

A STRICT VERIFICATION OF SMART CONTRACTS ON BLOCKCHAIN

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

By

PUTA ANIL KUMAR

(Reg. No. 37110606)

PULIGEDDA SAIRAVITEJA

(Reg. No. 37110604)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119. TAMILNADU.

MARCH-2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work P.SAIRAVITEJA (Reg. No.37110604), who carried out the project entitled “**A STRICT VERIFICATION OF SMART CONTRACTS ON BLOCKCHAIN**” under my supervision from Dec 2020 to Mar 2021.

Internal Guide

Dr. G. Mathivanan, B.Sc., B.Tech., M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **PULIGEDDA SAIRAVITEJA (REG.No:37110604)** hereby declare that the Project Report on "**A STRICT VERIFICATION OF SMART CONTRACTS ON BLOCKCHAIN**" done under the guidance of **Dr. G. MATHIVANAN, B.Sc., B.Tech., M.E., Ph.D.**, 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 INSTITUTE OF SCIENCE AND TECHNOLOGY** 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.VIGNESHWARI M.E., Ph.D., and Dr.L.LAKSHMANAN M.E., Ph.D., Heads** of the Department, 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. G . MATHIVANAN , B.SC ., B.Tech., M.E., Ph.D.,** 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

Blockchain is an integrated technology to ensure keeping record and process transactions with decentralized manner. The maturity of this technology including security of the fundamental protocol and its applications is not enough. Due to the immutable nature of distributed ledger technology such as blockchain, it is of utter importance that a smart contract works as intended before employment outside test network. This is since any bugs or errors will become permanent once published to the live network, and could lead to substantial economic losses. In order to avoid this, formal verification methods can be used to ensure that the contract behaves according to given specifications. The proposed system performs formal verification of smart contracts. Being a relatively new research area, a standard or best practice for formal verification of smart contracts has not yet been established. Thus, several different methods and approaches have been used to perform the formal verification. The survey presented in this paper shows that some variant of model checking or theorem proving methodology seems to be most successful. However, as of today, formal verification is only successful on simple contracts, and does not support more advanced smart contract syntax. The proposed system is a novel formal modelling approach to verify a smart contract behavior in its execution environment. We apply this formalism on a concrete smart contract example and analyze its breaches with a statistical model checking approach.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
2	LITERATURE SURVEY	2
3	METHODOLOGY	5
	3.1 MODULES	6
	3.1.1 Hash Pointer	6
	3.1.2 Digital Signature	7
	3.1.3 Vulnerability Scanning	7
	3.2 EXISTING SYSTEM	8
	3.3 Drawbacks Of Existing System	9
	3.4 Proposed System	9
	3.5 Advantages of Proposed System	9
	3.6 Architecture	10
	3.7 SYSTEM REQUIREMENTS	10
	3.7.1 Hardware requirements	10
	3.7.2 Software requirements	10
	3.7.3 Backend Technologies	11
	3.7.4 Frontend Technologies	11
	3.8 OVERVIEW OF THE PLATFORM	11
	3.8.1 Jupyter	12
	3.8.2 Jupyter Notebook	12
	3.8.3 User Interface Description	13

	3.9 EXISTING ALGORITHM	15
	3.10 PROPOSED ALGORITHM	15
	3.11 ADVANTAGES OF PROPOSED ALGORITHM	16
	3.12 XGBOOST MODEL	16
	3.12.3 Implementation	
	3.12.4 Testing	
	3.12.5 Delivery/deployment	
	3.14.6 Maintenance	
	3.13 RAD model	18
4	RESULTS AND DISCUSSION	21
5	CONCLUSION AND FUTURE WORK	24
	5.1 Conclusion	24
	5.2 Future Work	24
	REFERENCES	25
	APPENDIX	27
	A SOURCE CODE	27
	B PAPER WORK	42

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.6	SYSTEM ARCHITECTURE	8
3.8.1	JUPYTER NOTEBOOK	12
3.12	WATERFALL MODEL	15
3.13	RAD MODEL	16
4.1	TOP 10 CRYPTOCURRENCIES IN THE MARKET.	19
4.2	BITCOIN USED,PRICE AND TRADE VOLUME \$ IN YEAR 2017.	19
4.3	ETHEREUM COIN SUPPLY,PRICE AND MARKET CAP IN YEAR 2017.	20
4.4	CREDENTIAL DATASET FOR BITCOIN OF UTC	20
4.5	CREDENTIAL DATASET FOR BITCOIN	21

LIST OF ABBREVIATIONS

ML	-	Machine Learning
IP	-	Internet Protocol
DSDM	-	Dynamic Systems Development Model
XP	-	Extreme Programming
FDD	-	Feature Driven Development
JAD	-	Joint Application Development
LD	-	Lean Development
RAD	-	Rapid Application Development
RUP	-	Rational Unified Process
SDLC	-	Systems Development Life Cycle

CHAPTER 1

INTRODUCTION

Blockchain is an integrated technology to ensure keeping record and process transactions with decentralized manner. The maturity of this technology including security of the fundamental protocol and its applications is not enough. Due to the immutable nature of distributed ledger technology such as blockchain, it is of utter importance that a smart contract works as intended before employment outside test network. This is since any bugs or errors will become permanent once published to the live network, and could lead to substantial economic losses. In order to avoid this, formal verification methods can be used to ensure that the contract behaves according to given specifications.

The proposed system performs formal verification of smart contracts. Being a relatively new research area, a standard or best practice for formal verification of smart contracts has not yet been established. Thus, several different methods and approaches have been used to perform the formal verification. The survey presented in this paper shows that some variant of model checking or theorem proving methodology seems to be most successful. However, as of today, formal verification is only successful on simple contracts, and does not support more advanced smart contract syntax.

The proposed system is a novel formal modelling approach to verify a smart contract behavior in its execution environment. We apply this formalism on a concrete smart contract example and analyze its breaches with a statistical model checking approach.

CHAPTER 2

LITERATURE SURVEY

TITLE: A Blockchain-based key Management Scheme for Named Data Networking

AUTHOR: QICHAO ZHANG, ZHUYUN QI

YEAR: 2018

DESCRIPTION:

Named Data Networking is built with security which requires each named Data object to be digitally signed by its producer. Thus, the NDN project has proposed a key management model on NDN testbed for verification of the Data packet to be immune to distributing poisoned content. However, in practice, this model poses two challenges for verifying fake content: (1) the centralized architecture easily leads to a single point of failure, especially when the root key fails, its difficult to verify the keys across sites due to the lack of trust between them, and (2) excessive overhead of certificate chain traversal when verifying signature. This paper first proposes a blockchainbased key management scheme in NDN to address the problem of lack of mutual trust between sites without trust anchors. Specifically, all site nodes form a permissioned blockchain for storing public key hashes to ensure the authenticity, and the proxy gateway participates in verifying to reduce excessively frequent communication between the router and the blockchain. In addition, the NDN public key content object and the scheme of their storage, verification, and revocation are redesigned. The result of our analysis and evaluation shows that the proposed scheme is capable of supporting less verification numbers and higher verification efficiency.

TITLE: Blockchain-enabled Security-aware Applications in Home Internet of Thing

AUTHOR: GUOLEI ZHANG, JUNLIN XIE

YEAR: 2019

DESCRIPTION:

Data security and privacy protection in the Internet of Things (IoT) have attracted wide attention, especially in private scenarios, such as Home Internet of Things (HIoT). Data decentralization and mature encryption technology based on

blockchain technology provide a proper security guarantee for HIoT. However, common blockchain technology consensus algorithm consumes a lot of computing resources of devices, and it is easy to cause network congestion in LAN due to the size of each transaction and transactions' frequently spread. Just Blockchain Storage Model (JBSM) means every blockchain node in the network stores all of the transaction data that launched in the whole blockchain network, there is no doubt that JBSM consumes a lot of storage resources. In this paper, we propose a Blockchain and Centralized Storage Model (BCSM) for HIoT, which reduces the network bandwidth occupied by transaction data and increases the speed of transaction verification by reducing the size of transaction data. The model consists of three parts: HIoT, Blockchain Network and Cloud Storage Server. This paper analyses how BCSM improves transaction efficiency. Finally, through the simulation experiment on Tencent BaaS, the model is compared with the JBSM, which shows the method in this paper not only has the data security characteristic endowed by the blockchain but also improves the transaction efficiency to a certain extent.

TITLE: How formal analysis and verification add security to blockchain-based system

AUTHOR: SHIN'ICHIRO MATSUO

YEAR: 2017

DESCRIPTION:

Automated and tool-aided formal verification is strong approach to check the correctness of specification and code. Security of the fundamental protocol and its applications is not enough. Blockchain is an integrated technology to ensure keeping record and process transactions with decentralized manner. It is thought as the foundation of future decentralized ecosystem, and collects much attention. However, the maturity of this technology including security of the fundamental protocol and its applications is not enough, thus we need more research on the security evaluation and verification of Blockchain technology This tutorial explains the current status of the security of this technology, its security layers and possibility of application of formal analysis and verification.

