Trabajo Fin de Grado

Grado en Ingeniería de las Tecnologías de Telecomunicación

Desarrollo de plataforma en Ethereum para gestión de competiciones de Machine Learning

Autor: Pedro Alberto González Castro

Tutor: Juan José Murillo Fuentes

José Carlos Aradillas Jaramillo

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Desarrollo de plataforma en Ethereum para gestión de competiciones de Machine Learning

Autor:

Pedro Alberto González Castro

Tutor:

Juan José Murillo Fuentes

José Carlos Aradillas Jaramillo

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2022

Proyecto Fin de Carrera: Desarrollo de plataforma en Ethereum para gestión de competiciones de Machine Learning

Autor: Pedro Alberto González Castro

Tutor: Juan José Murillo Fuentes

José Carlos Aradillas Jaramillo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2022

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Por una parte, quisiera agradecer a mis padres, pareja y amigos por su constante apoyo, comprensión, ánimo y mano izquierda durante todo el tiempo que ha durado este proyecto (que no ha sido poco, precisamente).

Por otra parte, agradecer a José Carlos Aradillas todas las aportaciones en forma de consejos, correcciones y comentarios sin los cuales la realización de este trabajo no habría sido posible.

Pedro Alberto González Castro

Autor del proyecto

Sevilla, 2022

Resumen

El mundo de las criptomonedas es actualmente muy popular, principalmente por su aspecto especulativo. Pero ¿cómo funciona la tecnología que hay debajo? ¿Qué diferencia a una criptomoneda de otra? ¿Qué utilidad pueden tener?

Una de las funciones que más me llama la atención es la de los Smart Contracts (piezas de código que pueden ser almacenadas y ejecutadas para efectuar determinadas acciones de forma automática), y el que no todas las criptomonedas ofrezcan la posibilidad de crear Smart Contracts.

El objetivo de este proyecto ha sido crear una plataforma para la gestión de un concurso de Machine Learning. El administrador puede dirigir los concursos (solución, premio, ver participantes etc.) y los concursantes pueden subir su solución a la plataforma y ver la información del concurso. Para la interfaz gráfica se ha usado Python mientras que de la parte del Back-End (donde se almacenan todos los datos del concurso y se produce la computación) se encarga por completo la criptomoneda. Con ello, dejaríamos de necesitar poseer y gestionar un servidor (o red de servidores) para, por ejemplo, almacenar los datos, calcular las puntuaciones de cada participante y determinar un ganador.

La criptomoneda que he elegido ha sido Ethereum principalmente porque es una de las criptomonedas más famosas y una de las que tiene una mayor comunidad detrás. Esto facilitará el aprendizaje teniendo en cuenta que empiezo desde cero.

Abstract

The world of cryptocurrencies is currently very popular, mainly because of its speculative aspect. But how does the technology underneath work, what differentiates one cryptocurrency from another, and how useful can they be?

One of the functions that most catches my attention is Smart Contracts (pieces of code that can be stored and executed to perform certain actions automatically), and not all cryptocurrencies offer the possibility of creating Smart Contracts.

The goal of this project has been to create a platform for managing a Machine Learning contest. The administrator can manage the contests (solution, prize, view participants etc.) and the contestants can upload their solution to the platform and view the contest information. For the GUI Python has been used for the graphical interface while the Back-End (where all the contest data is stored and the computation takes place) is completely handled by cryptocurrency. With this, we would no longer need to own and manage a server (or network of servers) to, for example, store the data, calculate the scores of each participant and determine a winner.

The cryptocurrency I have chosen has been Ethereum mainly because it is one of the most famous cryptocurrencies and one of those with a larger community behind it. This will make learning easier considering that I am starting from scratch.

Index

FIGURE INDEX

Notation

AI	Artificial Intelligence
BFT	Byzantine Fault Tolerance
EBL	Explanation Based Learning
EVM	Ethereum Virtual Machine
FBA	Federated Byzantine Agreement
FN	False Negative
FP	False Positive
ML	Machine Learning
MPT	Merkle Patricia Trie
P2P	Peer To Peer
PoS	Proof of Stake
PoW	Proof of Work
RLP	Recursive Length Prefix
SSZ	Simple Serialize
TN	True Negative
TP	True Positive
TPS	Transactions Per Second

1 INTRODUCTION

The rise of Machine Learning in recent decades has caused the number of developers and researchers in this field to increase considerably. Therefore, a wide variety of problems and tasks have appeared that can be solved by applying Artificial Intelligence or Machine Learning models, and for each of these tasks a large number of solutions are proposed over time.

Generally, to evaluate the performance of a Machine Learning model, a set of test data that have not been used in any case for the design and training of the model is used. The evaluation of the model on this test data provides essential information for researchers or developers when choosing the best model to implement and deploy in production.

When the number of stakeholders interested in solving a problem is large, the number of proposed models grows considerably and the evaluation of the models for benchmarking becomes more complicated.

In order to manage the comparison of models as objectively as possible, several competitions have been developed for this purpose. The best known at present are the competitions managed by Kaggle, which is a platform dedicated to solving the above-mentioned problem. In this type of competition, the organizers provide a set of labeled data, i.e., with the solution for the participants to design and train their models. Subsequently, unlabeled data (without the solution) are provided for the participants to find the solution and send it to the organizers. The organizers will then evaluate the solution and publish a ranking of the best models.

Some of these contests are free of charge, i.e., participants do not have to pay any kind of fee to participate. The participation consists of submitting the solution of a model, and you can participate as many times as you want. This type of free contests has some disadvantages since there are authors who send a high number of entries adjusting the parameters of the model to obtain a better result in the test set.

With the aim of preventing participants from submitting too many solutions of this type, there are more serious contests where each submission has a cost. This makes the organization of a contest of this type more complex because, in addition to managing the evaluation of the models, there is also a monetary management of the models. Normally, this type of non-free contest has a monetary prize for the model or models that achieve the best result.

Currently, the monetary management of this type of contests is done through online payments and transfers using the different options that technology offers us, similar to the realization of an online purchase.

The objective of this work is to unify both the evaluation of the models and the monetary management of the contests using cryptocurrencies. The system on which cryptocurrencies are based allows us to perform both the computations and the transparent management of payments and collections.

For this purpose, an application has been designed and developed in Ethereum in which contestants are charged each time they send a solution, the performance of the solution is computed in the Ethereum system itself and once the contest is finished, the monetary prize is automatically sent to the contestant who has sent the best solution.

The rest of the paper is organized as follows: in Chapter 2 there is an introduction to cryptocurrencies and blockchain; in Chapter 3 there is an introduction to Ethereum, which is the cryptocurrency used in the work; in Chapter 4 there is an introduction to Machine Learning; in Chapter 5 the proposed solution and the whole implementation of the ML contest management system are detailed; and in Chapter 6 the conclusions of the work are shown.

2 INTRODUCTION TO CRYPTOCURRENCIES

Right now, Bitcoin feels like the Internet before the browser.

- Wences Casares -

As this work is based on concepts like *blockchain*, *Bitcoin* and *cryptocurrency*, a good starting point can be to briefly describe them. We could say -in a few words- that *blockchain* is the technology that enables the existence of cryptocurrency. Then, a *cryptocurrency* is a medium of exchange (such as dollar or Euro) but is digital. For example, it uses encryption techniques to control the creation of monetary units and to verify the transfer of funds. Lastly, *Bitcoin* is the name of the best-known cryptocurrency and the first real-world application of the blockchain technology.

2.1 Blockchain, cryptocurrencies and Ethereum

2.1.1 Cryptocurrency

Cryptocurrency is a form of payment that can be exchanged online for goods and services. Many companies have created their own currencies (also known as tokens), and these can be traded specifically for the good/service that the company provides. We can use the analogy of casino coins: you will have to exchange “real” currency for the cryptocurrency in order to access the good/service.

Usually, each cryptocurrency has its own blockchain network (blockchain is a database with all the data of the cryptocurrency, see section 2.1.2 for more information). It is not possible to send any amount of X cryptocurrency to an address of a different cryptocurrency.

2.1.1.1 Cryptocurrencies examples

Here are some examples of different cryptocurrencies. The key points that differentiate them from each other are mining algorithms (section 2.1.3), the block generation time, the total number of coins, transaction speed, market capitalization, and price.

Bitcoin is the world’s first cryptocurrency and the most popular one. Launched in 2009, it was the first real-life use of the blockchain technology. It was created by Satoshi Nakamoto, which is the pseudonym of the person (or people) who developed Bitcoin, authored the white paper and deployed the original implementation. At the time of writing, the price of one Bitcoin is near to 42.000$, and the market capitalization is about 800 billion dollars (more than companies like Toyota or Netflix) [1] [2].

Regarding mining (section 2.1.3), it uses a Proof of Work (PoW) consensus mechanism and the SHA-256 algorithm so, the bigger the network gets, the harder it is for standalone miners to obtain benefits.

Bitcoin has one of the slowest transaction speeds, with an average of 4-6 transactions per second (TPS) and it takes about 10 minutes to create a new block.

One of the biggest problems of this cryptocurrency is scalability. The more users trying to transfer funds, the more congested the network gets; this is causing high periods for transfer confirmation and also high bids for each transaction.

Litecoin is an altcoin ("spin-off") of Bitcoin, launched in 2011. At the time of preparation of this document, it has a market capitalization of 8.6 billion dollars, and one Litecoin costs 125$. The main differences between Litecoin and Bitcoin are [3] [4]:

- Time to confirm a transaction: Litecoin processes a block every 2.5 minutes, on average its speed is 56 transactions per second.
- Regarding mining (section 2.1.3), Litecoin uses the Scrypt algorithm (instead of SHA-256) in order to allow anyone with an average personal computer to mine and be able to be a standalone.

Ethereum was conceived in 2013 by the programmer Vitalik Buterin. Development started in 2014 and it was crowdfunded; the deployment was in 2015. The platform offers to anyone the possibility of creating decentralized applications so the rest of the users can interact with them. Currently, it is the second highest market capitalization cryptocurrency, with a value of 360 billion dollars. The price of one Ethereum is about 3.000$ [5].

Regarding mining (section 2.1.3), Ethereum has recently moved from PoW to Proof of Stake (PoS) in order to avoid the issues with the miners. All participants have a fair chance to get a reward.

This cryptocurrency has a higher average of transactions per second than Bitcoin, about 15; it is planned to increase it in the future.

