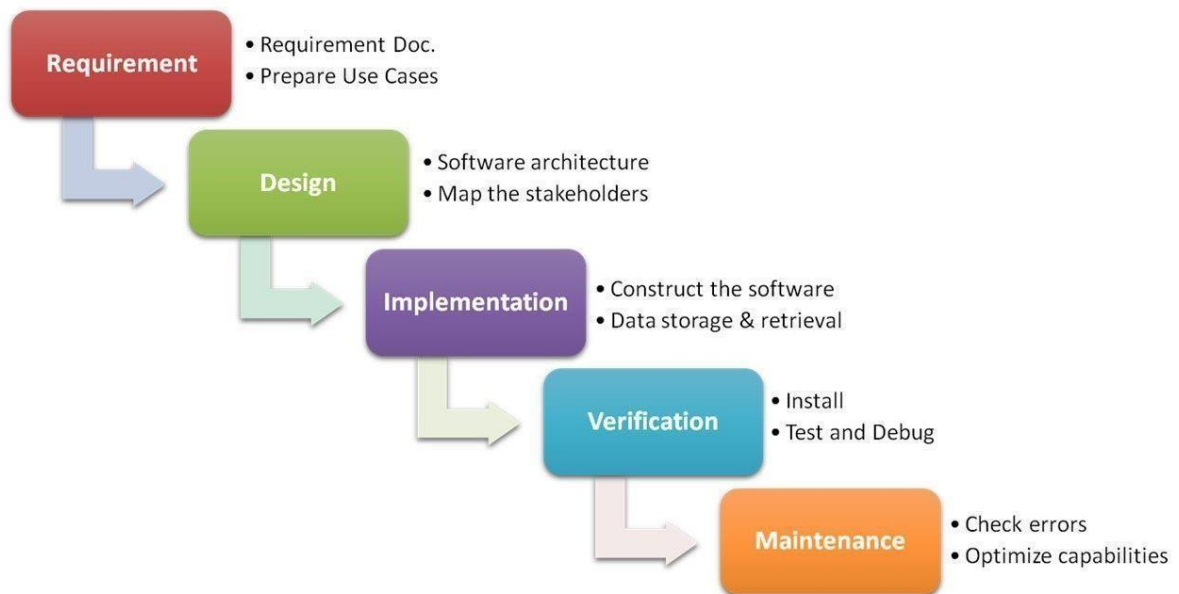


Fig 3.1 Waterfall model

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.

System design

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

Implementation

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

Testing

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase

one for

reevaluation.

Delivery/deploym

ent

In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

Maintenance

The final solution has been implemented to the client and is being used. As troubles arises, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

RAD MODEL

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of a software application in comprehensive functional components.

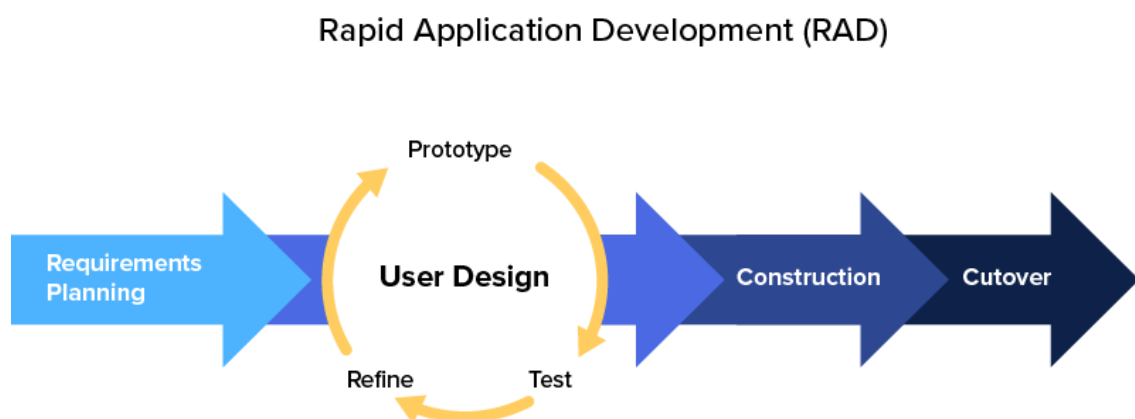


Fig 3.2 Rad model

Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning

phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, web application programmers, clients (software users), and team

members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

Phase 2: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks. Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the

thorough iterative design phase, web application programmers can construct the final

working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

Preparation for rapid construction

Program and application

development Coding

Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmer's work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training. All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients. Here are just a few advantages:

- RAD lets you break the project down into smaller, more manageable tasks.

- The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.

- Clients get a working product delivered in a shorter time frame.

- Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

3.1 EXISTING SYSTEM

The existing system 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 attribute the presence of abruptly high prices to Amazon EC2 spot pricing mechanism itself. This may be due to non-availability of resources in the spot pool in Amazon EC2 regions which terminate all the spot instances allocated. On-demand prices can be considered as an upper bound to bid for spot instances and additionally to increase spot prices. Cloud customers would not choose to acquire unreliable spot instances for long time periods at costs higher than on-demand instances which provide full availability guarantees as it would not result in any monetary savings. We therefore, restrict forecasting period of an instance type to the time period during which spot prices are less than on-demand price for that particular type of instance.

3.2 PROPOSED SYSTEM

Amazon's spot price history cannot directly be treated as pure price values without considering the prices' timestamps. For example, it is inaccurate to calculate the average price (or mean) by using the price values only, because the spot service changes (and records) price irregularly, while its unchanged price can still be recorded from time to time. On the other hand, it is improper to simply apply time series analysis to spot pricing without adapting the spot service's characteristics. For example, the spot service price can change at any time, and thus the price records have irregular time intervals; and Amazon uses irrationally high price to reclaim spot resources, and thus the outliers in the spot price history can be clearly observed and conveniently removed. Therefore, we decided to clarify the methods even for fundamental distribution analyses.

To achieve both successful and rational bids, we assume that the bidding system

wants to use a control signal to correct its previous output errors. Intuitively, the system would decrease its bid in next round if the previous output is too high (for the purpose of rational bidding), while bidding higher if the previous bid is lower than the real spot price (for the purpose of successful bidding). In extreme cases, however, it is unacceptable to bid higher than the price of on-demand service, because Cloud consumers would rather directly employ on demand resources if the spot service is too expensive; and it does not make sense to set a bid infinitely low either, because the Cloud provider could have a reserved price for the cost of developing and maintaining the spot service. In fact, Amazon's spot price traces have shown that there always exists a band in which the price fluctuation happens most of the time

By including the observations on timestamps, we employ the mode of spot price to refer to the price value that appears with the longest timespan in total in a particular price trace. Since the separate timespans for the appearances of the same price are accumulated together.