TITLE: Joint Transaction Relaying and Block Verification Optimization for
Blockchain Empowered D2D Communication

AUTHOR: LI JIANG, SHENGLI XIE, SABITA MAHARJAN,

YEAR: 2019

DESCRIPTION:

Joint optimization scheme are secure and efficient for data sharing among end users. Take long time to successfully finish the transaction confirmation procedure. Device to device (D2D) communication is a promising paradigm to support data sharing in the fifth generation (5G) small cell networks. However, D2D communication is more vulnerable to security and privacy threats due to direct wireless connection between the proximity devices. In this paper, we exploit blockchain technology to propose a new distributed and secure data sharing framework named D2D blockchain, where we deploy a set of Access Points (APs) to verify end users' transactions. Besides, in order to enable the end users to receive the required data on time in the D2D blockchain, we propose two new schemes for the transaction confirmation procedure including a relay-assisted transaction relaying scheme and a Delegated Proof-of-Stake (DPoS) based lightweight block verification scheme. Furthermore, to incentivize the relay devices to help in relaying transactions in congested areas, and to encourage the verifiers to contribute their resources in block verification, we design a two-stage contract theory based joint optimization scheme. To measure the quality of the relayed transaction and the verified block, we formulate two models: value of transaction relaying (VoTR) and value of block verification (VoBV), and then design the optimal contract for each relay device and each verifier. Security analysis and numerical results illustrate that the proposed D2D blockchain and the designed joint optimization scheme are secure and efficient for data sharing among end users.

TITLE: Nodal Authentication of IoT Data Using Blockchain

AUTHOR: BISMARCK TEI ASARE, KESTER QUIST-APHETSI,

YEAR: 2019

DESCRIPTION:

Achieved a nodal authentication and verification of the transmitted data. Services offered by these websites leading to serious consequences that span

across privacy violations. Pervasive systems over the years continuous to grow exponentially. Engagement of IoT in fields such as Agriculture, Home automation, industrial applications etc., is on the rise. Self-organizing networks within the IoT field give rise to engagement of various nodes for data communication. The rise in Cyber-attacks within IoT pose a lot of threat to these connected nodes and hence there is a need for data passing through nodes to be verified during communication. In this paper we proposed a nodal authentication approach in IoT using blockchain in securing the integrity of data passing through the nodes in IoT. In our work, we engaged the GOST algorithm in our approach. At the end, we achieved a nodal authentication and verification of the transmitted data. This makes it very difficult for an attacker to fake a node in the communication chain of the connected nodes. Data integrity was achieved in the nodes during the communication.

CHAPTER 3

METHODOLOGY

3.1 MODULES

RAFT is distributed crash Fault tolerance consensus algorithm which makes sure that in the event of failure, the system should be able to take a decision and process clients request. In technical term Raft is a consensus algorithm for managing a replicated log. Replicated log is a part of Replicated.

3.1.1 Hash Pointer

Hash pointer is a hash of the data by cryptography, pointing to the location in which the data is stored. Thus, a hash pointer can be used to check whether or not the data has been tampered. A block chain is organized using hash pointers to link data blocks together. With the hash pointer pointing to the predecessor block, each block indicates the address where the data of the predecessor block is stored.

Moreover, the hash of the stored data can be publicly verified by users to prove that the stored data has not been tampered. If an adversary attempts to change data in any block in entire chain, in order to disguise the tampering, the adversary has to change the hash pointers of all previous blocks. Ultimately the adversary has to stop tampering because he will not be able to falsify the data on the head of the chain, which is initially generated once the system has been built. We call this initial opening block in the chain the genesis block. Finally, the adversary's tampering will be uncovered, because by recording this single root hash pointer of genesis block, one can effectively make the entire chain have the property of tamper-resilient. Users are allowed to go back to some special block and verify it from the beginning of the chain.

3.1.2 Digital Signature

A digital signature establishes the validity of a piece of data by using a cryptographic algorithm. It is also a scheme for verifying that a piece of data has not been tampered with. There are three core components that formulate a digital signature scheme. The first component is the key generation algorithm, which creates two keys, one is used to sign messages and be kept privately and called the private

key, and the other is made available to the public, thus called the public key, used to validate whether the message has the signature signed with the corresponding private key. The second core component is the signing algorithm. It produces a signature on the input message endorsed by using the given private key. The third core component is the verification algorithm. It takes a signature, a message, and a public key as inputs, and validates the message's signature using the public key and returns a boolean value. A well-defined and secure signature algorithm should have two properties. The first property is valid signatures must be verifiable. The second property is signatures are existentially unforgeable. It means that an adversary who has your public key cannot forge signatures on some messages with an overwhelming probability.

3.1.3 Vulnerability Scanning

Vulnerabilities caused by contract design defects can also cause huge losses. Vulnerability scanning is the study of vulnerabilities that have been discovered, which aimed at avoiding the same mistakes. It can help us discover potential vulnerabilities in the execution of contracts, which is important to improve the security and credibility of contracts. Using vulnerability scanning method, we can more conveniently and comprehensively detect possible vulnerabilities in contracts when they are executing. So vulnerability scanning method is necessary for developing correct smart contracts. Data privacy is ensured by permissioned settings. User identity is revealed only within a private group of nodes; they are immune to external attacks in contrast to public networks. Fine-grained authorization controls are applied by the operator of the network to implement the security principle of minimal exposure and thus mitigate insider threat attacks. Resource availability is easier to manage, as all network participants and the deployment scenario are known ahead of time.

3.2 EXISTING SYSTEM

Smart contract is a series of symbolic protocols that can be executed by contract participants. Smart contracts usually need to interact with external data sources, but they often fail to get external data over HTTPS. Vulnerabilities caused by contract design defects can also cause huge losses. Vulnerability scanning is the study of vulnerabilities that have been discovered, which aimed at avoiding the same

mistakes. It can help us discover potential vulnerabilities in the execution of contracts, which is important to improve the security and credibility of contracts. In the existing system, the blockchain security provides a secure execution environment. The secure data source provides the credible data to ensure the secure execution. The text agreement of a smart contract made by contract participants needs to fully express the intention. of the makers and comply with the law. Then, implementing formal specification and formal verification on above established contract, which is an iterative process. Through multiple behaviouralmodelling and attribute modelling of smart contracts, we can validate whether the attributes meet the contract requirement or not. The existing system divides the formal verification method of smart contracts into two categories. One is program-based verification and the other one is behaviour-based verification. Program-based verification treats smart contracts as codes, it translates the contract codes into formal languages and then identifies vulnerabilities in contracts. This is a verification method for static analysis programs. Behavior-based verification takes into account issues that may be encountered during the execution of smart contracts, such as improper operations and maliciously attacks. By constructing a formal model, it finds out the weak position of the contract, which is a way to dynamically analyze smart contracts.

3.3 Drawbacks of Existing System

- ✓ Cross site key validation problem: Each site is a relatively independent
- ✓ Due to diversity the data format varied, and the device get damaged.
- ✓ Security of the fundamental protocol and its applications is not enough
- ✓ Take long time to successfully finish the transaction
- ✓ Services offered by these websites leading to serious consequences

3.4 PROPOSED SYSTEM

In the proposed system, the text agreement of a smart contract made by contract participants needs to fully express the intention. of the makers and comply with the law. Then, Implementing formal specification and formal verification on above established contract, which is an iterative process. Through multiple behaviouralmodelling and attribute modelling of smart contracts, we can validate whether the attributes meet the contract requirement or not. In the proposed system,

normal user behaviour and the contract itself, it models a simple blockchain and hacker behaviour. The goal of the hacker is to steal the identity of the user by registering with their own account. Three scenarios are evaluated in order to determine the probability of the user and hacker to successfully register the username. The user has already registered, and the hacker finds the username from the mined block.

In the proposed system, model checking is a method for formal verification that is suitable for systems that can be expressed by a finite-state model. The verification is performed on a computer, using a model checking software. The user provides the model checker with a finite-state model of the system, and a formally specified property it should possess. The model checker then checks if each state of the model satisfies the specification. Every system scenario is examined. If it does not satisfy the property, a counter example is provided, which shows a run of the system that violates the property. This is useful to identify mistakes and correct bugs. On the other hand, if each state of the model does satisfy the specification, the model is formally verified for that specific property.

3.5 Advantages of Proposed System

- ✓ Less verification numbers and higher verification efficiency.
- ✓ Increases the speed of transaction verification
- ✓ Tool-aided formal verification is strong approach to check the correctness of code
- ✓ Joint optimization scheme are secure and efficient
- ✓ Achieved a nodal authentication and verification of the transmitted data

3.6 ARCHITECTURE

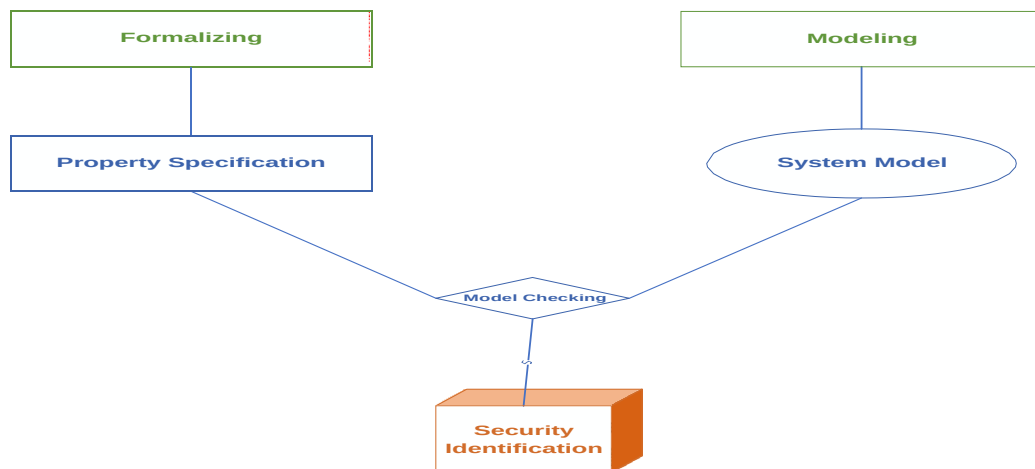


Fig .3.6 System Architecture

3.7 SYSTEM REQUIREMENTS

3.7.1 Hardware requirements:

- Hard disk :120 GB
- Ram : Minimum 2 GB
- Processor : Pentium iv

3.7.2 Software requirements:

- Operating system :Windows 10
- Languages :Python
- IDE :Anaconda

3.7.3 Backend Technologies

- Python
- Numpy
- Sci-learn
- Eclipse IDE

3.7.4 Frontend Technologies

- Web Technologies
- Bootstrap

3.8 OVERVIEW OF THE PLATFORM

3.8.1 Jupyter

Jupyter is an open source project that aims to allow the usage of different programming languages in one computational platform. In fact, its name originates from three of the most popular programming languages for scientific computing: Julia, Python and R. It is built upon the Jupyter Notebook web interface, which brings together the possibilities of executing code, including text along with latex equations (via MathJax), video, and everything that can be visualized within a browser. On the other hand, its features can be extended by means of external tools, an example of which is nbconvert, which can handle conversion to latex and pdf formats. Also available is the possibility to include web presentations based on Reveal.js, or interactive widgets within the document via ipywidgets. Jupyter's development has been supported by companies such as Microsoft or Google, as well as boosted by opendreamkit.horizon 2020 European Research Infrastructure project. In fact, its name originates from three of the most popular programming languages for scientific computing: Julia, Python and R. It is built upon the Jupyter Notebook web interface.