Dragonchain was originally developed by The Walt Disney Company in 201, became open-source in 2016 and the beta was released in 2018. As of this writing, it has a market capitalization of 30 million dollars, and the price of one Dragonchain is 0.080$ [6] [7].

It is a cryptocurrency designed for a quick development and deployment of blockchain based applications. It offers different consensus mechanisms to allow users to choose the level of verification needed for their transaction.

2.1.1.2 Wallets

Crypto wallets store the public and/or private keys needed for cryptocurrencies transactions, keeping the cryptocurrency safe and accessible (note that wallets do not store the cryptocurrency itself). Sometimes, in addition to this function, a wallet can offer the possibility of signing and/or encrypting information. In most of the cases, the public key allows others to send transactions to the address derived from it, while the private key is for spending the related cryptocurrency.

We can divide wallets into two types: physical wallets (to store the keys offline) and software wallets (to store the keys online).

Physical wallets

Probably the simplest kind of wallet are the paper wallets. It is just a piece of material with the private and public keys printed. They can also have a QR code scannable by an app. As they are not reachable by Internet, they are considered one of the most secure wallets.

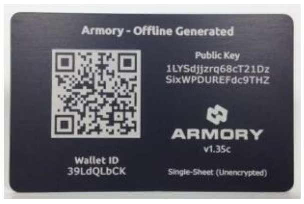


Figure 2-1: Physical Wallet

There are also hardware wallets. They are an encrypted USB containing public and private keys. Some of them allow the use of a two-factor authentication service or sending transactions to a different account.

Figure 2-2: Hardware Wallet

Software Wallets

A software wallet is a computer program or mobile app that holds private and public keys online. There are three main types: Web-based wallets (like MetaMask), Desktop wallets (like Exodus) and Mobile wallets (like blockchain.com wallet)

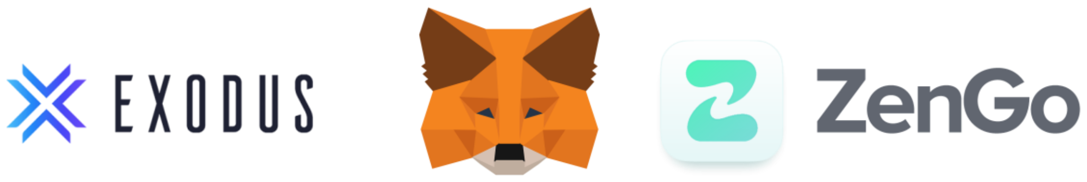


Figure 2-3: Software Wallets Examples – Exodus, MetaMask and ZenGo

2.1.2 Blockchain

A blockchain is a database, mostly used in a decentralized way. Different types of information can be stored in it, but the most common use so far has been as a ledger for transactions. By the use of this technology, transactions can be confirmed without a need for a central clearing authority (*i.e.* a bank).

It differs from a typical database in the way it stores information. Blockchains store data in blocks that are chained together. Then, as new data comes in it, is saved into a fresh block. Once the block is filled with data it is chained onto the previous block, which makes the data chained together in chronological order. This system also inherently makes an irreversible timeline of data. When a block is filled it is set in stone and becomes a part of this timeline [8].

2.1.2.1 Decentralization

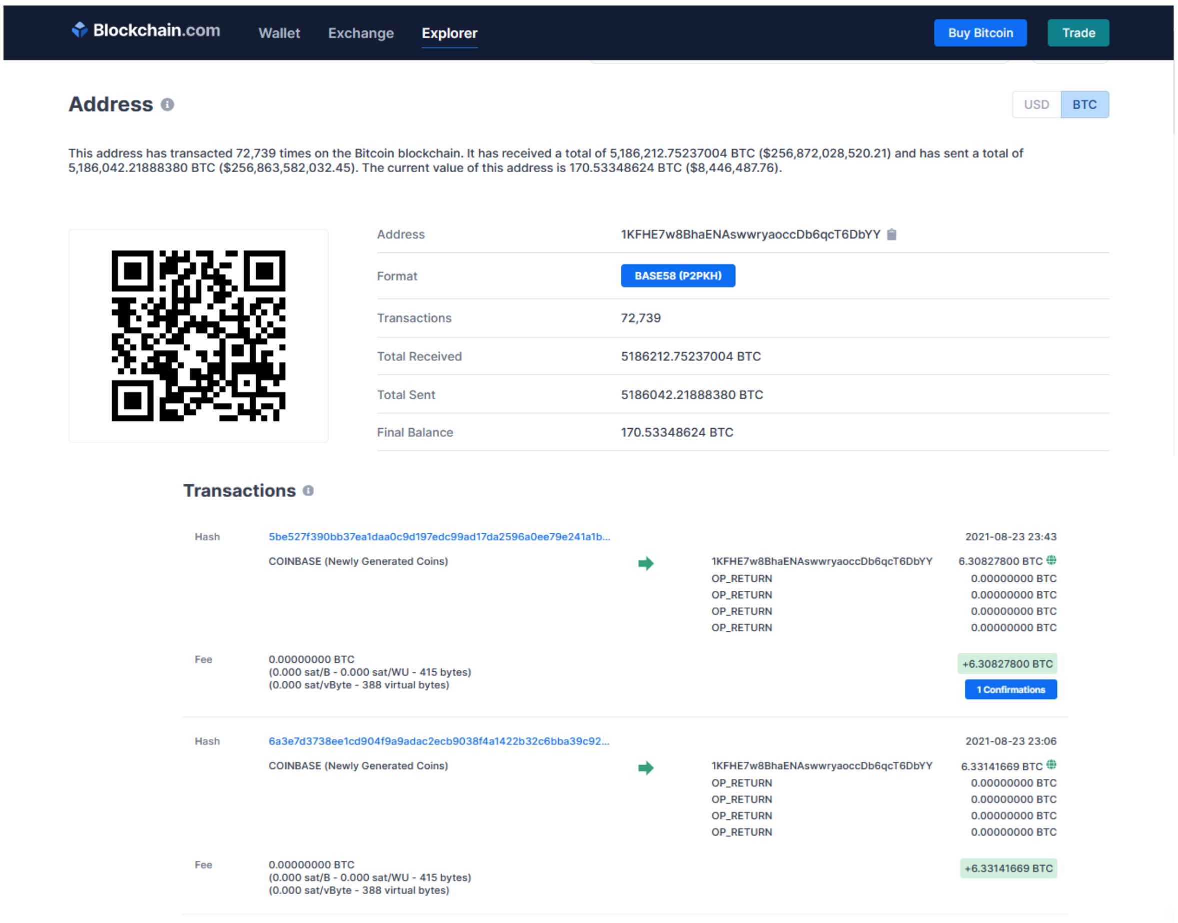
As with any database, computers are necessary to store and process the blockchain information. Taking as example the case of Bitcoin that stores every Bitcoin transaction ever made (uses its blockchain as a ledger), each computer or group of computers is operated by a unique individual or group of individuals. On other words, Bitcoin consists of thousands of computers, but each computer or group of computers that hold its blockchain is in a different geographic location and they are all operated by separate individuals or groups of people. These computers that makeup Bitcoin's network are called nodes [9].

Every node has a full record of the data that has been stored on the blockchain since its inception or, at least, is able to ask for any missing information to a different node. In this project we are going to work with Ethereum, which has a similar behaviour.

2.1.2.2 Transparency

Most of the cryptocurrencies are totally transparent regarding the tracking of transactions. Because of the decentralized nature of Bitcoin's blockchain (Bitcoin has been the reference for most of the cryptocurrencies), all transactions can be examined. It is possible to do it by having a personal node (which has its own copy of the chain) or using blockchain explorers. Blockchain explorers offer the option of examining a block and a specific transaction.

Here is an example of a random Bitcoin block (not all transactions are being shown in the image) https://www.blockchain.com/btc/address/1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY :



Address

This address has transacted 72,739 times on the Bitcoin blockchain. It has received a total of 5,186,212.75237004 BTC ($256,872,028,520.21) and has sent a total of 5,186,042.21888380 BTC ($256,863,582,032.45). The current value of this address is 170.53348624 BTC ($8,446,487.76).

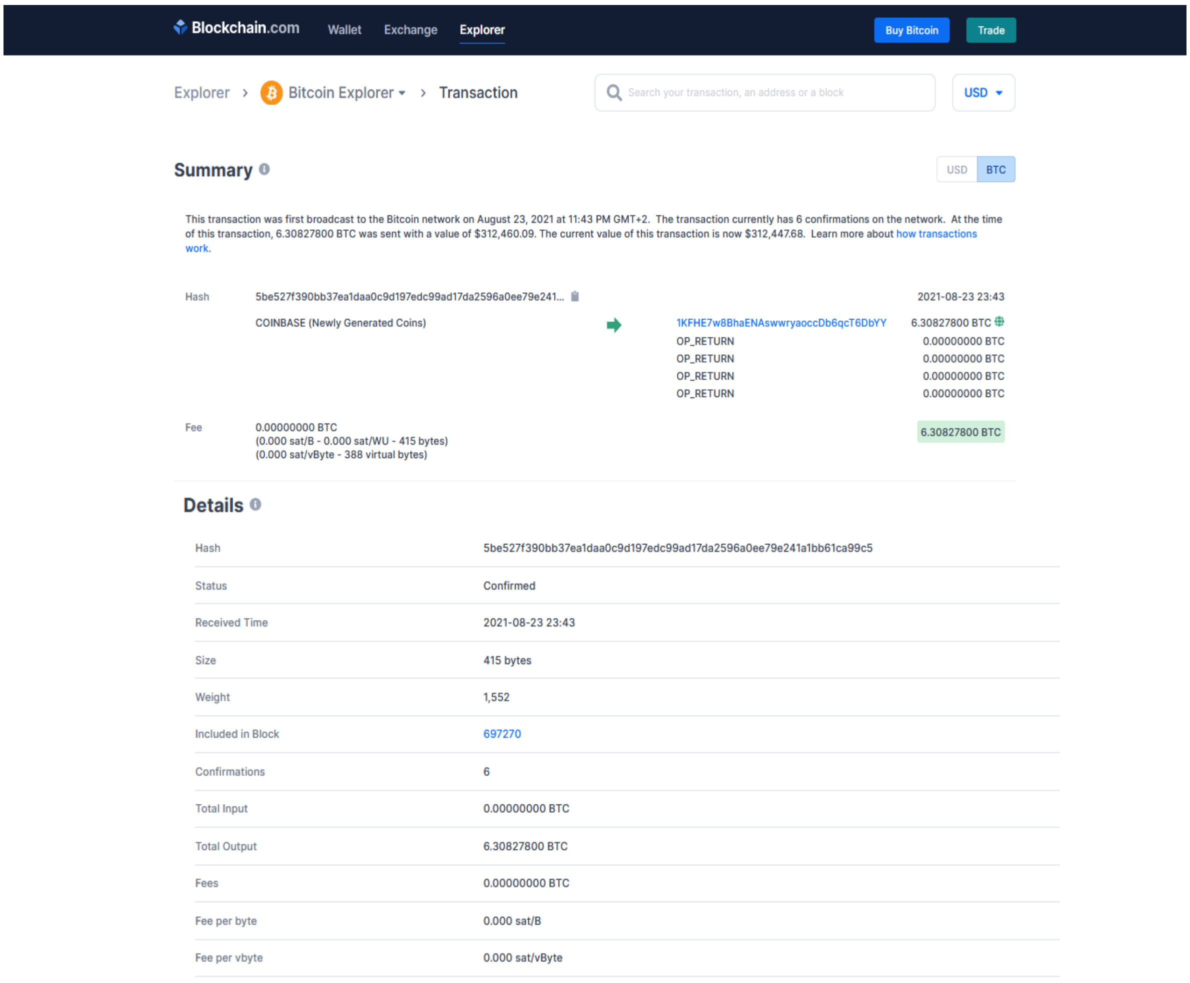
Address	1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY
Format	BASE58 (P2PKH)
Transactions	72,739
Total Received	5186212.75237004 BTC
Total Sent	5186042.21888380 BTC
Final Balance	170.53348624 BTC