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need. Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

3.3 HARDWARE & SOFTWARE SPECIFICATION

3.3.1 HARDWARE SPECIFICATION

Hardware Requirement

Hardware	Minimum Requirement
Disk Space	32 GB or more, 10 GB or more for Foundation
Edition Processor	1.4 GHz 64 bit
Memory	512 MB
Display	(800 × 600) capable video adapter and monitor

3.3.2 SOFTWARE SPECIFICATION

Backend Technologies

- ✓ Python
- ✓ Numpy
- ✓ Sci-learn
- ✓ Eclipse IDE

Frontend Technologies

- ✓ Web Technologies
- ✓ Bootstrap

3.4 SOFTWARE DESCRIPTION

3.4.1 ABOUT PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems. Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.

PYTHON LINE STRUCTURE

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter. As we saw that in Python, a new

line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement. Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement.

Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

PYTHON DATA TYPES

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment

- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries
- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth

ABOUT PANDAS

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The Data Frame is one of these structures. Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables. Pandas is built on top of the Numpy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn. Jupyter Notebooks offer a good environment for

using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' Data Frames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and Data Frames.

1. Series: a pandas Series is a one dimensional data structure ("a one dimensional ndarray") that can store values — and for every value it holds a unique index, too.
2. Data Frame: a pandas Data Frame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.

NUMPY

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy. A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

MATPLOTLIB

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib. Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy.

Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams. Matplotlib is a plotting library like GNUplot. The main

advantage

towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

3.4.2 ABOUT IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications. Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

- We can easily write our code in its text editor window.

- To store the resource files of a project, we have a project editor window.

- It makes debugging of the code easier.

- An IDE's autosuggest feature helps us in writing efficient code.

- We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and

grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio
- ✓ LiClipse (Eclipse)

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.

Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.

GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.

High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.

Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.

Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.

Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.

Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

INTRODUCTION

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

4.1 ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

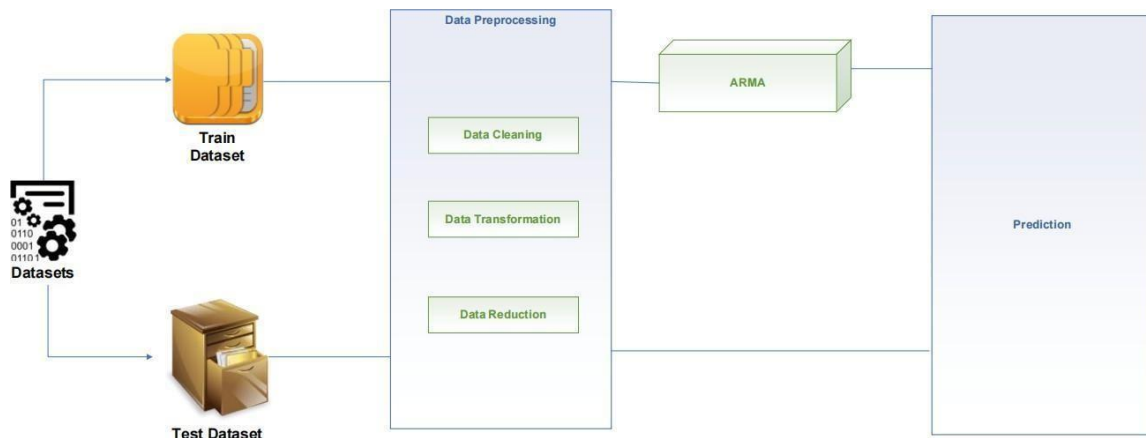


Fig 4.1 Architecture diagram

4.2 LIST OF MODULES

MODULE 1: DATA ANALYSIS

The terms data visualization and information visualization (casually, data viz and info viz) are useful for referring to any visual representation of data that is: algorithmically drawn (may have custom touches but is largely rendered with the help of computerized methods); easy to regenerate with different data (the same form may be repurposed to represent different datasets with similar dimensions or characteristics); often aesthetically barren (data is not decorated); and relatively data-rich (large volumes of data are welcome and viable, in contrast to infographics). Data visualizations are initially designed by a human, but are then drawn algorithmically with graphing, charting, or diagramming software. The advantage of this approach is that it is relatively simple to update or regenerate the visualization with more or new data. While they may show great volumes of data, information visualizations are often less aesthetically rich than infographics.

Exploratory data visualizations are appropriate when you have a whole bunch of data and you're not sure what's in it. When you need to get a sense of what's inside your data set, translating it into a visual medium can help you quickly identify its features, including interesting curves, lines, trends, or anomalous outliers. Exploration is generally best done at a high level of granularity. There

may be a whole lot of noise

in your data, but if you oversimplify or strip out too much information, you could end up missing something important. This type of visualization is typically part of the data analysis phase, and is used to find the story the data has to tell you.

Boto3 makes it easy to integrate your Python application, library, or script with AWS services including Amazon S3, Amazon EC2, Amazon DynamoDB, and more.

Resources represent an object-oriented interface to Amazon Web Services (AWS). They provide a higher-level abstraction than the raw, low-level calls made by service clients. Every resource instance has a number of attributes and methods. These can conceptually be split up into identifiers, attributes, actions, references, sub-resources, and collections. At its core, all that Boto3 does is call AWS APIs on your behalf. For the majority of the AWS services, Boto3 offers two distinct ways of accessing these abstracted APIs.

MODULE 2: DATA CLEANING & TRANSFORMATION

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. There is no one absolute way to prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset. But it is crucial to establish a template for your data cleaning process so you know you are doing it the right way every time. Data cleaning is the process that removes data that does not belong in your dataset. Data transformation is the process of converting data from one format or structure into another. Transformation processes can also be referred to as data wrangling, or data munging, transforming and mapping data from one "raw" data form into another format for warehousing and analyzing.

Remove unwanted observations from your dataset, including duplicate observations or irrelevant observations. Duplicate observations will happen most often during data collection. When you combine data sets from multiple places, scrape data, or receive data from clients or multiple departments, there are

opportunities to create duplicate data. De-duplication is one of the largest areas to be considered in this process.

Irrelevant observations are when you notice observations that do not fit into the specific problem you are trying to analyze.

Structural errors are when you measure or transfer data and notice strange naming conventions, typos, or incorrect capitalization. These inconsistencies can cause mislabeled categories or classes. For example, you may find "N/A" and "Not Applicable" both appear, but they should be analyzed as the same category.

MODULE 3: TIME SERIES ANALYSIS

Time series analysis is a statistical technique that deals with time series data, or trend analysis. Time series data means that data is in a series of particular time periods or intervals. A set of observations on the values that a variable takes at different times.

ARIMA stands for autoregressive integrated moving average. This method is also known as the Box-Jenkins method.

Autoregressive component: AR stands for autoregressive. Autoregressive parameter is denoted by p . When $p = 0$, it means that there is no auto-correlation in the series. When $p = 1$, it means that the series auto-correlation is till one lag.