The popularity of Jupyter for educational purposes has grown exponentially due to its flexibility, ease of access through a browser, and the added value brought by the possibility to isolate the document itself from the computational core. This architecture allows to forget about local installation. The lecturer can use an external server (or even their laptop) to supply students with lectures,

slides, or interactive assignments. This feature enables to completely remove the most common obstacle that arises when trying to put forward a new document or program: having the student to install it locally on their personal laptop. This new kind of documents makes it possible to create a rich variety of lectures and supplemental material that can include computing modules. These in turn can be used for self-study, but also for collaborative learning. Eventually, students of science will be faced with some kind of computer program during their studies, as well as in their future workplace. Thus, including this kind of computational elements endows the student with a highly valued transversal skill.

3.8.2 Jupyter Notebook

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. 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. Course Management with Jupyter Notebook: Jupyter Notebook allows to generate interactive documents for students to help them explore and delve into the different aspects considered in a scientific subject. In this sense, they represent a very useful tool for self-study, as well as for the teacher to extend the class explanations to other levels of difficulty. However, managing a course completely or partly based in this kind of documents forces to have a server able to offer access to these interactive documents. Moreover, if exercises in this format are included, we would need also a tool to mark and distribute those marks to all students. we

will consider two possibilities for course management using Jupyter Notebook. The first one is an external server, with the great advantage of possessing already a platform and needs no extra work to be launched. The second one is a local solution instead, in which the teacher configures this Jupyter Notebook server. The advantage in this last case is a full control by the teacher, at the expense of more complexity on its configuration.

3.8.3 User Interface Description

Although the installation process will be described in the next section, let us assume Jupyter Notebook is already installed. In order to launch it, we will type `Jupyter notebook` in a console. Automatically, a new tab in our browser will show up, displaying a list with the contents of the directory where the console was at. This is known as the Jupyter Dashboard. It is possible to install add-ons, such as Nbgrader, a course management and automatic correction system, which will also appear in this control panel. Another important element is the Upload button, at the top-right corner. This button is not particularly interesting if Jupyter Notebook is executed locally, in sharp contrast to the case where it is executed remotely from a different computer and a browser, where it becomes very convenient.

We will click New in order to generate a new Jupyter Notebook. Once we have done it, we can choose the computational kernel among those we have installed. By default, we will only have Python. An example of a Jupyter Notebook can be seen in Figure 3.4.1

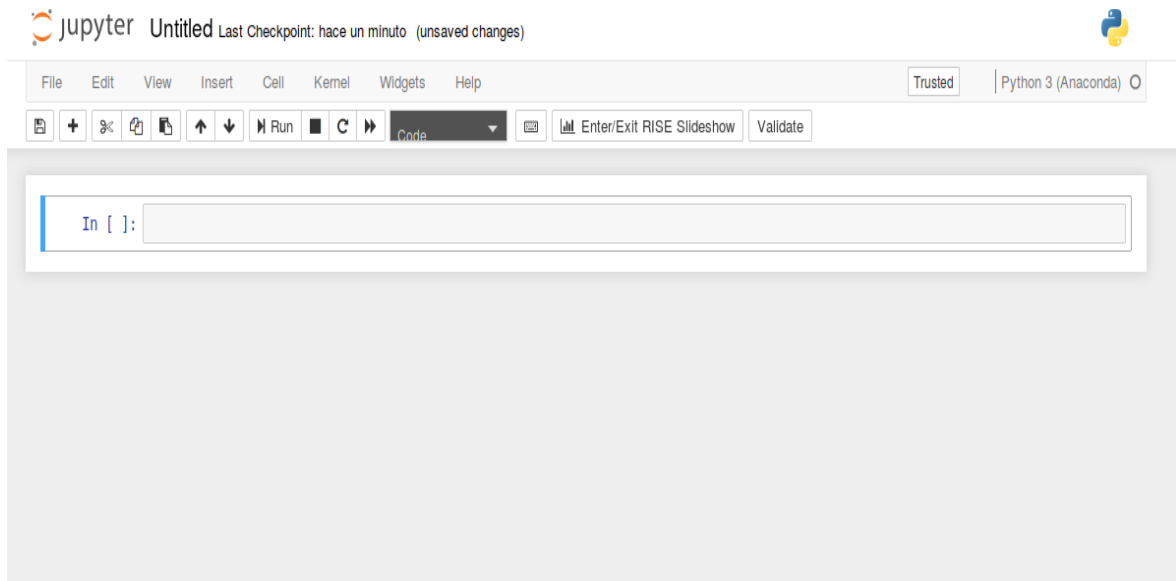


Fig 3.8.1 Jupyter Notebook

There are different menus at our disposal at the top part the document. These allow to edit, insert and delete parts of the document. Additionally, they let us download our document in different formats (e.g. in PDF). Also important is a menu which enables to choose and control the computational kernel (it even allows to stop the kernel if needed). Finally, there is a Help menu with a great deal of information concerning Jupyter Notebook's features. The document itself is divided into separate cells, the behavior of which can be chosen from different options: different size headings, plain or enhanced text by means of the Markdown language, or code. Apart from this options, specific libraries allow to import images or videos, include webpage links or even embed a whole webpage within the document.

In order to insert enhanced text, the Jupyter team decided to go for the Mark- down syntax, a powerful tool to include lists, bold and italics text, tables or images.

There are now myriads of tutorials about this syntax, so we will focus exclusively in the most used formatting options:

- **Bold and italics text:** in order to include bold text, the word must be placed within two asterisks. For instance, ****word**** will output word. For italics, the syntax is very much the same as with bold text, but only an asterisk is needed instead of two. For example, **word** will output word. Alternatively,

in both two cases the asterisks can be substituted by underscores.

- Lists: these are very straightforward. For unordered lists, asterisks play the role of bullets, whereas for ordered lists we will write a number followed by a dot. Markdown takes care of assigning correctly a number to each item.
- Including images: the syntax to include images is! [alternative name](Image location). If the image cannot be loaded, the alternative name will appear instead. The image location can be either that of a local image or a link to a webpage. If the image cannot be loaded, the alternative name will appear instead.
- Including HTML code: Markdown is a subcategory of HTML. Thus, if further control of format is needed, it can be directly included in HTML.
- Links: text cells can include links, both to other parts of the document and webpages, or even to local files. The syntax is [text](link).
- Mathematical formulae: thanks to MathJax, LaTeX code can be included to show all types of mathematical formulae and expressions. Equations in line with the text are written in between dollar signs, as in (\$...\$), whereas equations within two paragraphs use two dollar signs instead, that is, (\$\$...\$\$).

All these features will let us use text cells a in a versatile way to document code from other cells. For instance, it can be used to explain some theoretical concept in detail, which can be followed by an exercise related to it. When we create a new cell, it automatically corresponds to a code cell. The programming language will be determined by the kernel used for that document.

3.9 EXISTING ALGORITHM

- Vulnerability Scanning Method

3.10 PROPOSED ALGORITHM

- Raft Consensus Algorithm

3.11 ADVANTAGES OF PROPOSED ALGORITHM

- It's highly scalable, making it suitable for a variety of applications.
- Higher speed – Transactions are processed faster compared to PoW .
- Lesser energy consumption as there is no need for supercomputers

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

3.12 XGBOOST MODEL

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks.

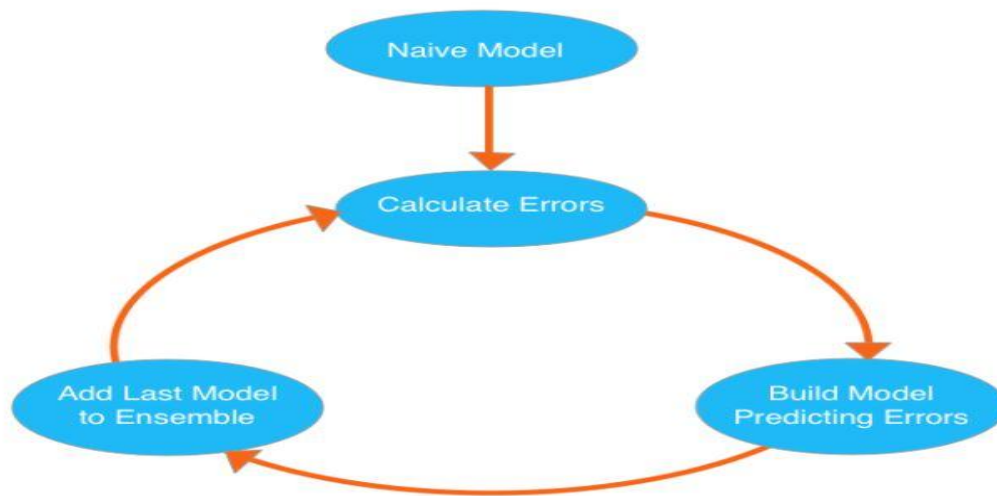


Fig.3.12 XGBoost Model

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.