Transactions

```
Hash   5be527f390bb37ea1daa0c9d197edc99ad17da2596a0ee79e241a1b...                           2021-08-23 23:43
       COINBASE (Newly Generated Coins)        →   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY   6.30827800 BTC
                                                   OP_RETURN                            0.00000000 BTC
                                                   OP_RETURN                            0.00000000 BTC
                                                   OP_RETURN                            0.00000000 BTC
                                                   OP_RETURN                            0.00000000 BTC
Fee    0.00000000 BTC                                                                  +6.30827800 BTC
       (0.000 sat/B - 0.000 sat/WU - 415 bytes)                                         1 Confirmations
       (0.000 sat/vByte - 388 virtual bytes)

Hash   6a3e7d3738ee1cd904f9a9adac2ecb9038f4a1422b32c6bba39c92...                           2021-08-23 23:06
       COINBASE (Newly Generated Coins)        →   1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY   6.33141669 BTC
                                                   OP_RETURN                            0.00000000 BTC
                                                   OP_RETURN                            0.00000000 BTC
                                                   OP_RETURN                            0.00000000 BTC
                                                   OP_RETURN                            0.00000000 BTC
Fee    0.00000000 BTC                                                                  +6.33141669 BTC
       (0.000 sat/B - 0.000 sat/WU - 415 bytes)
       (0.000 sat/vByte - 388 virtual bytes)
```

Figure 2-4: Bitcoin Block Example

Here is an example of a random Bitcoin transaction (not all sections are being shown in the image) https://www.blockchain.com/btc/tx/5be527f390bb37ea1daa0c9d197edc99ad17da2596a0ee79e241a1bb61ca99c5 :


Blockchain.com
Wallet
Exchange
Explorer
Buy Bitcoin
Trade
Explorer › Bitcoin Explorer › Transaction
Search your transaction, an address or a block
USD
USD
BTC

Summary

This transaction was first broadcast to the Bitcoin network on August 23, 2021 at 11:43 PM GMT+2. The transaction currently has 6 confirmations on the network. At the time of this transaction, 6.30827800 BTC was sent with a value of $312,460.09. The current value of this transaction is now $312,447.68. Learn more about how transactions work.

Hash	5be527f390bb37ea1daa0c9d197edc99ad17da2596a0ee79e241...			2021-08-23 23:43
	COINBASE (Newly Generated Coins)	➡	1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY	6.30827800 BTC
			OP_RETURN	0.00000000 BTC
			OP_RETURN	0.00000000 BTC
			OP_RETURN	0.00000000 BTC
			OP_RETURN	0.00000000 BTC
Fee	0.00000000 BTC (0.000 sat/B - 0.000 sat/WU - 415 bytes) (0.000 sat/vByte - 388 virtual bytes)			6.30827800 BTC

Details

Hash	5be527f390bb37ea1daa0c9d197edc99ad17da2596a0ee79e241a1bb61ca99c5
Status	Confirmed
Received Time	2021-08-23 23:43
Size	415 bytes
Weight	1,552
Included in Block	697270
Confirmations	6
Total Input	0.00000000 BTC
Total Output	6.30827800 BTC
Fees	0.00000000 BTC
Fee per byte	0.000 sat/B
Fee per vbyte	0.000 sat/vByte

Figure 2-5: Bitcoin Transaction Example

However, using blockchain does not imply transparency. For example, Monero is a cryptocurrency that is famous for being totally obscure. According to Monero's site: Monero is a secure, private and untraceable currency system. Monero uses a special kind of cryptography (it uses ring signatures, zero-knowledge proofs, and "stealth addresses" to obfuscate transaction details) to ensure that all its transactions remain 100% unlinkable and untraceable.

2.1.3 Mining: Transaction validation

In any centralized database, a central administrator has the authority to maintain and update the database. For example, if the database stores the voters in a country, the administrator would add or delete the people who can vote. Oppositely, public blockchains operate as decentralized, self-regulating systems without a central authority.

Consensus mechanisms are protocols that make sure the blockchain nodes are synchronized and agree on which transactions are legitimate. A transaction is added to the blockchain only after it has been validated, which ensures it is the only version of the truth.

The most popular consensus mechanism is the Proof of Work (PoW). It is the first one ever implemented on a blockchain (it was the solution for Bitcoin). After Bitcoin, most cryptocurrencies made the same decision. Any participant is allowed to announce their conclusions about the transmitted information to the chain. To do that, it must resolve a mathematical puzzle. Then, all the members of the system can independently verify these conclusions. Finding the solution and checking it are asymmetric tasks, it is hard to find the solution but it is really easy to check it. The only way to solve this puzzle is by trial and error; that implies that miners with higher computational power will be able to solve the problem faster.

A different approach is used in the Proof of Stake (PoS) consensus mechanism. PoS is based on the premise that those who own most coins on the blockchain have a vested interest in maintaining the network. It uses a pseudo-random election process to select a node to validate the next block. The selection is based on certain parameters such as staking age or the node's wealth.

Another consensus mechanism is the Federated Byzantine Agreement (FBA); Ripple [10] and Stellar [11] cryptocurrencies are using it for example. It is an evolution of the Byzantine Fault Tolerance (BFT) algorithm, in which there is a list of recommended validators. These validators are defined by a central authority (usually the company); even if anyone can potentially become a validator, only those the authority adds to that list can participate in consensus. The idea of FBA is to decentralize the BFT algorithm, the validators are free to choose other validators they trust. Their list of trusted nodes is known as their quorum slice. In this kind of system, several quorum slices will overlap and thus form a quorum, which is the number of nodes required to reach an agreement within the system. Anyone can become a validator and participate in the network’s consensus as long as any other participating validator adds them to their quorum slice [12].

3 Ethereum

Smart Contracts are not contracts, nor smarts

- Famous statement, author unknown -

There is a single (canonical) computer in the *Ethereum* universe. It is called the Ethereum Virtual Machine (EVM) whose state everyone on the Ethereum network agrees on. Everyone who participates in the Ethereum network (every Ethereum node) keeps a copy of the state of this computer. Any participant can request this computer to perform computation. Whenever such a request is broadcast, other participants on the network verify, validate, and execute the computation (this can cause a state change in the EVM, which will be propagated throughout the network). These requests are known as transaction requests; the blockchain will store the record of all of them.

The aim of *Ether*, as a cryptocurrency, is to bring the existence of a market for computation. This market will provide an economic reward for the people in charge of verify/execute transaction requests and for those that provide computational resources to the network. The participant who broadcasts a transaction request must also offer some amount of ether to the network, as a bounty to be awarded to whoever eventually does the work of verify/execute/commit the transaction and broadcasting it to the network.

Regarding *Smart Contracts*, we can describe them -on a simplified way- as scripts that perform certain actions (if some conditions are satisfied) when called with some specific parameters. Any developer can develop a smart contract and make it public using the blockchain as its data layer, for a fee paid to the network. Then, any user can call the smart contract to execute its code, again for a fee paid to the network.

Note: Regarding used documentation for this section, there are three main references: Ethereum whitepaper [13], Ethereum yellowpaper [14] and Ethereum official documentation [15].

3.1 Decentralized Applications (Dapps)

A decentralised application (Dapp) is an application built on a decentralised network that combines a smart contract and a frontend user interface. In Ethereum, smart-contracts are accessible and transparent – like open APIs – so it is possible to use a Smart Contract that someone else has deployed [16] [17] [18] [19] [20].

Ethereum Dapps will have a Front-End (and User Interface) written in any language, and its Back-End will be a Smart Contract.

3.1.1 Benefits of Ethereum Dapps

- **Zero downtime:** Once the smart contract at the core of an app is deployed and on the blockchain, the network will always be able to serve clients looking to interact with the contract. Malicious actors therefore cannot launch denial-of-service attacks targeted towards individual dapps.

- **Privacy:** No real-world identity is required to deploy or interact with a dapp.
- **Resistance to censorship:** No single entity on the network can block users from submitting transactions, deploying dapps, or reading data from the blockchain.
- **Complete data integrity:** Data stored on the blockchain is immutable and indisputable, thanks to cryptographic primitives. Malicious actors cannot forge transactions or other data that has already been made public.
- **Trustless computation/verifiable behaviour:** Smart Contracts can be analysed and are guaranteed to execute in predictable ways, without the need to trust a central authority. This is not true in traditional models. For example, when using online banking systems, people must trust that financial institutions will not misuse our financial data, tamper with records, or get hacked.