Integrated: In ARIMA time series analysis, integrated is denoted by d . Integration is the inverse of differencing. When $d = 0$, it means the series is stationary and we do not need to take the difference of it. When $d = 1$, it means that the series is not stationary and to make it stationary, we need to take the first difference. When $d = 2$, it means that the series has been differenced twice. Usually, more than two time difference is not reliable.

Moving average component: MA stands for moving the average, which is denoted by

q . In ARIMA, moving average $q = 1$ means that it is an error term and there is auto-correlation with one lag.

In order to test whether or not the series and their error term is auto correlated, we usually use W-D test, ACF, and PACF.

Decomposition: Refers to separating a time series into trend, seasonal effects,

and remaining variability Assumptions:

Stationarity: The first assumption is that the series are stationary. Essentially, this means that the series are normally distributed and the mean and variance are constant over a long time period.

Uncorrelated random error: We assume that the error term is randomly distributed and the mean and variance are constant over a time period. The Durbin-Watson test is the standard test for correlated errors.

No outliers: We assume that there is no outlier in the series. Outliers may affect conclusions strongly and can be misleading.

Random shocks (a random error component): If shocks are present, they are assumed to be randomly distributed with a mean of 0 and a constant variance.

CHAPTER 5

SOFTWARE

TESTING

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

Benefits of using Documentation

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks.
- It provides feedback for your planning cycle.
- It creates objective evidence for the performance of the quality management system.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

The most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process. Let's take a look at the key requirements for these files and see how they contribute to the process.

Test strategy. An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

Test data. The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, and media content, similar to files that would be uploaded by an end-user in a ready

solution.

Test plans. A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed

planning. Such a document is distributed between team members and shared with all stakeholders.

Test scenarios. In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

Test cases. If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing — functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

Traceability Matrix. This software testing documentation maps test cases and their requirements. All entries have their custom IDs — team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.

External documentation collects information from inner documentation but also emphasize on providing a visual data representation — graphs, diagrams, etc.

External reports — these documents collect information on test results and can describe an entire project or a particular piece of functionality.

Test summary report — the file with final test results and findings, presented to stakeholders.

Bug reports — such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem.

The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

But as a curiosity or Out of the box thinking, have you ever wondered how developers test their own code? What method do they use to test before releasing code for testing? How is dev-testing important in an agile process? The answer to all this is Unit Testing. I want to educate you on the importance of Unit Testing so that development and testing teams can work more collaboratively to design, test and release an excellent application.

Unit testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing

"Unit testing is the method of verifying the smallest piece of testable code against its purpose." If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.



Unit testing is used to design robust software components that help maintain code and eliminate the issues in code units. We all know the importance of finding and fixing defects in the early stage of the software development cycle. Unit testing serves the same purpose.

Unit testing is an integral part of the agile software development process. When a nightly build run unit test suite should run and report should be generated. If any of the unit tests have failed then the QA team should not accept that build for verification.

If we set this as a standard process, many defects would be caught in the early development cycle, saving much testing time.

I know many developers hate to write unit tests. They either ignore or write bad unit test cases due to tight scheduled or lack of seriousness (yea they write empty unit tests, so 100% of them pass successfully ;-)). It's important to write good unit tests or don't write them at all. It's even more important to provide sufficient time and a supportive environment for real benefits.

- Testing can be done in the early phases of the software development lifecycle when other modules may not be available for integration
- Fixing an issue in Unit Testing can fix many other issues occurring in later development and testing stages
- Cost of fixing a defect found in Unit Testing is very less than the one found in the system or acceptance testing
- Code coverage can be measured
- Fewer bugs in the System and Acceptance testing

- Code completeness can be demonstrated using unit tests. This is more useful in the agile process. Testers don't get the functional builds to test until integration is completed. Code completion cannot be justified by showing that you have written and checked in the code. But running Unit tests can demonstrate code completeness.
- Expect robust design and development as developers write test cases by understanding the specifications first.
- Easily identify who broke the build

Saves development time: Code completion may take more time but due to decreased defect count overall development time can be saved.

5.1 GENERAL

Unit Testing frameworks are mostly used to help write unit tests quickly and easily. Most of the programming languages do not support unit testing with the inbuilt compiler. Third-party open source and commercial tools can be used to make unit testing even more fun.

List of popular Unit Testing tools for different programming languages:

Java framework – JUnit

PHP framework – PHPUnit

C++ frameworks – UnitTest++ and Google C++

.NET framework – NUnit

Python framework – py.test

they know how to write. The idea is simple, but like most simple things, the execution is hard. TDD requires a completely different mindset from what most people are used to and the tenacity to deal with a learning curve that may slow you down at first.

Functional Testing types include:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Sanity Testing
5. Smoke Testing
6. Interface Testing
7. Regression Testing
8. Beta/Acceptance Testing
9. Non-functional Testing types

include: Performance Testing

- Load Testing
- Stress Testing
- Volume Testing
- Security Testing
- Compatibility Testing
- Install Testing
- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

5.3 UNIT TESTING

Unit testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing.

A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

Unit testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing –

“Unit testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.

5.4 BLACKBOX TESTING

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit

testing. It checks whether the code works as expected. Because unit testing focuses on the internal structure of the code, it is called the white box testing. On the other hand, functional testing checks app's functionalities without looking at the internal structure of the code, hence it is called black box testing. Despite how flawless the various individual code components may be, it is essential to check that the app is functioning as expected, when all components are combined. Here you can find a detailed comparison between functional testing vs unit testing.

5.5 INTEGRATION TESTING

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.

Component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet).

Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope.

As often with these things, it's best to start with a bit of history. When I first learned about integration testing, it was in the 1980's and the waterfall was the dominant influence of software development thinking. In a larger project, we would have a design phase that would specify the interface and behavior of the various modules in the system. Modules would then be assigned to developers to

program. It was not unusual for one programmer to be responsible for a single module, but this would be big enough

that it could take months to build it. All this work was done in isolation, and when the programmer believed it was finished they would hand it over to QA for testing. Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

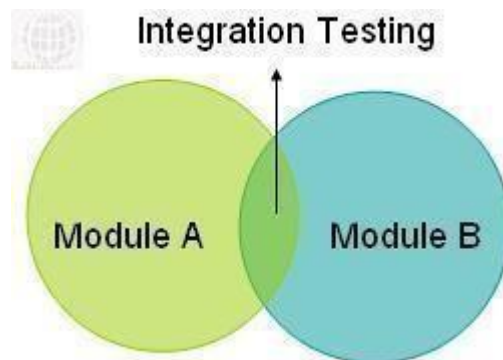


Fig 5.2 Integration testing

5.6 SYSTEM TESTING

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.