2. 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.

3. 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.

4. 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.

5. Delivery/deployment

In this phase, the solution is complete, and your project team submits the

deliverables to be deployed or released.

6. 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.

3.13 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.

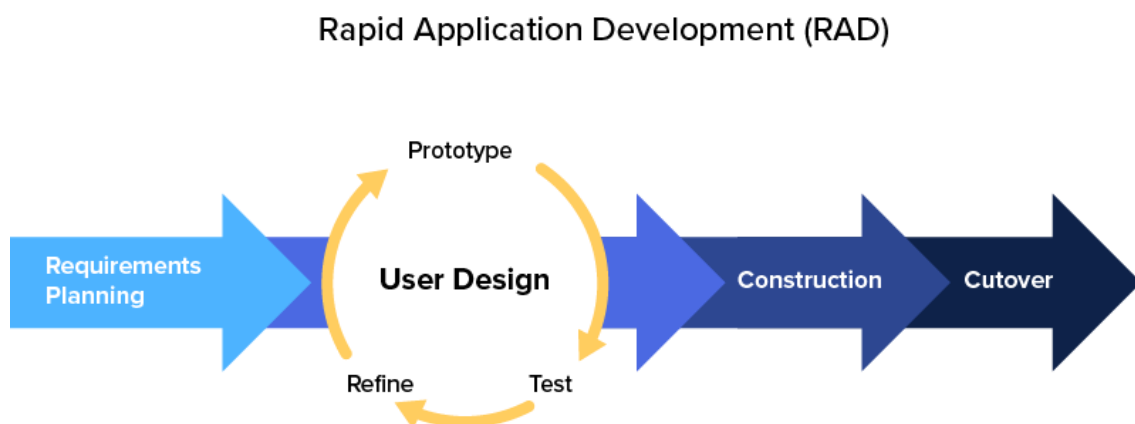


Fig :3.13 RAD Model

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.

CHAPTER 4

RESULTS AND DISCUSSION

The below figure describes about the graph related to the top 10 cryptocurrencies in the market. Where the x-axis denotes recall and y-axis denotes precision.

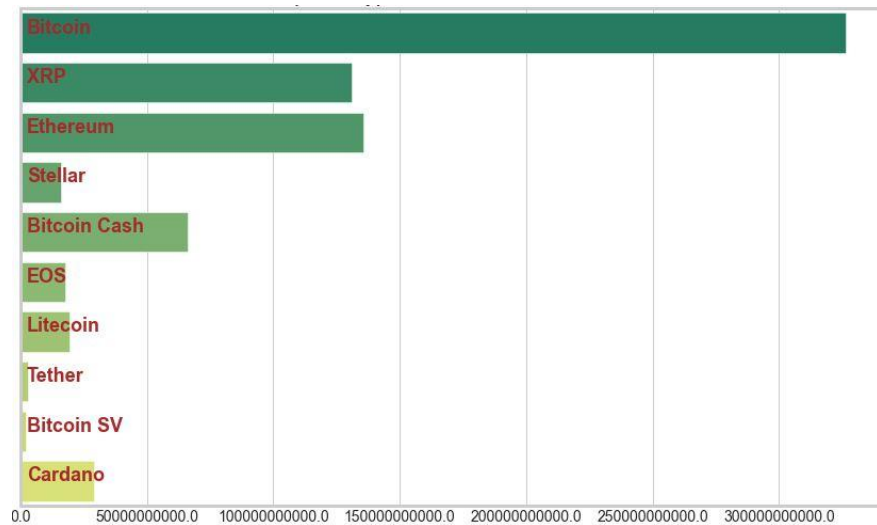


Fig. 4.1 TOP 10 CRYPTOCURRENCIES IN THE MARKET.

It allows the visualization of the performance of an algorithm is done by confusion matrix. Without normalization and it predict bitcoin used ,price and trade volume \$ in year 2017.



Fig. 4.2 BITCOIN USED,PRICE AND TRADE VOLUME \$ IN YEAR 2017 .

The graph of the ethereum coin. It show the complete data in graph .Which is train data and tested data, Input and output of there category.

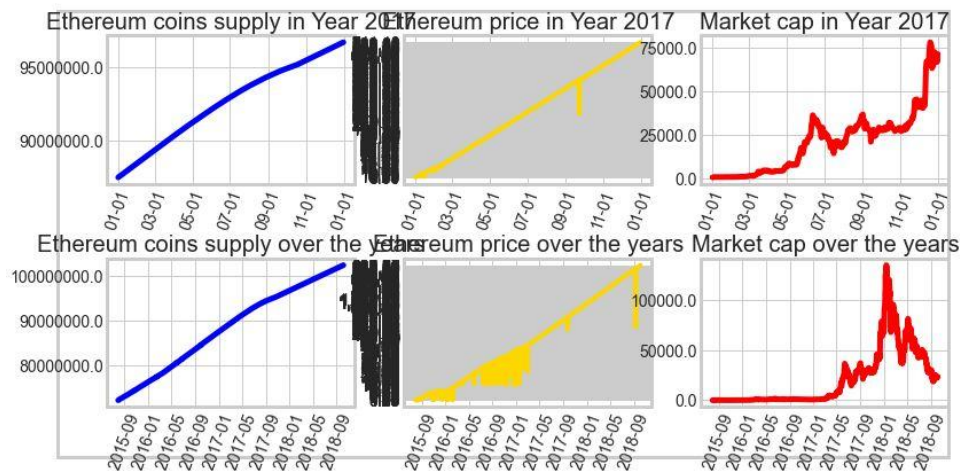


Fig. 4.3 ETHEREUM COIN SUPPLY,PRICE AND MARKET CAP IN YEAR 2017.

The analysed data from confusion matrix taken from test and predict for Bitcoin of UTC

	Date(UTC)	UnixTimeStamp	Supply	MarketCap	Price
1159	10/1/2018	2018-10-01 05:30:00	1.023094e+08	23622.225411	230.89
1160	10/2/2018	2018-10-02 05:30:00	1.023299e+08	23066.193811	225.41
1161	10/3/2018	2018-10-03 05:30:00	1.023502e+08	22513.983283	219.97
1162	10/4/2018	2018-10-04 05:30:00	1.023706e+08	22701.707388	221.76
1163	10/5/2018	2018-10-05 05:30:00	1.023908e+08	23334.861312	227.90

Fig. 4.4 CREDENTIAL DATASET FOR BITCOIN OF UTC.

The analysed data from confusion matrix taken from test and predict

for Bitcoin.

	slug	symbol	name	date	ranknow	open	high	low	close	volume	market	close_ratio	spread
0	bitcoin	BTC	Bitcoin	2013-04-28	1	135.30	135.98	132.10	134.21	0.0	1.488567e+09	0.5438	3.88
1	bitcoin	BTC	Bitcoin	2013-04-29	1	134.44	147.49	134.00	144.54	0.0	1.603769e+09	0.7813	13.49
2	bitcoin	BTC	Bitcoin	2013-04-30	1	144.00	146.93	134.05	139.00	0.0	1.542813e+09	0.3843	12.88
3	bitcoin	BTC	Bitcoin	2013-05-01	1	139.00	139.89	107.72	116.99	0.0	1.298955e+09	0.2882	32.17
4	bitcoin	BTC	Bitcoin	2013-05-02	1	116.38	125.60	92.28	105.21	0.0	1.168517e+09	0.3881	33.32

Fig. 4.5 CREDENTIAL DATASET FOR BITCOIN.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The proposed framework, formal confirmation of savvy contracts running on circulated records, for example, blockchain.

Currently, the methodologies can just deal with basic brilliant agreements and streamlined models, and are not reasonable for complex agreements. The proposed framework encourages the use of formal investigation and formal check by considering innovation layers and their security concerns. We picked three layers, execution, convention and language, as focuses of uses of formal investigation. It proposes a system to apply formal examination to each layer by utilizing existing benchmarks and results.

5.2 FUTURE WORK

Let's take a look at a typical transaction verification process. Suppose a sender wants to send a message. The sender will pass the message through the hash function and generate a hash value of the message. After the hash value has been created, it is passed through a signature algorithm, and with the private key, a digitally signed document is created.

The original message, the digitally signed document, and the public key are then transmitted to the receiver. At the receiver's end, the transaction message is passed through a hash function to get a hash value, and that hash value is compared with the hash value obtained bypassing the digital signature and public key through a verification function.

The hash function creates a unique digital fingerprint of data. The message is passed through the hashing function, and it generates a hash value. This hash value is called a digital print, and it has a unique property: Any hashing function is a one-way function; it cannot be reversed. You cannot decode the original value from the hashed value.

REFERENCES

- D. Vujičić, D. Jagodić, and S. Randić, “Blockchain technology, bitcoin, and Ethereum: A brief overview,” in Proc. 17th Int. Symp. INFOTEHJAHORINA (INFOTEH), Mar. 2018, pp. 21–23.
- V. Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Accessed: May. 18, 2019. [Online]. Available: <https://github.com/ethereum/wiki/wiki/White-Paper>
- K. Finley. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Accessed: May. 18, 2019. [Online]. Available: <https://www.wired.com/2016/06/50-million-hack-just-showed-daohuman/>
- S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. Accessed: May. 18, 2019. [Online]. Available: <https://bitcoin.org/bitcoin.pdf>
- Z. Zheng, S. Xie, H. Dai, and H. Wang, “An overview of blockchain technology: Architecture, consensus, and future trends,” in Proc. IEEE Int. Congr. Big Data, Big Data Congr., Honolulu, HI, USA, Jun. 2017, pp. 557–564
- N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts (SoK),” in Proc. Int. Conf. Princ. Secur. Trust, Apr. 2017, pp. 164–186. doi: 10.1007/978-3-662-54455-6_8.
- M. Alharby and A. V. Moorsel, “Blockchain-based smart contracts: A systematic mapping study,” in Proc. Int. Conf. Artif. Intell. Soft Comput., Aug. 2017, pp. 125–140. doi: 10.5121/csit.2017.71011.
- X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” Future Gener. Comput. Syst., pp. 1–25, Aug. 2017. doi: 10.1016/j.future.2017.08.020.