3.1.2 Implications/Drawbacks of Ethereum Dapps

- **Maintenance:** Dapps can be harder to maintain because code and data published to the blockchain is hard to modify. It’s complicated for developers to make updates to their dapps (or the underlying data stored by a dapp) once they are deployed - even if bugs or security risks are identified in an old version.
- **Performance overhead:** There is a huge performance overhead, and scaling is really complicated. To achieve the level of security, integrity, transparency, and reliability that Ethereum aspires to, every node runs and stores every transaction.
- **Network congestion:** If one dapp is using too many computational resources, the entire network gets backed up. Currently, the network is only able to process about 10-15 transactions per second. If transactions are being sent in faster than this, the pool of unconfirmed transactions can quickly balloon.
- **User experience:** It may be harder to engineer user-friendly experiences: The average end user might find it too difficult to set up a tool stack necessary to interact with the blockchain in a truly secure fashion.
- **Centralization:** User-friendly and developer-friendly solutions built on top of the base layer of Ethereum might end up looking like centralised services anyways: for example, such services may store keys or other sensitive information server-side, serve a frontend using a centralised server, or run important business logic on a centralised server before writing to the blockchain. This eliminates many of the advantages of blockchain over the traditional model.

3.2 Storage: Data Structures

Ethereum creates, stores and transfers large volumes of data. This data must be stored in a memory-efficient way to allow anyone to run a node on a -relatively- modest hardware. To achieve this, a structure named Merkle Patricia Tries (MPT) is used on the Ethereum stack. Patricia Merkle Tries are structures that encode key-value pairs into a deterministic and cryptographically authenticated trie. They are a combination between Patricia Tries and Merkle Tries [21] [22] [23] [24] [25] [26] [27] [28] [29].

3.2.1 Patricia Tries

Patricia trie is a data structure which is also called Prefix tree, binary radix tree or trie. Trie uses a key as a path so the nodes that share the same prefix can also share the same path. Starting from the root node of the tree, each character in the key points the next child node to follow, where the values are stored in the leaf nodes that terminate every path through the tree. This structure is fastest at finding common prefixes, simple to implement, and requires small memory. For example, it is mainly used for implementing routing tables.

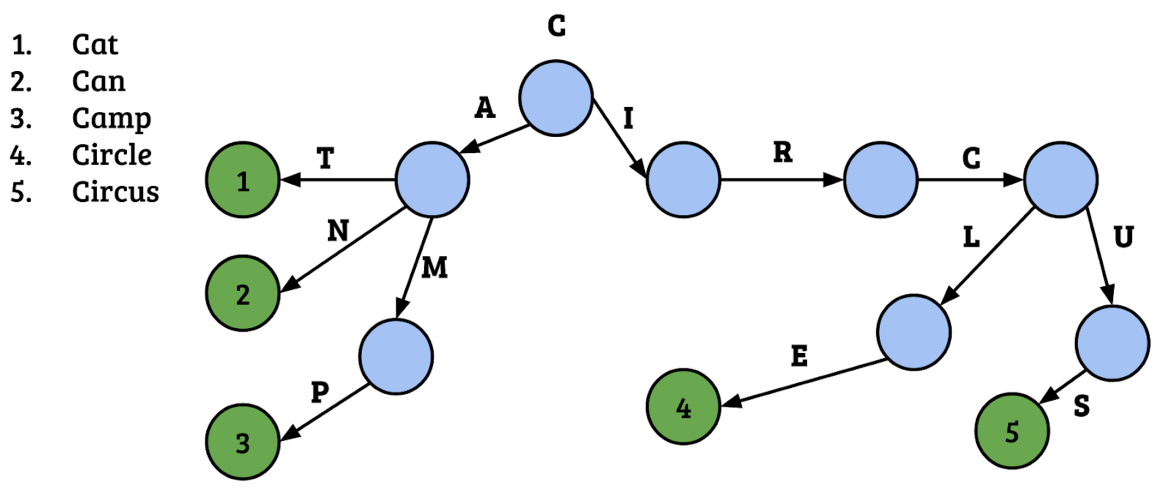


Figure 3-1: Patricia Trie Example Diagram

3.2.2 Merkle Tries

Merkle tree (also known as Hash Tree) is a tree of hashes: leaf nodes store data while parent nodes contain their children's hash and the hashed value of the sum of their children's hashes. The main benefit of this manner of hashing is the Merkle Proof. It allows verifying if data belongs in the Merkle Tree without the need of storing the complete dataset. Merkle proofs are established by hashing a hash's corresponding hash together and climbing up the tree until the root hash is obtained, which is publicly known. If the obtained root hash is the same as the publicly known root hash, the data is valid. We can see it with an example:

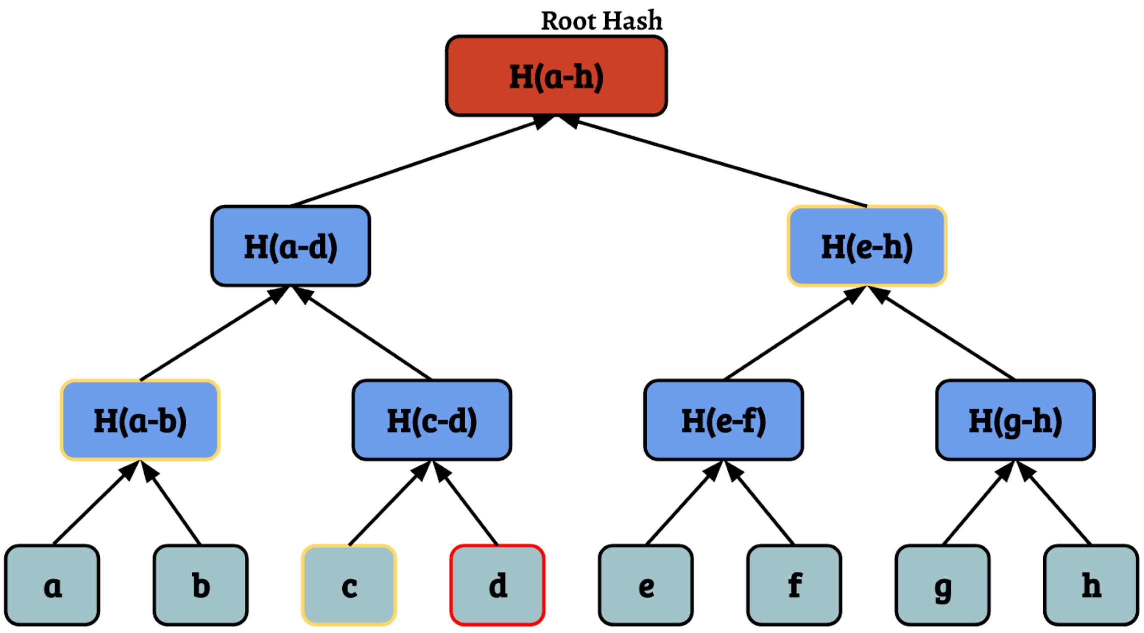


Figure 3-2: Merkle Trie Example Diagram

If 'd' needs to be verified, only 'c', 'H(a-b)' and 'H(e-h)' have to be requested (remember that the root hash is publicly known). With that, we would be able to obtain 'H(c-d)', 'H(a-d)' and finally 'H(a-h)'. This feature is very useful for light nodes (section 3.8) that need to ask for some information (to bigger nodes) in order to verify requests.

3.2.3 Merkle Patricia Tries

The ‘Merkle side’ of the Merkle Patricia Trie (MPT) is that the key of each node is a hash of the value. To be more precise, the ‘keccak256’ function is being used to hash, while the value itself is prior serialized with Recursive Length Prefix (RLP) [30] serialization currently (it is going to be replaced with Simple Serialize (SSZ) [31] after The Merge).

Patricia tries have one important limitation: they are not efficient. To store just one (key, value) binding where the path is 64 characters long (number of nibbles produced by keccak256 function), around one kilobyte of space to store one level per character would be needed, and each deletion or lookup will take the full 64 steps. To improve its efficiency, some kind of nodes have been created. With 64-character paths, after traversing the first nodes, a node where no divergent path exists will be reached (at least for part of the way down). There are 4 types of nodes (encodedPath contains the partial path to skip):

- **Null:** Represented as the empty string
- **Branch:** A 17-item node [i0 ... i15, value]
- **Leaf:** A 2-item node [encodedPath, value]. Their usage is to store the rest of the path and the value.
- **Extension:** A 2-item node [encodedPath, key]. Their usage is to store the rest of the path and the key to point to the next branch.

To differentiate Leaf and Extention nodes and to indicate if the remaining partial path is even or odd, the first nibble will be used as a flag:

- **0:** Extension node, even partial path (a padding ‘0’ will be added).
- **1:** Extension node, odd partial path.
- **2:** Leaf node, even partial path (a padding ‘0’ will be added).
- **3:** Leaf node, odd partial path.

For better understanding, we can use a simplified example. We observe a MPT containing the next (key:values): (‘do’:‘Verb’), (‘dog’:‘Puppy’), (‘doge’:‘Coin’) and (‘horse’:‘Stallion’). For an easier comprehension, values will be represented as strings (keys are represented in bytes):

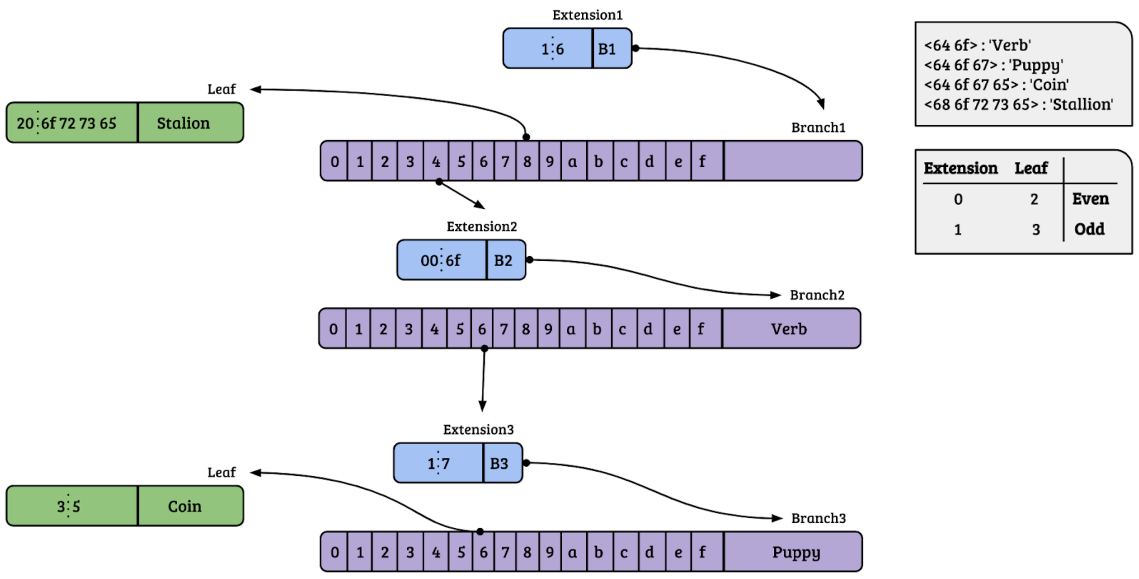


Figure 3-3: Merkle Patricia Trie Example Diagram

Please note that values would be stored with RLP or SSZ serialization and that we are not able to choose the key of those values, they would be the ‘keccak256’ hash of their own value.