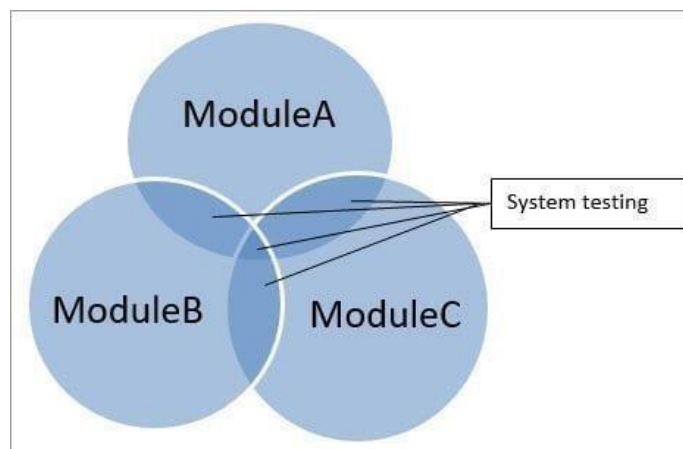


Fig 5.3 System testing

System testing is a method of monitoring and assessing the behavior of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the question "whether the complete system functions in accordance to its pre-defined requirements?"

It's comes under black box testing i.e. only external working features of the software are evaluated during this testing. It does not requires any internal knowledge of the coding, programming, design, etc., and is completely based on users-perspective.

A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This System testing tests the integrated system and validates whether it meets the specified requirements of the client.

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer's point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect both defects within the "inter-assemblages".

5.7 SANITY TESTING

Sanity Testing is done when as a QA we do not have sufficient time to run all the

test cases, be it Functional Testing, UI, OS or Browser Testing. Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected. This testing is a

checkpoint to determine if testing for the build can proceed or not.

The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing. Let's assume that in an e-commerce project, there are mainly five modules- like login page, home page, user profile page, new user creation, etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters. So, the bug is reported by the testing team to the development team to resolve it. When the development team fixes the defect and passes it to the testing team. Then the testing team checks if the changes done are working fine. It also checks if it does not have an impact on other related functionalities. Now there is a functionality to update the password in the user profile page. As part of the sanity check, we would need to validate the login page as well as the profile page to ensure that the checks are working fine at both the places. Sanity tests are generally performed on build where the production deployment is required immediately - like a critical bug fix.

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as "Day 0" check. It is done at the "build level". It helps not to waste the testing time to simply testing the whole application when the key features don't work or the key bugs have not been fixed yet. Here our focus will be on primary and core application work flow. To conduct smoke testing, we don't write test cases. We just pick the necessary test cases from already written test cases.

Do we really write test cases for all testing types? Here in this article, we have given clear idea on choosing testing types to write test cases.

As mentioned earlier, here in Smoke Testing, our main focus will be on core application work flow. So we pick the test cases from our test suite which cover major functionality of the application. In general, we pick minimal number of test

cases that won't take

more than half an hour to execute.

After receiving a Software build with the minor issues fixes in code or functionality, Sanity testing is carry out to check whether the bugs reported in previous build are fixed & there is regression introduced due to these fixes i.e. not breaking any previously working functionality. The main aim of Sanity testing to check the planned functionality is working as expected. Instead of doing whole regression testing the Sanity testing is perform.

Sanity tests helps to avoid wasting time and cost involved in testing if the build is failed. Tester should reject the build upon build failure.

After completion of regression testing the Sanity testing is started to check the defect fixes & changes done in the software application is not breaking the core functionality of the software. Typically this is done nearing end of SDLC i.e. while releasing the software. You can say that sanity testing is a subset of acceptance testing. We can also say Tester Acceptance Testing for Sanity testing.

5.8 REGRESSION TESTING

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change. Regression Testing is a Software Testing type in which test cases are re- executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs. This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix. For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety—such as well-designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization—to prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression

testing

as an essential aspect of a dynamic, iterative development and deployment schedule. But no matter what software development and quality-assurance process your organization uses, if you take the time to put in enough careful planning up front, crafting a clear and diverse testing strategy with automated regression testing at its core, you can help prevent projects from going over budget, keep your team on track, and, most importantly, prevent unexpected bugs from damaging your products and your company's bottom line.

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size. The process incorporates "performance" indicators such as:

- Browser, page, and network response times
- Server request processing times
- Acceptable concurrent user volumes
- Processor memory consumption; number and type of errors that might be encountered with app

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application. Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.



Fig 5.4 Regression testing

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure that application is working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as "Endurance testing" and "Volume testing". The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load. Load testing comes under the Non Functional Testing & it is designed to test the

non-functional requirements of a software application.

Load testing is performed to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero load...

CHAPTER 6

RESULTS

```
In [8]: south.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2031033 entries, 0 to 2031032  
Data columns (total 5 columns):  
#   Column      Dtype  
---  ---  
0   datetime    object  
1   instance_type object  
2   os           object  
3   region       object  
4   price        float64  
dtypes: float64(1), object(4)  
memory usage: 77.5+ MB
```

```
In [9]: south.head()
```

```
Out[9]:
```

	datetime	instance_type	os	region	price
0	2017-03-07 16:05:01	c4.xlarge	Windows	ap-south-1b	1.8865
1	2017-03-07 16:05:01	m4.xlarge	Linux/UNIX	ap-south-1b	0.0366
2	2017-03-07 16:05:01	m4.large	Linux/UNIX	ap-south-1a	0.0243
3	2017-03-07 16:05:01	m4.large	Linux/UNIX	ap-south-1b	0.0346
4	2017-03-07 16:05:01	c4.xlarge	Linux/UNIX	ap-south-1a	0.2895

```
In [10]: south.describe()
```

```
Out[10]:
```

	price
count	2.031033e+06
mean	8.175382e-01
std	7.695543e+00
min	1.450000e-02
25%	3.400000e-02
50%	1.110000e-01
75%	4.368000e-01
max	2.426200e+02

```
In [11]: east = pd.read_csv("ca-central-1.csv", sep=";", header=None, names=columns)
```

```
In [12]: east.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 881526 entries, 0 to 881525  
Data columns (total 5 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   datetime    881526 non-null object  
1   instance_type 881526 non-null object  
2   os           881526 non-null object  
3   region       881526 non-null object  
4   price        881526 non-null float64  
dtypes: float64(1), object(4)
```