- F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, “Town crier: An authenticated data feed for smart contracts,” in Proc. ACM Conf. Comput. Commun. Secur., Oct. 2016, pp. 1–20. doi: 10.1145/2976749.2978326.
- L. Anderson, R. Holz, A. Ponomarev, P. Rimba, and I. Weber, “New kids on the block: An analysis of modern blockchains,” Jun. 2016, arXiv:1606.06530. [Online]. Available: <https://arxiv.org/abs/1606.06530>
- G. Bissias, B. N. Levine, and N. Kapadia, “Market-based security for distributed applications,” in Proc. New Secur. Paradigms Workshop, Oct. 2017, pp. 19–34. doi: 10.1145/3171533.3171541.
- Y. Marcus, E. Heilman, and S. Goldberg. Low-Resource Eclipse Attacks on Ethereum’s Peer-to-Peer Network. Accessed: May. 18, 2019. [Online]. Available: <https://eprint.iacr.org/2018/236.pdf>
- C. Liu, H. Liu, Z. Cao, Z. Chen, B. Chen, and B. Roscoe, “ReGuard: Finding reentrancy bugs in smart contracts,” in Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. Companion, May 2018, pp. 65–68
- T. C. Le, L. Xu, L. Chen, and W. Shi, “Proving Conditional Termination for Smart Contracts.” in Proc. 2nd ACM Workshop Blockchains, Cryptocurrencies, Contracts, Jun. 2018, pp. 57–59
- C. F. Torres, J. Schütte, and R. State, “Osiris: Hunting for integer bugs in ethereum smart contracts,” in Proc. 34th Annu. Comput. Secur. Appl. Conf., Dec. 2018, pp. 19–34. doi: 10.1145/3274694.3274737.
- G. Shelly, A. Ittai, G.-G. Guy, M. Yan, R. Noam, S. Mooly, and Z. Yoni, “Online detection of effectively callback free objects with applications to smart contracts,” in Proc. ACM Program. Lang., Aug. 2018, pp. 1–28.

APPENDIX

A.SOURCE CODE

```
import xgboost as xgb
from sklearn.model_selection import cross_validate
from sklearn.metrics import mean_squared_error
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib
import datetime as dt
import matplotlib.dates as mdates
plt.style.use('fivethirtyeight')
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
import random as rnd
import re
import io
import datetime as dt
from sklearn.metrics import confusion_matrix
import matplotlib.gridspec as gridspec
from sklearn.preprocessing import StandardScaler
from numpy import genfromtxt
from scipy.stats import multivariate_normal
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score, average_precision_score
from sklearn.metrics import precision_score, precision_recall_curve
from mpl_toolkits.basemap import Basemap
from matplotlib import animation, rc
import base64
from IPython.display import HTML, display
from matplotlib.pyplot import imread
import codecs
import matplotlib.dates as mdates
from mpl_finance import candlestick_ohlc
import warnings
warnings.filterwarnings('ignore')
np.set_printoptions(suppress=True)
sns.set_palette("dark")
sns.set_style("whitegrid")
```

```

def readabledate (unixtime):
    return dt.datetime.fromtimestamp(float(unixtime))
Bitcoin = pd.read_csv('bitstampUSD_1-min_data_2012-01-01_to_2020-12-
31.csv',parse_dates=[0], date_parser=readabledate)
EthereumPrice = pd.read_csv('EtherMarketCapChart.csv',parse_dates=[1],
date_parser=readabledate)
EthereumAddress = pd.read_csv('EthereumUniqueAddressGrowthRate.csv',parse_dates=[1],
date_parser=readabledate)
Allcrypto = pd.read_csv('crypto-markets.csv')
name = Allcrypto['name'].unique()
Feature = []
currency = []
marketval = []
v_features = name[:10]
for i, cn in enumerate(v_features):
    Feature.append(str(cn))
    filtered = Allcrypto[(Allcrypto['name']==str(cn))]
    temp = filtered[filtered['market'] == filtered['market'].max()]['name'].values
    temp1 = temp[0]
    tempval = filtered['market'].max()
    currency.append(temp1)
    marketval.append(tempval)
f, ax = plt.subplots(figsize=(13, 8))
g = sns.barplot( y = Feature,
                x = marketval,
                palette="summer")
plt.title("Top 10 Cryptocurrencies in the market")
ax.set_xticklabels(ax.get_xticks())
ax.get_yaxis().set_visible(False)
for i, v in enumerate(currency):
    ax.text(2500000000, i, v,fontsize=18,color='brown',weight='bold')
fig=plt.gcf()
plt.show()
years = np.unique(Bitcoin['Timestamp'].dt.year)
mean_open = []
mean_volume = []
mean_close = []
mean_high = []
mean_low = []
mean_BTC = []
mean_average = []
for year in years:
    mean_volume.append(Bitcoin[Bitcoin['Timestamp'].dt.year ==
year]['Volume_(Currency)'].mean())
    mean_BTC.append(Bitcoin[Bitcoin['Timestamp'].dt.year == year]['Volume_(BTC)'].mean())
    mean_open.append(Bitcoin[Bitcoin['Timestamp'].dt.year == year]['Open'].mean())

```

```

    mean_close.append(Bitcoin[Bitcoin['Timestamp'].dt.year == year]['Close'].mean())
    mean_high.append(Bitcoin[Bitcoin['Timestamp'].dt.year == year]['High'].mean())
    mean_low.append(Bitcoin[Bitcoin['Timestamp'].dt.year == year]['Low'].mean())
    mean_average.append(Bitcoin[Bitcoin['Timestamp'].dt.year ==
year]['Weighted_Price'].mean())
trace0 = go.Scatter(
    x = years,
    y = mean_average,
    #fill='tonexty',
    mode='lines',
    name='Weighted_Price of 1 Bitcoin',
    line=dict(
        color='rgb(199, 121, 093)',
    )
)

data = [trace0]

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='Bitcoin price in $'),
    title=' Bitcoin value over the years ',
    showlegend = True)

fig = go.Figure(data=data, layout=layout)
iplot(fig)
Months = {
1 : 'January',
2 : 'February',
3 : 'March',
4 : 'April',
5 : 'May',
6 : 'June',
7 : 'July',
8 : 'August',
9 : 'September',
10 : 'October',
11 : 'November',
12 : 'December',
}

Traces = {
1 : 'trace0',
2 : 'trace1',
3 : 'trace2',
4 : 'trace3',

```

```

5 : 'trace4',
6 : 'trace5',
7 : 'trace6',
8 : 'trace7',
9 : 'trace8',
10 : 'trace9',
11 : 'trace10',
12 : 'trace11',
}

DF_Bitcoin_price = {
1 : 'Bitcoin_price1',
2 : 'Bitcoin_price2',
3 : 'Bitcoin_price3',
4 : 'Bitcoin_price4',
5 : 'Bitcoin_price5',
6 : 'Bitcoin_price6',
7 : 'Bitcoin_price7',
8 : 'Bitcoin_price8',
9 : 'Bitcoin_price9',
10 : 'Bitcoin_price10',
11 : 'Bitcoin_price11',
12 : 'Bitcoin_price12'
}
Bitcoin['Century'] = Bitcoin['Timestamp'].dt.year
j = 1
data = []
for i in range(12):
    Bitcoin_month = Bitcoin[Bitcoin['Timestamp'].dt.month == j]
    DF_Bitcoin_price[j] = Bitcoin_month.groupby(['Century'])['Weighted_Price'].mean()
    Traces[j] = go.Scatter(
        x = DF_Bitcoin_price[j].index,
        y = DF_Bitcoin_price[j].values,
        mode = 'lines',
        name = Months[j]
    )
    data.append(Traces[j])
    j = j + 1

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='Bitcoin Price$$'),
    title=('Monthly distribution of Bitcoin prices'))
fig = go.Figure(data=data, layout=layout)
iplot(fig)
years = np.unique(Bitcoin['Timestamp'].dt.year)

```

```

sum_coins = []
sum_volume = []
for year in years:
    sum_volume.append(Bitcoin[Bitcoin['Timestamp'].dt.year ==
year][['Volume_(Currency)']].sum())
    sum_coins.append(Bitcoin[Bitcoin['Timestamp'].dt.year == year][['Volume_(BTC)']].sum())
trace0 = go.Scatter(
    x = years,
    y = sum_volume,
    fill= None,
    mode='lines',
    name='Trade volume in $ yearwise',
    line=dict(
        color='rgb(0, 255, 255)',
    )
)
data = [trace0]

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='Trade volume in $'),
    title='Trade volume in $ yearwise',
    showlegend = True)

fig = go.Figure(data=data, layout=layout)
iplot(fig)
trace0 = go.Scatter(
    x = years,
    y = sum_coins,
    mode='lines',
    name='Bitcoin mined /used for trading yearwise',
    line=dict(
        color='rgb(199, 121, 093)',
    )
)
data = [trace0]

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='Bitcoins'),
    title='Bitcoins mined /used for trading yearwise',
    showlegend = True)

fig = go.Figure(data=data, layout=layout)
iplot(fig)