Ethereum uses four Merkle Patricia Tries, three of them are stored (the root hash) on the blocks:

- **State Trie:** For storing account-related information. There is one global state trie, and it updates over time. The key is ‘keccack256’ hash of the Ethereum address, and the value is the elements of

an Ethereum account (nonce, balance, codeHash, storageRoot) serialized with RLP or SSZ. Note that storageRoot is the root hash of another MPT, the Storage Trie.
    - **Storage Trie:** Where Smart Contracts data lives (its variables and their content).
- **Transactions Trie:** There is a separate transactions trie for every block, storing (key, value) pairs. The key is the RLP serialization of the transactionIndex, where the transaction sits in the block (The order of the transactions in a block is decided by the miner who assembles the block). Mined blocks are never updated; the position of the transaction in a block is never changed.
- **Receipts Trie:** Similar to the transactions trie; there is a separate receipts trie for every block. The key is the same as transactions trie (the serialization of the transaction index) and the value would be the content of the receipt. All transactions generate receipts (any interaction with the blockchain that modifies its content will generate one). The receipt will provide information such as the amount of consumed gas, the gas price, logs (they can be generated by a Smart Contract) or the transaction status (failure, successful).

3.3 Accounts

An Ethereum account is an entity with an ether (ETH) balance that can send transactions on Ethereum. There are two types of accounts:

- **Externally owned:** They are controlled by anyone with the private keys.
- **Contract:** The associated account of a Smart Contract. It is controlled by the code of the Smart Contract.

Both account types can manage ETH (receive, send and hold) and interact with Smart Contracts. The creation of an account is costless, but here a nuance should be clarified; in case of contract account a payment is required to develop the contract (as it requires computation and storage of the EVM), but not for the contract account itself [32] [33] [34] [35].

3.3.1 Elements

Each Ethereum account has four elements:

- **Nonce:** A counter that indicates the number of transactions sent from the account. The purpose of this field is to assure that transactions are only processed once. The value of this field will only increase when a transaction is processed (if someone tries to send two transactions with the same nonce, the first one will be the processed one). In a contract account, this number represents the number of contracts created by the account.
- **Balance:** The amount of ETH owned by the address. It is stored in Wei, the smallest denomination of ETH (there are 1e+18 Wei per ETH).
- **CodeHash:** This field is for contract accounts (this hash identifies the code of an account on the EVM). Contract accounts have code fragments that can perform different operations. It cannot be changed unlike the other account fields. All such code fragments are contained in the state database under their corresponding hashes (codeHash) for later retrieval. For externally owned accounts, the codeHash field is the hash of an empty string.
- **StorageRoot:** Also known as storage hash. A 256-bit hash of the root node of a Merkle Patricia trie that encodes the storage contents of the account.

3.4 Transactions

We could say that transferring ETH from one account to another is the simplest transaction, but in general a transaction is a cryptographically signed instruction from an account. This instruction will update the state of the Ethereum network [36] [37] [38] [39].

3.4.1 Caveat: Gas

The concepts of transactions, blocks and gas are closely related. There is a later section (section 3.6) for gas, hence here are some key concepts for the better understanding of transactions and block sections:

- Gas if for paying transaction fees.
- Units of gas represent computational steps. Each transaction requires computational effort.
- Gas fees are payed in ETH but, as they are “small” amounts of ETH, there are two main denominations used:
    - **GWei:** 1 GWei is equal to 10^{-9} ETH.
    - **Wei:** 1 Wei is equal to 10^{-18} ETH.

3.4.2 Elements

A submitted transaction includes the following information:

- **Recipient:** The receiving address. If it is an externally owned account, the transaction will transfer a certain amount of ETH. If it is a contract account, the transaction will execute the contract code (but it can also transfer ETH to the contract address).
- **Signature:** The identifier of the sender. This is generated when the sender's private key signs the transaction and confirms the sender has authorised this transaction.
- **Value:** Amount of ETH to transfer from sender to recipient (in Wei).
- **Data:** Optional field to include arbitrary data (for example, a Smart Contract reply with previously requested information).
- **Gas Limit:** The maximum amount of gas units that can be consumed by the transaction.
- **Maximum Fee per Gas:** The maximum amount of gas willing to be paid for the transaction.
- **Max Priority Fee per Gas:** the maximum amount of gas to be included as a tip to the miner.

3.5 Blocks

Blocks are groups of transactions with a hash of the previous block. This links blocks together in an ordered way, like a chain. There are two main reasons on why it was decided this specific usage of blocks [40] [41] [42]:

- **Fraud prevention:** As hashes are -cryptographically- derived from the block content, one change in any block in history would invalidate all the following blocks. The rest of the subsequent hashes would be modified too and everyone running the blockchain would notice.
- **Synchronization:** Transaction requests occur dozens of times per seconds. If a new commit to the chain was required for every -valid- transaction, it would be almost impossible to spread the new EVM state on the rest of the nodes before a new valid transaction is committed. Blocks are meant to be generated every 15 seconds (approximately), providing enough time for the status synchronization all over the Ethereum net.

3.5.1 Elements

- **Timestamp:** The time when the block was mined.
- **BlockNumber:** The length of the blockchain in blocks.
- **BaseFeePerGas:** The minimum fee per gas required for a transaction to be included in the block.
- **Difficulty:** The effort required to mine the block. The expected block time is set as a constant and is used to protect the network's security when the miners add more computational power. The

average block time gets compared with the expected block time, and if the average block time is higher, then the difficulty is decreased in the block header. If the average block time is smaller, then the difficulty in the block header will be increased

- **MixHash:** A unique identifier for that block.
- **ParentHash:** The unique identifier for the block that came before (this is how blocks are linked in a chain).
- **Transactions:** The transactions included in the block.
- **StateRoot:** The entire state of the system: account balances, contract storage, contract code and account nonces are inside.
- **TransactionsRoot:** The Keccak 256-bit hash of the root node of the trie structure populated with each transaction in the transactions list portion of the block.
- **ReceiptsRoot:** The Keccak 256-bit hash of the root node of the trie structure populated with the receipts of each transaction in the transactions list portion of the block
- **Nonce:** A hash that, when combined with the mixHash, proves that the block has gone through proof of work.

3.5.2 Block Size

An important note is that blocks are bounded in size. This is because it ensures that blocks can't be arbitrarily large. If blocks could be arbitrarily large, then less performant full nodes would gradually stop being able to keep up with the network due to space and speed requirements. Each block has a target size of 15 million gas, but the size of blocks will increase or decrease depending on network demands. The upper limit is the double of the target size (30 milion gas). The total amount of gas expended by all transactions in the block must be less than the block gas limit.

Please note that block size is measured in gas instead of transactions because not all transactions need the same computational effort. A transaction can range from a "simple" transfer of money from one account to another to a heavy execution of a Smart Contract.

3.6 Gas

Gas refers to the unit that measures the amount of computational effort required to execute specific operations on the Ethereum network. Since each Ethereum transaction requires computational resources to execute, each transaction requires a fee. Gas refers to the fee required to successfully conduct a transaction on Ethereum [43] [44] [45].

Gas fees are paid in Ethereum's native currency, ether (ETH). Gas prices are denoted in GWei, which itself is a denomination of ETH - each GWei is equal to 10^{-9} ETH. For example, instead of saying that your gas costs 0.000000001 ether, you can say your gas costs 1 gwei. The word 'GWei' itself means 'giga-wei', and it is equal to 1,000,000,000 Wei (wei is the smallest unit of ETH, 1 Wei = 10^{-18} ETH).

Every block has a base fee, the minimum price per unit of gas for inclusion in this block, calculated by the network based on demand for block space. As the base fee of the transaction fee is burnt, users are also expected to set a tip (priority fee) in their transactions (this is usually automatically done by wallets). The tip compensates miners for executing and propagating user transactions in blocks. To calculate the total fee on a transaction: Gas units (limit) * (Base fee + Tip). We can see it better with an example: A is going to pay B 1 ETH. The gas limit is 21,000 units, the base fee is 100 GWei and A includes a tip of 10 GWei. We can calculate this as 21,000 * (100 + 10) = 2,310,000 GWei or 0.00231 ETH. When A sends the money, 1.00231 ETH will be deducted from A's account; B will receive 1 ETH and the miner receives the tip of 0.00021 ETH. Base fee of 0.0021 ETH is burned.

3.7 Ethereum Virtual Machine

The analogy of a 'distributed ledger' is usually used to describe blockchains like Bitcoin, which enable a decentralised currency using fundamental tools of cryptography. A cryptocurrency behaves like a 'normal' currency because of the rules which govern what one can and cannot do to modify the ledger (like, for example, a Bitcoin address cannot spend more Bitcoin than it has previously received). These rules are under all transactions on Bitcoin and many other similar blockchains [46] [47] [48] [49].

Ethereum has Ether -its own cryptocurrency- so almost the same rules are applied on transactions in Ethereum blockchain. However, Ethereum has another powerful feature, Smart Contracts. Therefore, the proper analogy to describe Ethereum's blockchain would be a 'distributed state machine'.

3.7.1 EVM Properties

3.7.1.1 Determinism

A program is deterministic when it provides the same output to the same set of inputs. It doesn't matter how many times the code is executed. This is important because decentralised apps or dApps on Ethereum may handle financial transactions involving large amounts of money at any given time. Therefore, it is crucial to know how the code will react in every stage of execution. Determinism is essential to the foundations of the Ethereum Virtual Machine.