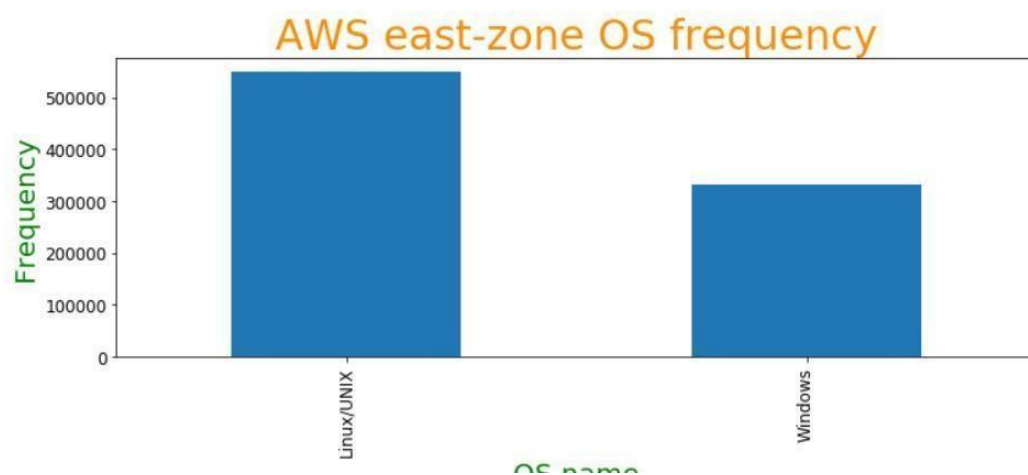
```
] data = east['os'].value_counts()
```

```
] data
```

```
] Linux/UNIX    549423  
Windows        332103  
Name: os, dtype: int64
```

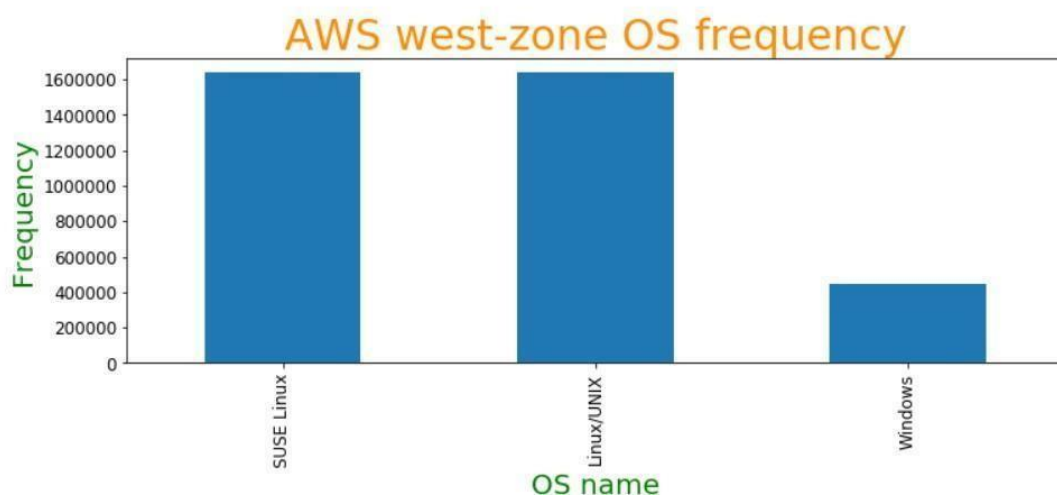
```
] s=data.plot(kind="bar",figsize=(12,4),fontsize=12)  
s.set_title("AWS east-zone OS frequency",color='darkorange',fontsize=30)  
s.set_xlabel("OS name",color='g',fontsize=20)  
s.set_ylabel("Frequency",color='g',fontsize=20)
```

```
] Text(0, 0.5, 'Frequency')
```

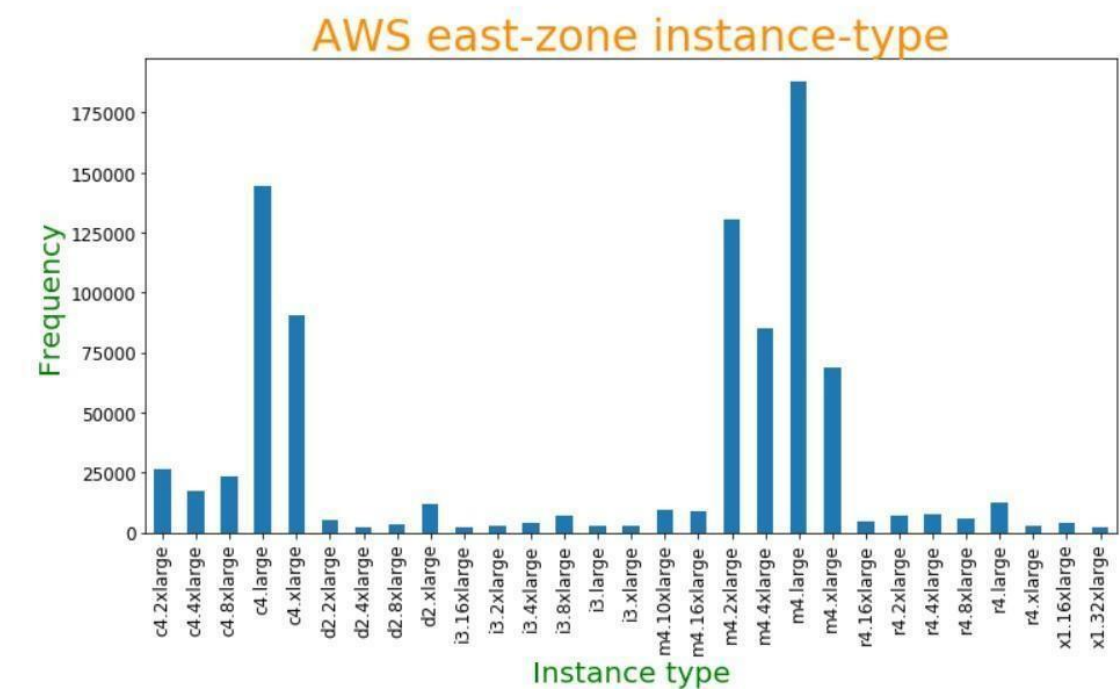


```
: s=west['os'].value_counts().plot(kind="bar",figsize=(12,4),fontsize=12)  
s.set_title("AWS west-zone OS frequency",color='darkorange',fontsize=30)  
s.set_xlabel("OS name",color='g',fontsize=20)  
s.set_ylabel("Frequency",color='g',fontsize=20)
```