```

```

Bitcoin_2017 = Bitcoin[Bitcoin['Timestamp'].dt.year == 2017]
months =
Bitcoin_2017.groupby(Bitcoin_2017['Timestamp'].dt.month)['Volume_(BTC)'].sum().keys()
months = np.array(months)
fig = plt.figure(figsize=(12,6))
axes1 = fig.add_axes([0.1,0.1,0.9,0.9])
axes2 = fig.add_axes([0.15,0.65,0.2,0.3])
axes3 = fig.add_axes([0.45,0.65,0.2,0.3])
axes4 = fig.add_axes([0.75,0.65,0.2,0.3])
axes5 = fig.add_axes([0.15,0.2,0.2,0.3])
axes6 = fig.add_axes([0.45,0.2,0.2,0.3])
axes7 = fig.add_axes([0.75,0.2,0.2,0.3])

axes1.set_title("")

axes2.plot(months,
Bitcoin_2017.groupby(Bitcoin_2017['Timestamp'].dt.month)['Volume_(BTC)'].sum().values,
color="Blue", lw=5);
axes2.set_title('bitcoins mined/used in Year 2017')

axes3.plot(Bitcoin_2017['Timestamp'].dt.month,Bitcoin_2017['Weighted_Price'],color='Gold')
axes3.set_title('bitcoin prices in Year 2017')

axes4.plot(months,
Bitcoin_2017.groupby(Bitcoin_2017['Timestamp'].dt.month)['Volume_(Currency)'].sum().values,
color="Red", lw=5);
axes4.set_title('Trade volume $ in Year 2017');
axes4.set_yticklabels(axes4.get_yticks());

axes5.plot(years,sum_coins , color="Blue", lw=5);
axes5.set_title('bitcoins mined/used over the years')
axes5.set_yticklabels(axes5.get_yticks())

axes6.plot(years,mean_average,color='Gold')
axes6.set_title('bitcoin prices over the years')

axes7.plot(years,sum_volume , color="Red", lw=5);
axes7.set_title('Trade volume over the years');
axes7.set_yticklabels(axes7.get_yticks());

axes1.set_xticks([])
axes1.set_yticks([]);
x = EthereumPrice['UnixTimeStamp'],
y = EthereumPrice['Price'],
mode='lines',
name='Price of Ethereum',

```

```

        line=dict(
            color='rgb(199, 121, 093)',
        )
    )

data = [trace0]

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='Ethereum price in $'),
    title=' Ethereum price over the years ',
    showlegend = True)

fig = go.Figure(data=data, layout=layout)
iplot(fig)
trace0 = go.Scatter(
    x = EthereumPrice['UnixTimeStamp'],
    y = EthereumPrice['Supply'],
    mode='lines',
    name='supply of Ethereum coins',
    line=dict(
        color='rgb(199, 121, 093)',
    )
)

data = [trace0]

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='Supply of Ethereum coins'),
    title='supply of Ethereum coins over the years ',
    showlegend = True)

fig = go.Figure(data=data, layout=layout)
iplot(fig)
trace0 = go.Scatter(
    x = EthereumPrice['UnixTimeStamp'],
    y = EthereumPrice['MarketCap'],
    mode='lines',
    name='MarketCap in Million',
    line=dict(
        color='rgb(199, 121, 093)',
    )
)

data = [trace0]

```

```

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='MarketCap in Million $'),
    title='MarketCap of Ethereum over the years ',
    showlegend = True)

```

```

fig = go.Figure(data=data, layout=layout)
iplot(fig)
trace0 = go.Scatter(
    x = EthereumAddress['UnixTimeStamp'],
    y = EthereumAddress['Value'],
    mode='lines',
    name='No of Address',
    line=dict(
        color='rgb(199, 121, 093)',
    )
)

```

```

data = [trace0]

```

```

layout = go.Layout(
    xaxis=dict(title='year'),
    yaxis=dict(title='No of Address'),
    title='No of growing Addresses over the years ',
    showlegend = True)

```

```

fig = go.Figure(data=data, layout=layout)
iplot(fig)
hfmt = mdates.DateFormatter('%Y-%m')
hfmt2017 = mdates.DateFormatter('%m-%d')
EthereumPrice_2017 = EthereumPrice[EthereumPrice['UnixTimeStamp'].dt.year == 2017]

```

```

fig = plt.figure(figsize=(12,6))
axes1 = fig.add_axes([0.1,0.1,0.9,0.9])
axes2 = fig.add_axes([0.15,0.65,0.25,0.3])
axes3 = fig.add_axes([0.45,0.65,0.25,0.3])
axes4 = fig.add_axes([0.75,0.65,0.25,0.3])
axes5 = fig.add_axes([0.15,0.2,0.25,0.3])
axes6 = fig.add_axes([0.45,0.2,0.25,0.3])
axes7 = fig.add_axes([0.75,0.2,0.25,0.3])
axes1.set_title("")

```

```

axes2.plot(EthereumPrice_2017['UnixTimeStamp'],EthereumPrice_2017['Supply'] , color="Blue",
lw=5);
axes2.set_title('Ethereum coins supply in Year 2017')

```

```

axes2.set_yticklabels(axes2.get_yticks())
axes2.set_xticklabels(axes2.get_xticks(),rotation=70)
axes2.xaxis.set_major_formatter(hfmt2017)

axes3.plot(EthereumPrice_2017['UnixTimeStamp'],EthereumPrice_2017['Price'],color='Gold')
axes3.set_xticklabels(axes3.get_xticks(),rotation=70)
axes3.xaxis.set_major_formatter(hfmt2017)
axes3.set_title('Ethereum price in Year 2017')

axes4.plot(EthereumPrice_2017['UnixTimeStamp'],EthereumPrice_2017['MarketCap'],
color="Red", lw=5);
axes4.set_title('Market cap in Year 2017');
axes4.set_xticklabels(axes4.get_xticks(),rotation=70);
axes4.xaxis.set_major_formatter(hfmt2017)
axes4.set_yticklabels(axes4.get_yticks());

axes5.plot(EthereumPrice['UnixTimeStamp'],EthereumPrice['Supply'], color="Blue", lw=5);
axes5.set_title('Ethereum coins supply over the years')
axes5.set_yticklabels(axes5.get_yticks())
axes5.set_xticklabels(axes5.get_xticks(),rotation=70)
axes5.xaxis.set_major_formatter(hfmt)

axes6.plot(EthereumPrice['UnixTimeStamp'],EthereumPrice['Price'],color='Gold')
axes6.set_title('Ethereum price over the years')
axes6.set_xticklabels(axes6.get_xticks(),rotation=70)
axes6.xaxis.set_major_formatter(hfmt)

axes7.plot(EthereumPrice['UnixTimeStamp'],EthereumPrice['MarketCap'], color="Red", lw=5);
axes7.set_title('Market cap over the years');
axes7.set_xticklabels(axes7.get_xticks(),rotation=70)
axes7.xaxis.set_major_formatter(hfmt)
axes7.set_yticklabels(axes7.get_yticks());

axes1.set_xticks([])
axes1.set_yticks([]);
Bitcoin = Allcrypto[Allcrypto['ranknow'] == 1]
Ethereum = Allcrypto[Allcrypto['ranknow'] == 2]
Ripple = Allcrypto[Allcrypto['ranknow'] == 3]
BitcoinOHLC = Bitcoin[['date','open','high','low','close']]
EthereumOHLC = Ethereum[['date','open','high','low','close']]
RippleOHLC = Ripple[['date','open','high','low','close']]
BitcoinOHLC['date'] = pd.to_datetime(BitcoinOHLC['date'])
BitcoinOHLC['date'] = mdates.date2num(BitcoinOHLC['date'])
RippleOHLC['date'] = pd.to_datetime(RippleOHLC['date'])
RippleOHLC['date'] = mdates.date2num(RippleOHLC['date'])
EthereumOHLC['date'] = pd.to_datetime(EthereumOHLC['date'])

```

```

EthereumOHLC['date'] = mdates.date2num(EthereumOHLC['date'])
f,ax=plt.subplots(figsize=(15,11))
ax.xaxis_date()
plt.xlabel("Date")
candlestick_ohlc(ax,BitcoinOHLC.values,width=5, colorup='g', colordown='r',alpha=0.75)
ax.set_xticklabels(ax.get_xticks(),rotation=70)
ax.xaxis.set_major_formatter(hfmt)
plt.ylabel("Price")
plt.legend()
plt.show()
f,ax=plt.subplots(figsize=(15,11))
ax.xaxis_date()
plt.xlabel("Date")
candlestick_ohlc(ax,EthereumOHLC.values,width=5, colorup='g', colordown='r',alpha=0.75)
ax.set_xticklabels(ax.get_xticks(),rotation=70)
ax.xaxis.set_major_formatter(hfmt)
plt.ylabel("Price")
plt.legend()
plt.show()
data.dtypes
timestamp          float64
total_addresses    float64
blocksize          float64
price_USD          float64
hashrate           float64
total_eth_growth   float64
market-cap-value   float64
transactions       float64
dtype: object
data['date'] = data.apply(lambda x: dt.datetime.fromtimestamp(x['timestamp']), axis = 1)
data['year'] = data.apply(lambda x: dt.datetime.fromtimestamp(x['timestamp']).year, axis = 1)
plt.subplots(figsize=(15, 9))
plt.plot(data['date'], data['transactions'])
plt.xlabel("Date", fontsize = 15)
plt.ylabel("Number of Transactions", fontsize = 15)
plt.subplots(figsize=(15, 9))
plt.plot(data['date'], data['price_USD'])
plt.xlabel("Date", fontsize = 15)
plt.ylabel("Price_USD", fontsize = 15)
plt.subplots(figsize = (8, 8))
correlation_object = data.iloc[:, 1:8].corr()
sns.heatmap(correlation_object)
testing_data = data.loc[data['year'] == 2017]
training_data = data.loc[data['year'] != 2017]
del data
def remove_marketcapvalue():