3.7.1.2 Terminability

EVM is Turing-complete (its smart contracts should be able to solve any computational problem). But there is no way to tell whether such smart contracts can finish all the given operations within a specific time frame. Therefore, it is essential to put in a terminating mechanism to create exact limits. In Ethereum, the concept of "gas" is used to facilitate traffic. Gas limits are set at the beginning. When these limits are used up, the machine simply stops operations or halts its processing.

3.7.1.3 Isolation

Isolation is important so that the system can contain hacks or bugs within a smart contract. The bug will not spread to the whole computer (the code running on it has no access to other processes on the computer where the EVM node is running). The feature is in place so that such issues or incidents do not affect the underlying protocol.

3.7.2 Elements of the EVM

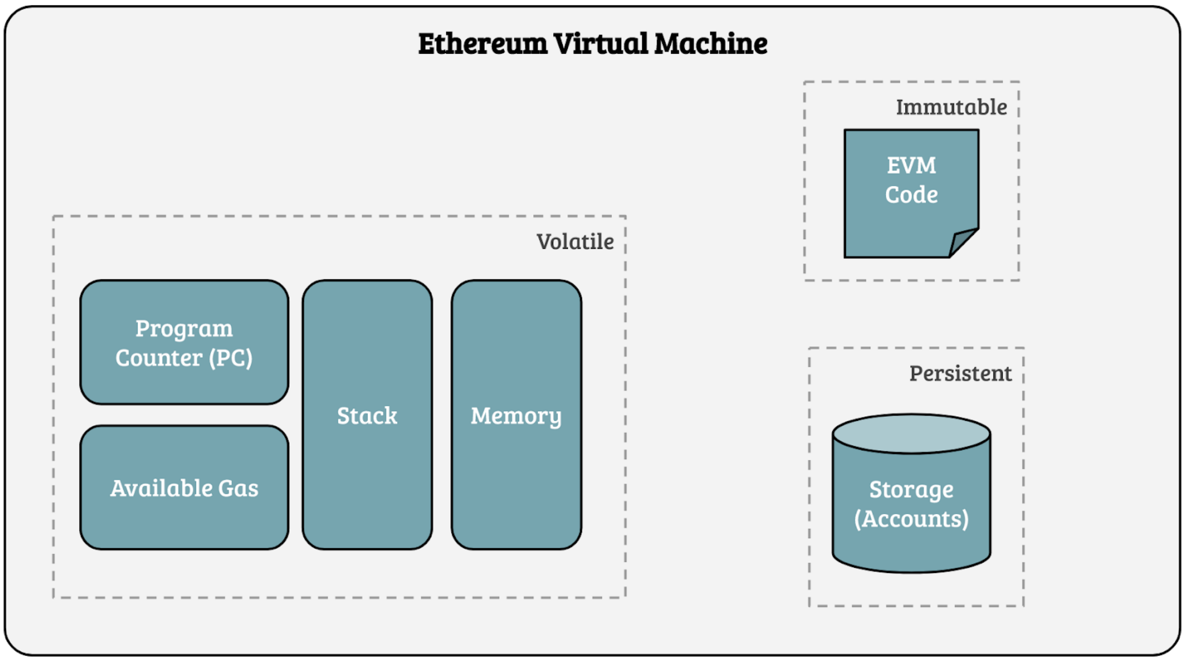


Figure 3-4: Ethereum Virtual Machine Components Diagram [64]

- **EVM Code:** The instructions/operations of the Smart Contracts. Ethereum provides high level programming languages (Solidity, Vyper) to facilitate the creation of Smart Contracts. Then, the code is transformed into Operation Codes (OP_CODES) and finally into bytecode (which is executed by the EVM when an account makes a message call to the contract). Regarding the Operation Codes, there are currently 140 which allows the Turing-completeness. For example, there are usual stack operations (XOR, AND, ADD, SUB), and blockchain specific operations (ADDRESS, BALANCE, KECCAK256, BLOCKHASH).
    - Complete list of OPCODES: https://www.ethervm.io/#opcodes
- **Account Storage:** It is located inside each account and stores the contract state variables. Storage is assigned during the process of creating a contract. You can only modify the content of those variables with a transaction.
- **Memory:** It holds temporary variables. As they only exist in the calling function, memory gets erased between calls. You can address memory at byte level. However, the limit of one reading is 256 bits, and writes can be of 8 or 256 bits in width. You need to pay in gas to expand memory. It scales quadratically, and the more it grows, the more it costs.
- **Stack:** Ethereum virtual machine specification defines it as a stack machine. The stack is where the computations happen. This can include Smarts Contracts, Ethereum nodes, client APIs and end-users applications.
- **Available Gas:** When a transaction is requested, a gas limit is set. Gas is going to be spent in each computational operation. If, eventually, there is no more gas, the rest of the operations will be cancelled.
- **Program Counter:** The register that contains the location of the instruction that is being executed. When the instruction is executed, it will point to the next instruction.

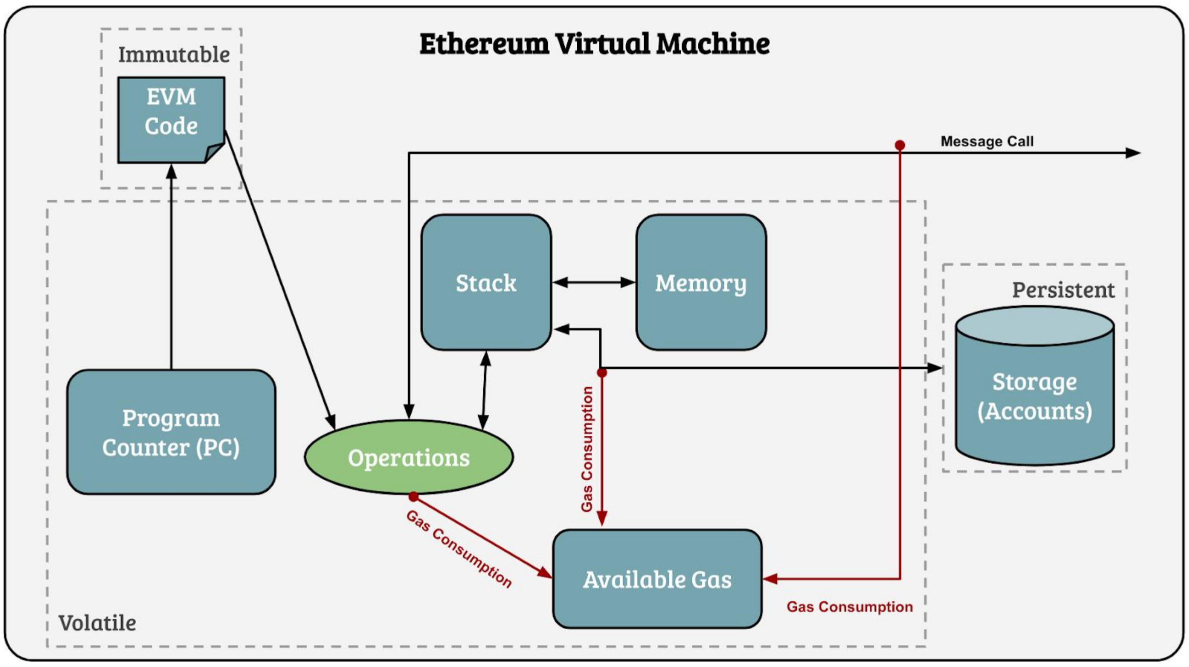


Figure 3-5: Ethereum Virtual Machine Components Interactions Diagram [61]

3.8 Nodes and Clients

"Node" refers to a running piece of client software. The client is a piece of software capable of establishing a p2p communication channel with other clients, signing and broadcasting transactions, mining, deploying and interacting with smart contracts [50] [51] [52] [53] [54].

There are three different types of nodes - light, full and archive. There are also options of different sync strategies which enables faster synchronisation time (Synchronisation refers to how quickly it can get the most up-to-date information on Ethereum's state).

The formal definition of the functionality an Ethereum node must follow is defined in the ethereum yellow paper. The yellow paper defines the required functions of nodes on the network, the mining algorithm, private/public key ECDSA parameters. It defines the entirety of features which make nodes fully compatible Ethereum clients. Many Ethereum clients exist, in a variety of programming languages such as Go, Rust, JavaScript, Python, C# .NET and Java.

3.8.1 Full Node

Full nodes store full blockchain data. They participate in block validation, verifying blocks that are broadcasted onto the network. That is, they ensure that the transactions contained in the blocks (and the blocks themselves) follow the rules defined in the Ethereum specifications. They maintain the current state of the network (as defined according to the Ethereum specifications). For example, if A tries to send 100 Ether to B but A has 0 ethers and a block includes this transaction, the full nodes will realise this does not follow the rules of Ethereum and reject that block as invalid.

All states can be derived from a full node (except very old ones, that are requested to archive nodes). They serve the network and provide data on request (mostly to light nodes).

3.8.2 Light node

They only store the header chain (not the whole blockchain) and request what they need to Full or Archive nodes. That means they can verify the validity of the data against the state roots in the block headers, but they do not participate in the consensus (they can not be miners nor validators).

Light nodes are useful for low-capacity devices, such as embedded devices or mobile phones, which can't afford to store gigabytes of blockchain data.

3.8.3 Archive node

Archive nodes store everything kept in the full node and build an archive of historical states. They are needed when querying an account balance at block #4,000,000. They can also be useful to reliably test your own transactions set without mining them using OpenEthereum.

These data represent units of terabytes which makes archive nodes less attractive for average users but can be handy for services such as block explorers, wallet vendors, and chain analytics.

Syncing clients in any mode other than archive will result in pruned blockchain data. This means, there is no archive of all historical states, but the full node can build them on demand.

3.9 Network

Networks are different Ethereum environments you can access for development, testing, or production use cases. They do not interact with each other [55] [56] [57].

3.9.1 Public Networks

Public networks are accessible to anyone in the world with an internet connection. Anyone can read or create transactions on a public blockchain and validate the transactions being executed.

3.9.1.1 Mainnet

Mainnet is the primary public Ethereum production blockchain, where actual-value transactions occur on the distributed ledger. For example, when people and exchanges discuss ETH prices, they're talking about Mainnet ETH.