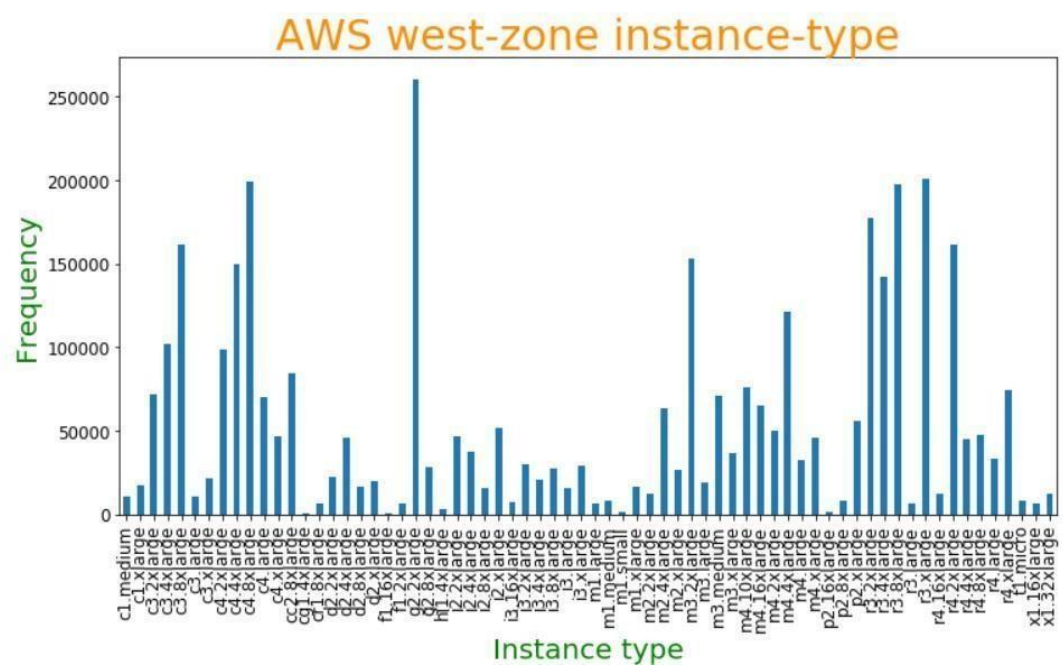
```
: Text(0, 0.5, 'Frequency')
```



```
s=east.groupby('instance_type').size().plot(kind="bar",figsize=(12,6),fontsize=12)
s.set_title("AWS east-zone instance-type",color='darkorange',fontsize=30)
s.set_xlabel("Instance type",color='g',fontsize=20)
s.set_ylabel("Frequency",color='g',fontsize=20)
```



```
s=west.groupby('instance_type').size().plot(kind="bar",figsize=(12,6),fontsize=12)
s.set_title("AWS west-zone instance-type",color='darkorange',fontsize=30)
s.set_xlabel("Instance type",color='g',fontsize=20)
s.set_ylabel("Frequency",color='g',fontsize=20)
```



```
s=east.groupby('os').sum().plot(kind="bar",fontsize=12,color='g',legend=False)
s.set_title("AWS east-zone price per OS",color='darkorange',fontsize=30)
s.set_xlabel("OS name",color='g',fontsize=20)
s.set_ylabel("Price",color='g',fontsize=20)
```

```
Text(0, 0.5, 'Price')
```



```
: s=east.groupby('region').sum().plot(kind="bar",fontsize=12,color='r',legend=False)
s.set_title("AWS east-zone region wise price collection",color='darkorange',fontsize=30)
s.set_xlabel("Region name",color='g',fontsize=20)
s.set_ylabel("Frequency",color='g',fontsize=20)
```

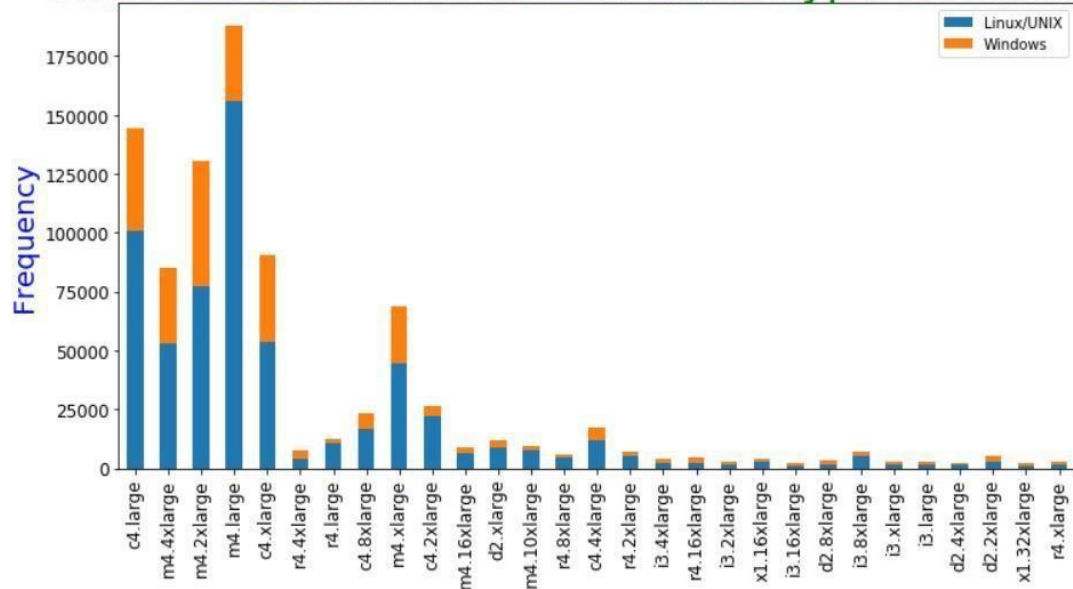
```
: Text(0, 0.5, 'Frequency')
```



```
s=df.plot(kind="bar",stacked=True,figsize=(12,6),fontsize=12)
s.set_title("AWS east-zone: OS wise instance type distribution",color='g',fontsize=30)
s.set_xlabel("Instance type",color='b',fontsize=20)
s.set_ylabel("Frequency",color='b',fontsize=20)
```

Text(0, 0.5, 'Frequency')

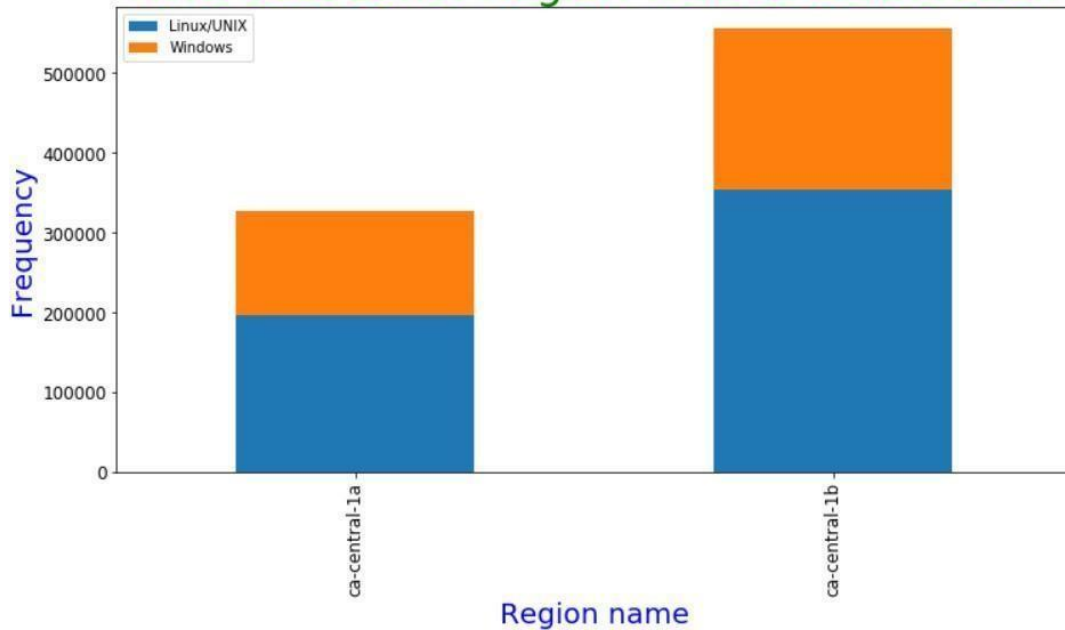
AWS east-zone: OS wise instance type distribution



```
s=df.plot(kind="bar",stacked=True,figsize=(12,6),fontsize=12)
s.set_title("AWS east-zone: Region wise os distribution",color='g',fontsize=30)
s.set_xlabel("Region name",color='b',fontsize=20)
s.set_ylabel("Frequency",color='b',fontsize=20)
```