```

```

global testing_data, training_data
testing_data.drop('market-cap-value', axis = 1, inplace = True)
training_data.drop('market-cap-value', axis = 1, inplace = True)
remove_marketcapvalue()
training_data.head()
plt.subplots(figsize=(15, 9))
plt.plot(training_data['date'], training_data['price_USD'])
plt.xlabel("Date", fontsize = 15)
plt.ylabel("Price_USD", fontsize = 15)
def prepare_modelling():
    global training_data, testing_data
    concat_data = pd.concat([training_data, testing_data])
    target = concat_data['price_USD']
    concat_data.drop(['timestamp', 'price_USD'], axis = 1, inplace = True)

    train_index = concat_data[concat_data['year'] != 2017].index.tolist()
    test_index = list(set(train_index).symmetric_difference(set(range(len(concat_data)))))

    def dummified_data(dataframe):
        dataframe = pd.concat([pd.get_dummies(dataframe['year'], prefix= 'year',
        prefix_sep='_').reset_index(drop = True), dataframe.reset_index(drop = True)], axis = 1)
        dataframe = pd.concat([pd.get_dummies(dataframe['month'], prefix='month',
        prefix_sep='_').reset_index(drop = True), dataframe.reset_index(drop = True)], axis = 1)
        dataframe = pd.concat([pd.get_dummies(dataframe['day'], prefix='day',
        prefix_sep='_').reset_index(drop = True), dataframe.reset_index(drop = True)], axis = 1)
        dataframe = pd.concat([pd.get_dummies(dataframe['week'], prefix='week',
        prefix_sep='_').reset_index(drop = True), dataframe.reset_index(drop = True)], axis = 1)

        r_cols = ['year', 'month', 'day', 'week']
        dataframe.drop(r_cols, axis = 1, inplace = True)

        return(dataframe)
    concat_data = dummified_data(concat_data)

    train_X = concat_data.loc[train_index]
    train_Y = target[train_index]

    test_X = concat_data.loc[test_index]
    test_Y = target[test_index]

    return(train_X, train_Y, test_X, test_Y)

train_X, train_Y, test_X, test_Y = prepare_modelling()
import sklearn
xgbfolds = sklearn.model_selection.KFold(n_splits=5)

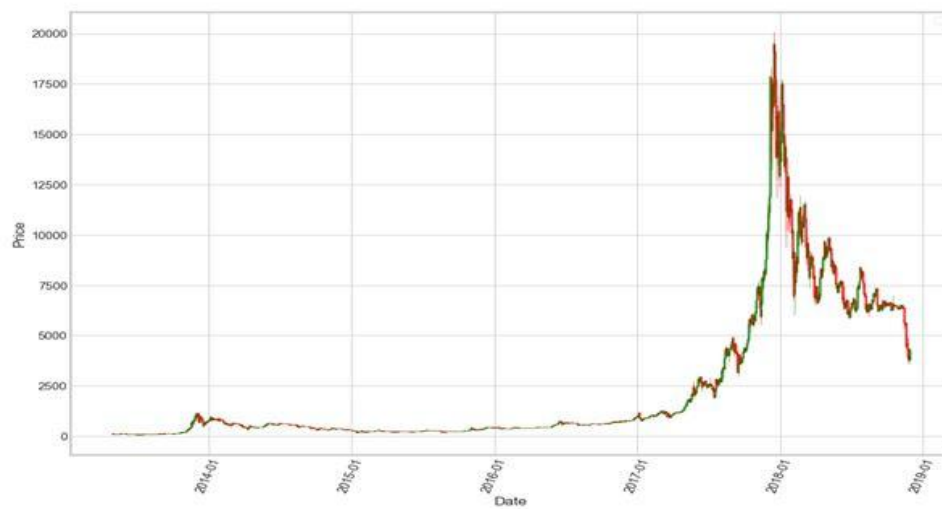
```

```

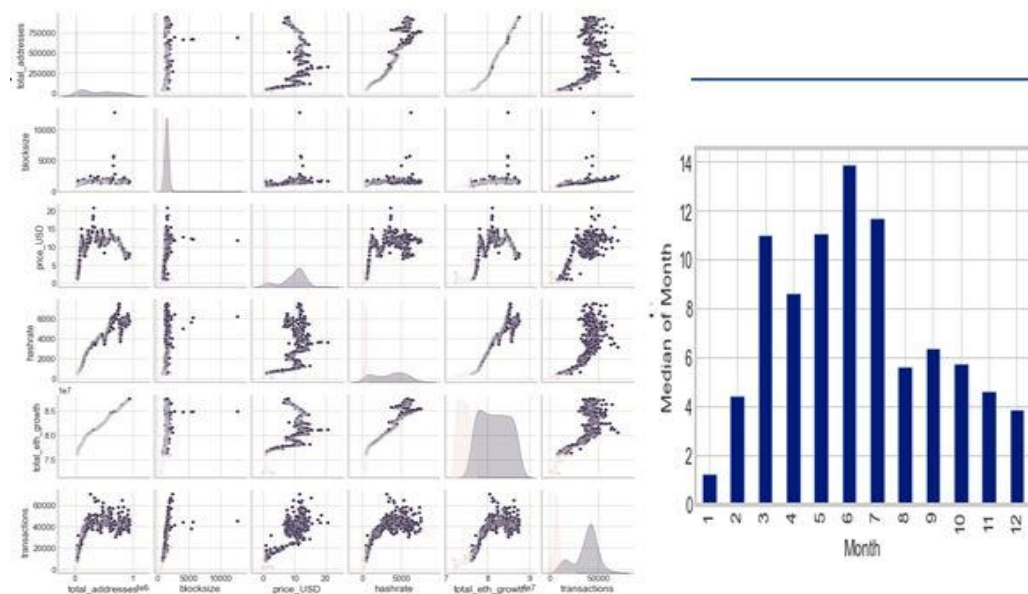
xgb_dtrain = xgb.DMatrix(train_X, train_Y)
xgb_dtest = xgb.DMatrix(test_X)
xgb_params = {'learning_rate' : 0.03,
              'subsample' : 0.7,
              'max_depth' : 5,
              'colsample_bytree' : 0.8,
              'objective': 'reg:linear',
              'eval_metric': 'rmse',
              'silent': 0
             }
xgb_obj = xgb.cv(params = xgb_params, dtrain = xgb_dtrain, early_stopping_rounds=10,
                 verbose_eval=True, show_stdv=False, folds = xgbfolds, num_boost_round =
9999)

```

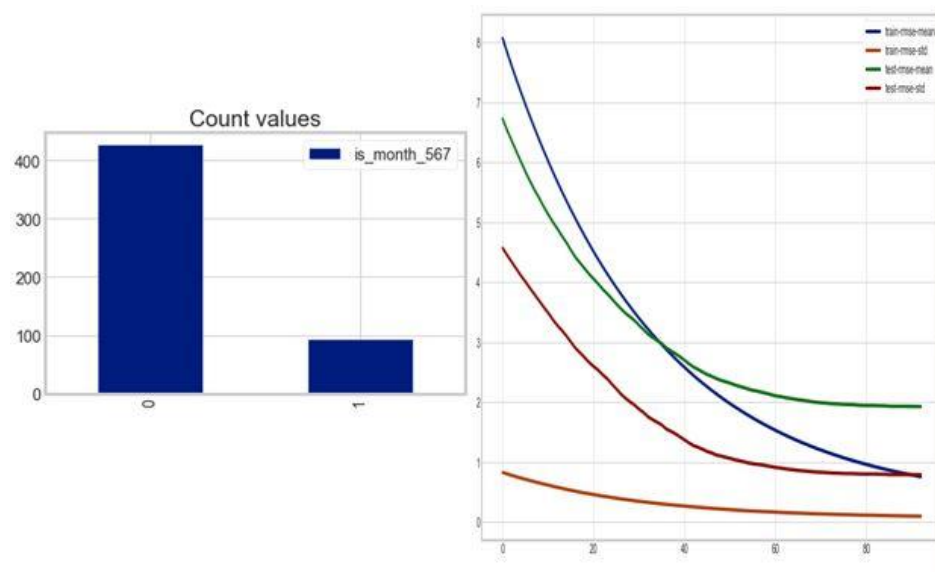
B). SCREENSHOTS.



DATE AND PRICE OF BITCOIN .