3.9.1.2 Testnets

In addition to Mainnet, there are public testnets. These are networks used by protocol developers or smart contract developers to test both protocol upgrades as well as potential smart contracts in a production-like environment before deployment to Mainnet.

Most testnets use a Proof of Authority consensus mechanism. This means a small number of nodes are chosen to validate transactions and create new blocks – staking their identity in the process. It is hard to incentivise mining on a Proof of Work testnet which can leave it vulnerable.

- **Görli:** A Proof of Authority testnet that works across clients.
- **Kovan:** A Proof of Authority testnet for those running OpenEthereum clients.
- **Rinkeby:** A Proof of Authority testnet for those running Geth client. It is deprecated as of today.
- **Ropsten:** A Proof of Work testnet. This means it's the best like-for-like representation of Ethereum. Traditionally has been the “main” testnet; it seems that it will be deprecated during summer 2023, as The Merge will be completed.

Regarding Proof of Stake testnets there are a few (Pyrmont, Prater) but, as of today, you can only register as Staker. There is nothing open for testing Smart Contracts.

3.10 Consensus Mechanism

Consensus mechanisms are protocols to make sure the blockchain nodes are synchronized and agree on which transactions are legitimate. A transaction is added to the blockchain only after it has been validated, which ensures it is the only version of the truth [58] [59] [60] [61] [62] [63] [64].

For Ethereum there are two consensus protocols that are worth understanding: Proof of Work (the current one) and Proof of Stake (the protocol Ethereum is moving to).

3.10.1 Proof of Work Protocol

The Proof of Work protocol -Ethash, for Ethereum specifically- requires miners to go through a race of trial and error to find the nonce for a block, as introduced in section 2.1.3. A miner will repeatedly put a dataset (the block header and the transactions included in it) through a mathematical function to obtain its hash. The key aspect is that not all hashes are valid, they must be under a certain threshold so the block is mined; the higher the block difficulty, the more restrictive the set of valid hashes. The only parameter that the miner is able to change on a block (for a specific set of transactions) is the nonce; it will have to find a nonce that provokes a valid hash.

The objective of PoW is to extend the chain; the longest chain is most believable as the valid one because it has the most computational work done. It is almost impossible to create new blocks that erase transactions, create fake ones, or maintain a second chain, a malicious miner would need to always solve the block nonce faster than everyone else. To consistently create malicious yet valid blocks, over 51% of the network mining power would be required to beat everyone else.

A major criticism of PoW is the amount of energy required to keep the network safe. To maintain security and decentralization, Ethereum on Proof of Work consumes an average of 73.2 TWh annually, (the energy equivalent of a country like Austria).

3.10.2 Proof of Stake Protocol

In Proof of Stake, validators explicitly stake capital in the form of ether into a smart contract on Ethereum. The validator is responsible for checking that new blocks propagated over the network are valid and occasionally creating and propagating new blocks themselves. This staked ether then acts as collateral that can be destroyed if the validator behaves dishonestly or lazily.

To become a validator, a user must deposit 32 ETH into the deposit contract and run three separate pieces of software: an execution client, a consensus client, and a validator. Validators receive new blocks from peers on the Ethereum network, and the validator has to re-execute all of its transactions to check them. The validator then sends a vote in favour of that block across the network.

Whereas under PoW the timing of blocks is determined by the mining difficulty, in PoS the tempo is fixed. Time in Proof of Stake Ethereum is divided into slots (12 seconds) and epochs (32 slots). One validator is randomly selected to be a block proposer in every slot. This validator is responsible for creating a new block and sending it out to other nodes on the network. Also in every slot, a committee of validators is randomly chosen, whose votes are used to determine the validity of the block being proposed.

3.11 Token Standards

Ethereum tokens are digital assets that can be built on top of the Ethereum blockchain. They can be -potentially- anything: skills of a character in a game, a new cryptocurrency, reputation points in an online platform, lottery tickets or some collectible item. You can see a full list of them on https://etherscan.io/tokens [65] [66] [67] [68].

There is a symbiotic relationship between Ethereum and its tokens: tokens benefit from Ethereum's existing infrastructure instead having to build an entirely new blockchain. Ethereum's benefit is that tokens strengthen the ecosystem by driving demand for ETH, needed to power the smart contracts.

Depending on the token’s nature, there are different standards.

3.11.1 ERC-20: Fungible Tokens

Fungible tokens they have a property that makes each token be the same in type and value as another token. Some examples can be currencies or reputation points.

ERC-20 was proposed by developer Fabin Vogelstellar in 2015 as a way to standardize the tokens within smart contracts on the Ethereum blockchain. It contains several functions and events that a token must implement:

- **TotalSupply**: The total number of tokens that will ever be issued
- **BalanceOf**: The account balance of a token owner's account
- **Transfer**: Automatically executes transfers of a specified number of tokens to a specified address for transactions using the token
- **TransferFrom**: Automatically executes transfers of a specified number of tokens from a specified address using the token
- **Approve**: Allows a spender to withdraw a set number of tokens from a specified account, up to a specific amount
- **Allowance**: Returns a set number of tokens from a spender to the owner
- **Transfer**: An event triggered when a transfer is successful (an event)
- **Approval**: A log of an approved event (an event)

3.11.2 ERC-721: Non-Fungible Tokens

Non-fungible tokens are used to identify something or someone in a unique way. They can only have one official owner at a time and they are secured by the Ethereum blockchain; the record of ownership can not be modified or or copy/paste a new NFT into existence. These tokens are not interchangeable for other tokens because they have unique properties.

The ERC-721 introduces a standard for NFT, all NFTs have a ‘uint256’ variable called ‘tokenId’. For any ERC-721 Contract, the pair ‘contract address’, ‘uint256 tokenId’ must be globally unique.

4 MACHINE LEARNING

By far, the greatest danger of Artificial Intelligence is that people conclude too early that they understand it

- Eliezer Yudkowsky -

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Its main purpose is to develop computer programs that can access data and use it to learn for themselves.

4.1 What is Machine Learning?

The machine learning process begins with observations of the input data, such as training data or knowledge graphs. It looks for patterns in data so it can later (with a different input data) make inferences based on the examples provided. Through the use of statistical methods, algorithms are trained to make classifications or predictions [69] [70] [71] [72].

4.1.1 Machine Learning Introduction

Machine Learning is the most widely used branch of computer science nowadays. It is used by many companies for performing complex data analysis and automating tasks. It is behind abundant market applications and devices such as intelligent assistants (Siri, OK Google), fitness trackers (FitBit) or personalised suggestions (suggested films based on previously viewed films). The purpose of this section is to briefly introduce Machine Learning.

4.1.1.1 History

We can mark the Machine Learning history starting point in 1943, with the first mathematical model of neural networks presented in the scientific paper "A logical calculus of the ideas immanent in nervous activity" by Walter Pitts and Warren McCulloch. Then, in 1949, the book “The Organization of Behaviour” by Donald Hebb was published. The book had theories on how behaviour relates to neural networks and brain activity and would go on to become one of the monumental pillars of machine learning development [73] [74] [75] [76].

The first computer learning program was developed in 1952 by Arthur Samuel [77]. It was the game of checkers; the IBM computer improved at the game every time it played. The mechanism was to study which moves lead to winning, and they were incorporated later to its strategy. Then, in 1957, Frank Rosenblatt designed the first neural network for computers - the perceptron - which simulated the thought processes of the human brain [78].

The next step forward in Machine Learning will be in 1967, when the “nearest neighbor” algorithm was written [79]. This algorithm allowed computers to start using -very basic- pattern recognition. More than a decade later, in 1986, the concept of Explanation Based Learning (EBL) was introduced by Gerald Dejong [80]. A computer was able to analyse training data and create a general rule to discard non-relevant data. In the 1990s work on machine learning shifted from a knowledge-driven approach to a data-driven approach. Scientists began creating programs to analyse large amounts of data and obtain conclusions from the results.

The term “Deep Learning” was coined in 2006 by Geoffrey Hinton [81] to explain new algorithms that let computers distinguish objects and text in images and videos. In 2010 Microsoft presented their Kinect technology (it permitted people to interact with the computer through movements and gestures). In 2014, Facebook developed DeepFace, a software algorithm that is able to recognize people in photos and videos, like humans can.

In 2016 Google’s artificial intelligence algorithm beat a professional player at the Chinese board game Go, [82] which is considered the world’s most complex board game and is many times harder than chess. Waymo introduced completely autonomous taxis in the city of Phoenix in 2017 [83]. In 2020 open AI announced a natural language processing algorithm (GPT-3) with an ability to generate human-like text when given a prompt [84].

From now on, the effort on improving machine learning algorithms will continue to help to make predictions from data sets. This function is immensely important as it allows algorithms to discover hidden patterns, helping businesses understand their market and customers better.

4.1.1.2 Clarifying Basic Concepts

Along with Machine Learning there are some concepts such as Deep Learning, Artificial Intelligence, Neural Network and Data Science. As per their usage, there is still some confusion and they may seem to be synonyms. They are highly associated but they are not interchangeable; they are all sub-branches of one another. Here is a diagram of their relation [85] [86] [87] [88] [89] [90]:

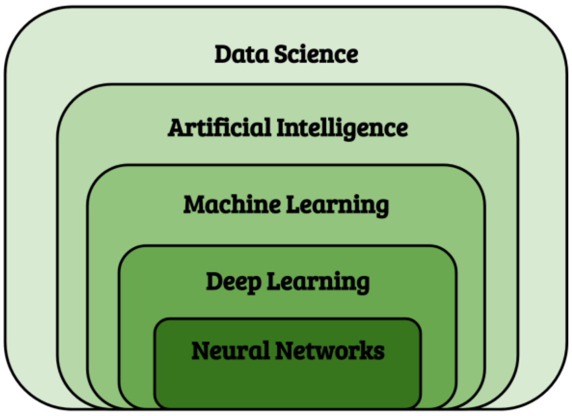


Figure 4-1 Data Science Branches Diagram [87]

Data science would be the field that uses scientific methods, processes, algorithms and systems to extract valuable information from potentially noisy, structured and/or unstructured huge amounts of data, usually with the purpose of applying this knowledge onto a wide range of application domains. It is considered a data scientist who creates programming code and combines it with statistical knowledge to create insights from data.

Artificial intelligence (AI) is, at its simplest form, a field which combines computer science and robust datasets to reach problem-solving. The purpose is to build smart machines with the ability of performing tasks that typically require human intelligence.