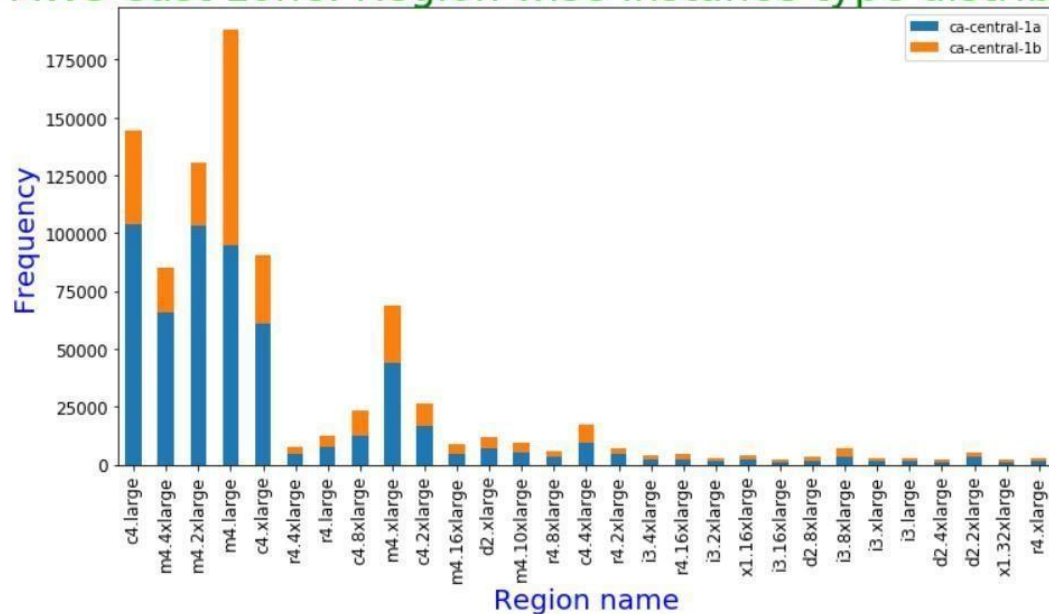
Text(0, 0.5, 'Frequency')

AWS east-zone: Region wise os distribution



```
Text(0, 0.5, 'Frequency')
```

AWS east-zone: Region wise instance type distribution



```
y1_rbf = clf1.predict(X1_test)
```

```
y1_rbf
```

```
array([[ 0.47668365,  0.94883288,  0.52189932, ...,  0.13567177,
        -0.03968387,  0.55823644])
```

```
from sklearn.metrics import mean_squared_error
```

```
print(mean_squared_error(y1_test, y1_rbf))
```

```
1.026702265109996
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
regr1 = RandomForestRegressor(max_depth=2, random_state=0)
```

```
regr1.fit(X1_train, y1_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                        max_depth=2, max_features='auto', max_leaf_nodes=None,
                        max_samples=None, min_impurity_decrease=0.0,
                        min_impurity_split=None, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        n_estimators=100, n_jobs=None, oob_score=False,
                        random_state=0, verbose=0, warm_start=False)
```

```
yregr1=regr1.predict(X1_test)
```

```
yregr1[0], y1_test[0]
```

```
(0.4031265734937783, 0.2804)
```

```
yregr1[1], y1_test[1]
```

```
(0.4031265734937783, 0.7377)
```

```
yregr1[2], y1_test[2]
```

```
(0.4031265734937783, 0.2832)
```

CHAPTER

7

CONCLUSI

ON

CONCLUSION

Is more often used for evaluating patients with suspected acute stroke which is represents high cost and low value endeavor. We have developed an algorithm for helping clinicians determine which patients evaluated for acute stroke need a classifiers. This algorithm has superior diagnostic accuracy in comparison to the algorithm, which is currently the most accurate and widely used algorithm for determining which acute stroke patient needs a classifiers.

Using readily available clinical information, our algorithm is capable of helping clinicians avoid nearly two times the number of acute stroke that can be avoided when the algorithm is utilized.

Clinical implementation of this algorithm can help decrease financial costs associated with evaluations for traumatic brain injury, while decreasing patient exposure to avoidable ionizing radiation. Although the algorithm currently utilizes only easily obtainable clinical data.

FUTURE ENHANCEMENT

In the future, we will explore a dynamic solution to cope with the changing demands of users and providers. We plan to exploit the best-fit auction based policies to suit the new fine-grained pricing scheme in the future

REFERENCES

- [1] H. Zhang, H. Jiang, B. Li, F. Liu, A. V. Vasilakos, and J. Liu, "A Framework for Truthful Online Auctions in Cloud Computing with Heterogeneous User Demands," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 805–818, mar 2016.
- [2] S. Yi, A. Andrzejak, and D. Kondo, "Monetary Cost-Aware Checkpointing and Migration on Amazon Cloud Spot Instances," *IEEE Transactions on Services Computing*, vol. 5, no. 4, pp. 512–524, 2012.
- [3] S. K. Mondal, A. S. Sabyasachi, and J. K. Muppala, "On Boosting Cloud Service Dependability through Optimized Checkpointing," in *15th Int. Symp. Parallel Distrib. Comput.*, 2016.
- [4] H. M. D. Kabir, A. Hosen, S. Nahavandi, and A. Khosravi, "Prediction Interval with Examples of Similar Pattern and Prediction Strength," in the *30th Annual IEEE Canadian Conference on Electrical and Computer Engineering*, CCECE 2017.
- [5] V. K. Singh and K. Dutta, "Dynamic Price Prediction for Amazon Spot Instances," *2015 48th Hawaii International Conference on System Sciences*, Jan. 2015.
- [6] R. Wolski, and J. Brevik. "Providing statistical reliability guarantees in the AWS spot tier." In *Proceedings of the 24th High Performance Computing Symposium*, p. 13. Society for Computer Simulation International, 2016.
- [7] S. Arevalos, F. Lopez-Pires, and B. Baran, "A Comparative Evaluation of Algorithms for Auction-Based Cloud Pricing Prediction," *2016 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 99-108, Apr. 2016.