MEDIAN OF MONTH OF BITCOIN



COUNT VALUE.

	timestamp	total_addresses	blocksize	price_USD	hashrate	total_eth_growth	transactions	date	year
0	1.438214e+09	9205.0	644.0	0.0	23.7569	7.204931e+07	8893.0	2015-07-30 05:30:00	2015
1	1.438301e+09	9361.0	582.0	0.0	48.1584	7.208550e+07	0.0	2015-07-31 05:30:00	2015
2	1.438387e+09	9476.0	575.0	0.0	55.2709	7.211320e+07	0.0	2015-08-01 05:30:00	2015
3	1.438474e+09	9563.0	581.0	0.0	64.1779	7.214143e+07	0.0	2015-08-02 05:30:00	2015
4	1.438560e+09	9639.0	587.0	0.0	69.8559	7.216940e+07	0.0	2015-08-03 05:30:00	2015

CREDENTIAL DATASET FOR TRANSACTIONS.

train-rmse:8.07303	test-rmse:6.72937
train-rmse:7.83880	test-rmse:6.53860
train-rmse:7.61198	test-rmse:6.36411
train-rmse:7.39178	test-rmse:6.19026
train-rmse:7.18015	test-rmse:6.02068
train-rmse:6.97311	test-rmse:5.85128
train-rmse:6.77188	test-rmse:5.69434
train-rmse:6.57710	test-rmse:5.54495
train-rmse:6.38812	test-rmse:5.41398
train-rmse:6.20522	test-rmse:5.26993
train-rmse:6.02733	test-rmse:5.13786
train-rmse:5.85397	test-rmse:5.01375
train-rmse:5.68675	test-rmse:4.90105
train-rmse:5.52485	test-rmse:4.78075
train-rmse:5.36688	test-rmse:4.66830
train-rmse:5.21393	test-rmse:4.54258
train-rmse:5.06613	test-rmse:4.42114
train-rmse:4.92236	test-rmse:4.32946
train-rmse:4.78267	test-rmse:4.23435
train-rmse:4.64732	test-rmse:4.14209
train-rmse:4.51535	test-rmse:4.06016
train-rmse:4.38772	test-rmse:3.97655
train-rmse:4.26450	test-rmse:3.88832
train-rmse:4.14545	test-rmse:3.81262
train-rmse:4.02967	test-rmse:3.73072
train-rmse:3.91697	test-rmse:3.64260
train-rmse:3.80823	test-rmse:3.56314
train-rmse:3.70268	test-rmse:3.48745
train-rmse:3.59996	test-rmse:3.42869
train-rmse:3.50010	test-rmse:3.36457
train-rmse:3.40295	test-rmse:3.28681
train-rmse:3.31003	test-rmse:3.21235
train-rmse:3.22062	test-rmse:3.13640
train-rmse:3.13128	test-rmse:3.08179
train-rmse:3.04607	test-rmse:3.03036
train-rmse:2.96283	test-rmse:2.97952
train-rmse:2.88209	test-rmse:2.91007
train-rmse:2.80451	test-rmse:2.85841
train-rmse:2.72846	test-rmse:2.81213
train-rmse:2.65473	test-rmse:2.76177
train-rmse:2.58395	test-rmse:2.70271

TRAIN AND TEST THE DATASET.



A strict verification of smart contracts on Blockchain

P. Anil Kumar, P. Sai Ravi Teja, under guidance of Dr. G. Mathivanan

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

A strict verification of smart contracts on Blockchain

Abstract— Blockchain is an evolving technology which helps in keeping records and process transactions in decentralized manner. Blockchain is considered as safest medium because of its decentralized nature and many protocols, algorithms which it follows to make sure that transaction are immutable.

Blockchain concept basically uses BZT theorem, this is considered as one of secured algorithm to predict secure results .however the formal verification approach for the smart contract is still the best way to perform verification. In our paper, we have depicted various algorithm according to which we can verify the smart contract in best possible way.

Index Terms—Smart Contracts, Blockchain, Formal verification,BZT (Byzantine fault tolerance method)

I. INTRODUCTION

smart contract is basically a way to interact between the different parties which agree on the code of smart contract. There may be the case that people may not trust each other but they can trust the machine which runs over the code. smart contract works basically like a code of agreement between two or more smart contract interacting parties. These are a set of self executable codes. the codes in smart contracts are immutable which means once party agrees to specific condition and the deal is locked through smart contract then none can make changes to that since the transaction block is proceed.

In the existing system, the blockchain security provides a secure execution environment. The secure data source provides the credible data to ensure the secure execution.

The contract made by contract participants needs to fully express the intention. of the makers and comply with the law. Then, implementing formal specification and formal verification on above established contract, which is an iterative process. Through multiple behavioural modelling and smart

Contract,we can validate whether the attributes meet the contract requirement or not.

II. MODULE INVOLVED IN SMART CONTRACT

Hash pointer:-

Hash pointer is a hash(key) of the data by cryptography, its point to the location in which data is stored. Hence by using hash pointer we can check whether the data has been tampered or not. A block chain is organized in such a way using hash pointers to link data blocks together. With the hash pointer which pointing to the pre block, each block indicates the address where the data of the pre block is stored.

Digital Signature:-

A digital signature establishes the validity of a set of data by using a cryptographic algorithm. It is also a scheme for regulating that data should not been tampered with. There are three core components that define a digital signature system. The first component is the key generation algorithm, which use to creates two keys, one of it is used to sign messages and to kept privately and called the private key, and the others are made easily available to the public system, thus called the public key, used to validate whether the message has the signature signed with the indicating the private key or not.

Vulnerability Scanning:-

Vulnerability scanning is the study of vulnerabilities which has been discovered, which aimed at avoiding the same mistakes. It use to discover potential vulnerabilities in the execution of contracts, which is important to maximize the level of security and credibility of contracts. Using vulnerability scanning method we can easily detect the possible vulnerabilities in contract while executing.

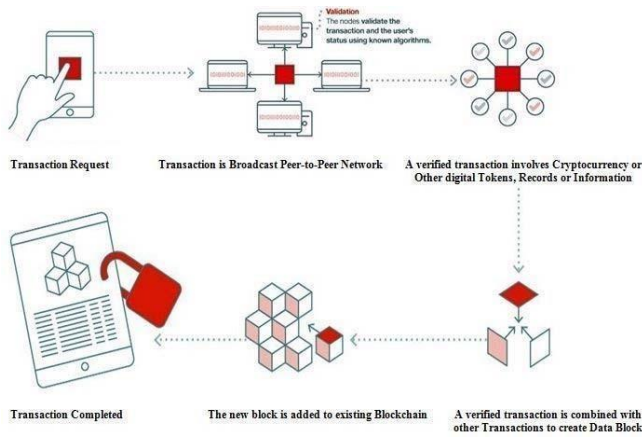
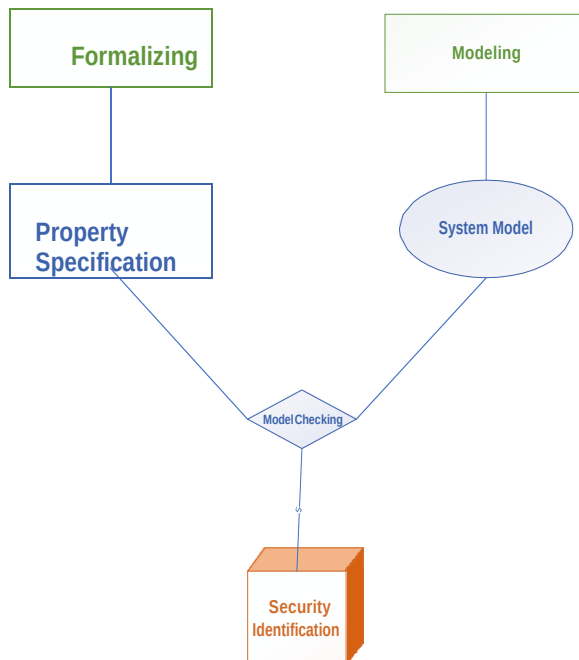


Fig. 1. How Blockchain Works

III. ADVANTAGES OF PROPOSED SYSTEM

- ❖ Less verification numbers and higher verification efficiency.
- ❖ Increases the speed of transaction verification
- ❖ Tool-aided formal verification is strong approach to check the correctness of code
- ❖ Joint optimization scheme are secure and efficient
- ❖ Achieved a nodal authentication and verification of the transmitted data

IV. ARCHITECTURE



V. TECHNOLOGY

- ❖ Python USED Backend Technologies
- ❖ Postmen
- ❖ Spider
- ❖ Eclipse IDE
- Frontend Technologies
- ❖ Web Technologies
- ❖ Bootstrap

VI. ADVANTAGES OF PROPOSED ALGORITHM

- ❖ It's highly scalable, making it suitable for a variety of applications.
- ❖ Higher speed – Transactions are processed faster compared to PoW .
- ❖ Lesser energy consumption as there is no need for supercomputers

VII. FUTURE RESEARCH DIRECTIONS

Recently the smart contracts are emerging key feature of every decentralized application and playing prominent role to solve complex business logics but many research directions are vacant in future some of them are as follows.

- ❖ Resiliency against Hybrid Attacks
- ❖ Optimal Platform for IOT objects
- ❖ Security and Privacy Auditing Protocols
- ❖ Trust and Reliance Management in Social Networks
- ❖ Smart Energy-efficient Mining
- ❖ Innovation of Hybrid Consensus Protocols

VIII. CONCLUSION

The proposed system, actual verification of smart contracts running on distributed ledgers such as blockchain. Currently, the approaches can only handle simple smart contracts and simplified models,

and are not suitable for complex contracts. The proposed system facilitates the application of formal analysis and formal verification by considering technology layers and their security concerns. We picked three layers, implementation, protocol and language, as targets of applications of formal analysis. It proposes a framework to apply formal analysis to each layer by using existing standards and results.

REFERENCES

- I.D. Vujičić, D. Jagodić, and S. Randić, "Blockchain technology, bitcoin, — and Ethereum: A brief overview," in Proc. 17th Int. Symp. INFOTEHJAHORINA (INFOTEH), Mar. 2018, pp. 21–23.
- II.K. Finley. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. Accessed: May. 18, 2019. [Online]. Available: <https://www.wired.com/2016/06/50-million-hack-just-showed-daohuman/>
- III.Z. Zheng, S. Xie, H. Dai, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in Proc. IEEE Int. Congr. Big Data, Big Data Congr., Honolulu, HI, USA, Jun. 2017, pp. 557–564
- IV.N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on ethereum smart contracts (SoK)," in Proc. Int. Conf. Princ. Secur. Trust, Apr. 2017, pp. 164–186. doi: 10.1007/978-3-662-54455-6_8.
- v.M. Alharby and A. V. Moorsel, "Blockchain-based smart contracts: A systematic mapping study," in Proc. Int. Conf. Artif. Intell. Soft Comput., Aug. 2017, pp. 125–140. doi: 10.5121/csit.2017.71011.