SOURCE CODE

```
import pandas

as pd import

numpy as np

import matplotlib.pyplot

as plt import seaborn as

sns

from subprocess import check_output

columns=["datetime", "instance_type", "os", "region", "price"]

south = pd.read_csv("ap-south-1.csv",sep=",", header=None,

names=columns) south.info()

south.head()

south.descri

be()

east = pd.read_csv("ca-central-1.csv",sep=",", header=None,

names=columns) east.info()

east.head()

east.descri

be()

west = pd.read_csv("us-east-1.csv",sep=",", header=None,

names=columns) west.info()

west.head()

west.descri
```

```
be()
```

```
central = pd.read_csv("us-west-1.csv",sep="," , header=None,
```

```
names=columns) central.info()
```

```
central.head()
```

```
central.descri
```

```
be()
```

```
data =  
  
east['os'].value_counts()  
  
data  
  
s=data.plot(kind="bar",figsize=(12,4),fontsize=12)  
  
s.set_title("AWS east-zone OS frequency",color='darkorange',fontsize=30)  
  
s.set_xlabel("OS name",color='g',fontsize=20)  
  
s.set_ylabel("Frequency",color='g',fontsize=20)  
  
s=west['os'].value_counts().plot(kind="bar",figsize=(12,4),fontsize=12)  
  
s.set_title("AWS west-zone OS frequency",color='darkorange',fontsize=30)  
  
s.set_xlabel("OS name",color='g',fontsize=20)  
  
s.set_ylabel("Frequency",color='g',fontsize=20)  
  
s=south['os'].value_counts().plot(kind="bar",figsize=(12,4),fontsize=12)  
  
s.set_title("AWS south-zone OS frequency",color='darkorange',fontsize=30)  
  
s.set_xlabel("OS name",color='g',fontsize=20)  
  
s.set_ylabel("Frequency",color='g',fontsize=20)  
  
s=central['os'].value_counts().plot(kind="bar",figsize=(12,4),fontsize=12)  
  
s.set_title("AWS central-zone OS frequency",color='darkorange',fontsize=30)  
  
s.set_xlabel("OS name",color='g',fontsize=20)  
  
s.set_ylabel("Frequency",color='g',fontsize=20)  
  
s=east.groupby('instance_type').size().plot(kind="bar",figsize=(12,6),fontsize=12)  
  
s.set_title("AWS east-zone instance-type",color='darkorange',fontsize=30)  
  
s.set_xlabel("Instance type",color='g',fontsize=20)  
  
s.set_ylabel("Frequency",color='g',fontsize=20)
```

```
s=west.groupby('instance_type').size().plot(kind="bar",figsize=(12,6),fontsize=12  
)
```

```

s.set_title("AWS west-zone instance-type",color='darkorange',fontsize=30)

s.set_xlabel("Instance type",color='g',fontsize=20)

s.set_ylabel("Frequency",color='g',fontsize=20)

s=south.groupby('instance_type').size().plot(kind="bar",figsize=(12,6),fontsize=12)
2) s.set_title("AWS South-zone instance-type",color='darkorange',fontsize=30)

s.set_xlabel("Instance type",color='g',fontsize=20)

s.set_ylabel("Frequency",color='g',fontsize=20)

s=central.groupby('instance_type').size().plot(kind="bar",figsize=(12,6),fontsize=12)
12) s.set_title("AWS Central-zone instance-type",color='darkorange',fontsize=30)
s.set_xlabel("Instance type",color='g',fontsize=20)
s.set_ylabel("Frequency",color='g',fontsize=20)

s=east.groupby('region').sum().plot(kind="bar",fontsize=12,color='r',legend=False)

e)

s.set_title("AWS east-zone regionwise price collection",color='darkorange',fontsize=30)
s.set_xlabel("Region name",color='g',fontsize=20)

s.set_ylabel("Frequency",color='g',fontsize=20)

s=east.groupby('os').sum().plot(kind="bar",fontsize=12,color='g',legend=False)

s.set_title("AWS east-zone price per OS",color='darkorange',fontsize=30)

s.set_xlabel("OS name",color='g',fontsize=20)

s.set_ylabel("Price",color='g',fontsize=20)

d =east["instance_type"].unique()

s1=east.groupby('instance_type')['os'].value_counts()

s1.head()

```

$I = []$

```

for p in d:

    type_per_os =

    s1[p]

    l.append(type_per_os.values)

df =

pd.DataFrame(l,index=d,columns=['Linux/UNIX','Windows'])

s=df.plot(kind="bar",stacked=True,figsize=(12,6),fontsize=12

)

s.set_title("AWS east-zone: OS wise instance type

distribution",color='g',fontsize=30) s.set_xlabel("Instance

type",color='b',fontsize=20) s.set_ylabel("Frequency",color='b',fontsize=20)

d =east["region"].unique()

s1=east.groupby('region')['os'].value_count

s() s1.head()

l=[]

for p in d:

    type_per_os =

    s1[p]

    l.append(type_per_os.values)

df = pd.DataFrame(l,index=d,columns=['Linux/UNIX','Windows'])

s=df.plot(kind="bar",stacked=True,figsize=(12,6),fontsize=12)

s.set_title("AWS east-zone: Region wise os

distribution",color='g',fontsize=30) s.set_xlabel("Region

```

```
name",color='b',fontsize=20)
```

```
s.set_ylabel("Frequency",color='b',fontsize=20)
```

```
d =east["instance_type"].unique()
```

```
s1=east.groupby('instance_type')['region'].value_counts()
```

```
s1.head()
```

```

l=[]

for p in d:

    type_per_os =

    s1[p]

    l.append(type_per_os.values)

df =

pd.DataFrame(l,index=d,columns=['ca-central-1a','ca-central-1b'])

s=df.plot(kind="bar",stacked=True,figsize=(12,6),fontsize=12)

s.set_title("AWS      east-zone:      Region      wise      instance
type distribution",color='g',fontsize=30)

s.set_xlabel("Region

name",color='b',fontsize=20)

s.set_ylabel("Frequency",color='b',fontsize=2

0) central.dropna(inplace=True)

X1 =

central.drop(['price','datetime'],axis=1)

central_df = pd.get_dummies(X1)

X1 = central_df.values

y1 = central['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)
```

```
y1_rbf
```

```
from sklearn.metrics import  
  
mean_squared_error  
  
print(mean_squared_error(y1_test, y1_rbf))  
  
from sklearn.ensemble import RandomForestRegressor  
  
regr1 = RandomForestRegressor(max_depth=2,  
random_state=0) regr1.fit(X1_train, y1_train)  
  
yregr1=regr1.predict(X1_t  
est) yregr1[0], y1_test[0]  
  
yregr1[1], y1_test[1]  
  
yregr1[2], y1_test[2]
```