Machine learning is one way to use Artificial Intelligence. According to AI pioneer Arthur Samuel, it is “the field of study that gives computers the ability to learn without explicitly being programmed”. We can compare ML with classical programming. In classical programming, the computer is provided with the input and the algorithm in order to obtain the output. On the contrary, in machine learning the computer is supplied with a dataset and associated outputs. Then, the computer learns and generates an algorithm that describes the relationship between the two.

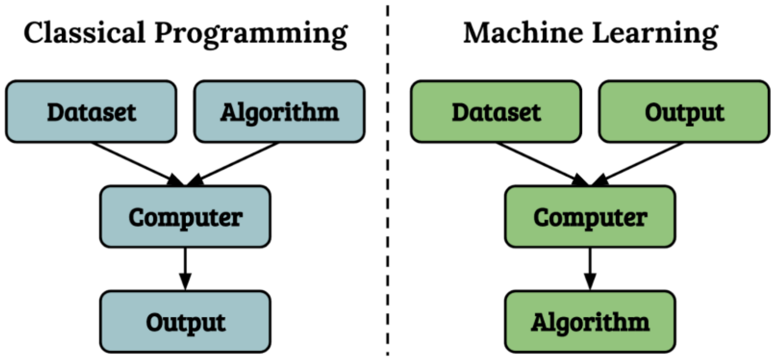


Figure 4-2: Classical Programming vs Machine Learning Diagram

Deep Learning is a sub-field of machine learning. Its main feature is that it automates the human intervention required in ML. Machine Learning is more dependent on data scientists, they need to provide structured data as input. By contrast, Deep Learning is able to work with non-structured data and obtain valuable information from it.

Lastly, Artificial Neural Networks are a subset of Deep Learning (note: there are some authors with an opposite opinion on this [91]). They are computing systems that reflect the behaviour of the human brain, allowing computer programs to recognize patterns and solve problems. They are based on a sum of connected nodes (called perceptrons), modelling neurons in a biological brain. Each connection can transmit a signal to other neurons (imitating the synapses in a biological brain). When an artificial neuron receives a signal, it can process it and then signal neurons connected to it. The signal is actually a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. Neurons and connections typically have a weight that adjusts as learning proceeds.

4.2 Types of Machine Learning tasks

Machine learning implementations can be classified into three major categories, depending on the kind of data that is going to be predicted. These categories are supervised (uses labeled data to help with the prediction), unsupervised (it does not use labeled data) and reinforcement (trial and error, after every prediction the system is told whether it was correct or wrong). It also exists the semi-supervised category, where only part of the dataset is labeled [92] [93] [94] [95].

4.2.1 Supervised

In this approach of machine learning, data scientists need to supply algorithms with labeled training data and define the variables they want the algorithm to assess for correlations. Both the input and the output of the algorithm is specified (with the usage of labeled inputs and outputs, the model can measure its accuracy and learn over time).

This approach can be compared to human learning under the supervision of a teacher. The teacher provides good examples for the student to memorise, and the student then subtracts general rules from these specific examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

We can separate supervised learning into two types: classification and regression, the difference between both is how they are used for different machine learning problems. On one hand, the classification approach tries to accurately assign test data into specific categories, such as separating apples from oranges (given one or more inputs a classification model will try to predict the value of one or more outcomes). Or, a more useful example, supervised learning algorithms are being used to filter spam emails.

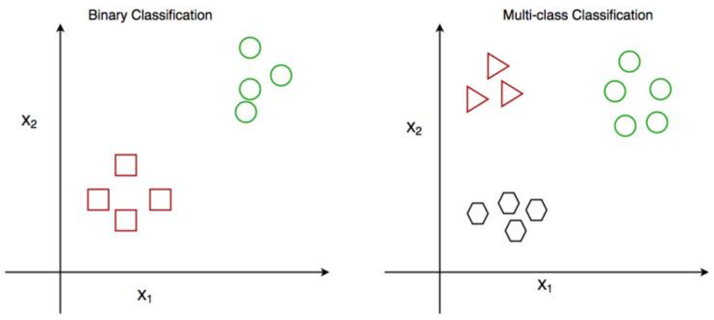


Figure 4-3: Examples of classification approach graphs [96]

On the other hand, the regression approach uses an algorithm to understand the relationship between dependent and independent variables. If classification algorithms are used to predict/Classify discrete values (Spam or Not Spam), regression algorithms are used to predict continuous values such as price, salary, age, etc. They can be used for predicting sales revenue projections for a given business, for example.

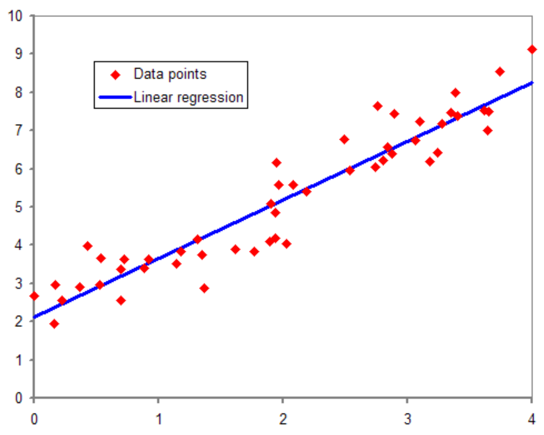


Figure 4-4: Example of a regression approach graph [96]

4.2.2 Unsupervised

Unsupervised machine learning is similar to the methods that humans use to figure out that certain objects are from the same class by observing the degree of similarity between them. For example, a baby knows and recognizes the family dog. Later, a family friend brings along a dog and tries to play with the baby. The baby has not seen this animal before, but she identifies it as a dog (it is able to recognize some features like the form of the ears, walking on four legs, the mouth etc.).

This type of machine learning involves algorithms that train on unlabeled data; the algorithm scans through datasets looking for any meaningful connection. These algorithms discover hidden patterns in data without the need for human intervention (that is the reason why it is called “unsupervised”). In contrast to supervised learning (data is tagged by an expert), unsupervised methods exhibit self-organization that captures patterns as probability densities or a combination of neural feature preferences.

This model is utilised for three main tasks: clustering, association and dimensionality reduction. The idea of clustering is grouping data based on their similarities/differences. It is possible to specify how many clusters the algorithm should identify (it allows to adjust the granularity of these groups). The purpose of association is finding relationships between variables in a given dataset. It determines the set of items that occur together in the dataset. For example, it can make marketing strategies more effective; people who buy X item (suppose a bread) also tend to purchase Y (Butter/Jam) item. Lastly, dimensionality reduction is for reducing the number of data inputs to a manageable size preserving the integrity of the dataset. It is commonly used in the preprocessing data stage.

4.2.3 Reinforcement

For this kind of machine learning, data scientists program an algorithm to complete a task and then they provide to it positive or negative cues as it works out how to complete a task (its goal is to maximise the positives cues). In the human world, it is similar to learning by trial and error. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected.

This task englobes all problems that involve making a sequence of decisions. Some examples can be controlling the motor of a robot so that it is able to run, making business decisions (pricing and inventory management), or even playing video games.

The main challenge in reinforcement learning lies in preparing the simulation environment, which is highly dependent on the task that it is wanted to be accomplished. When the objective is to master a simple game (chess, old Atari games), preparing the simulation environment is not complex. Nevertheless, when the purpose is to obtain a model capable of driving an autonomous car, building a realistic simulator is crucial before letting the car ride on the street. The model has to figure out how to brake or avoid a collision in a safe environment. Transferring the model out of the training environment and into the real world is the most complicated task.

4.3 How do we measure the performance of a Machine Learning model?

The next step after implementing a machine learning algorithm is to inquire how effective the model is. For example, just by considering if the system has predicted well the ‘positives’ and the ‘negatives’ (the supervised → classification type of ML model) we can obtain 6 useful measures [97] [98] [99].

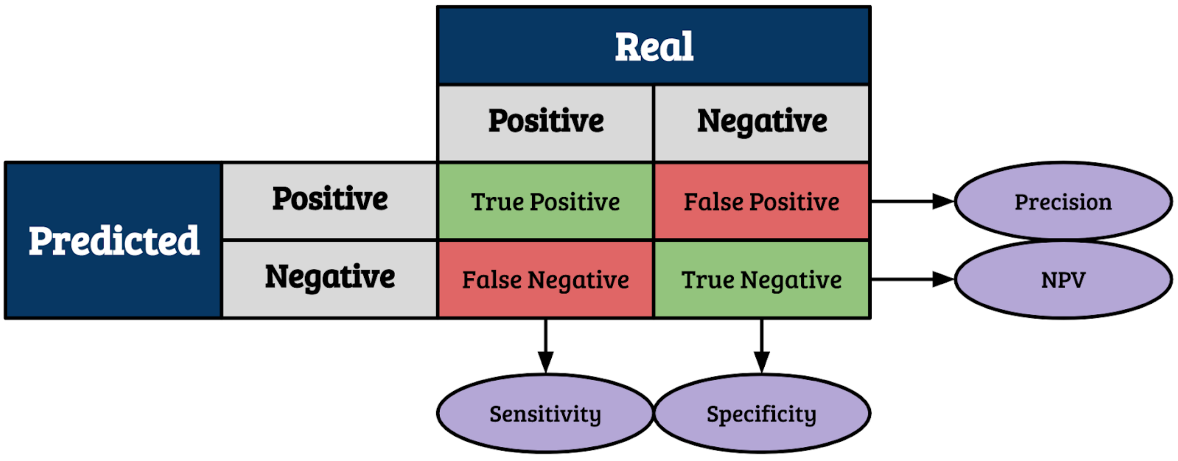


Figure 4-5: Measures of ML model performance

Please note that, for example, in the case of predicting if an email is SPAM or not:

- **True Positive (TP):** The mail was classified as SPAM, and it was SPAM.
- **True Negative (TN):** The mail was classified as not SPAM, and it was not SPAM.
- **False Positive (FP):** The mail was classified as SPAM, and it was not SPAM.
- **False Negative (FN):** The mail was not classified as SPAM, and it was SPAM.

4.3.1 Accuracy and Error Rate

When we use the term ‘accuracy’ it really means ‘classification accuracy’. It is the ratio of the number of correct predictions to the total number of input samples (the total numbers of predictions made):

$$Acc = \frac{TP+TN}{TP+FP+TN+FN}$$

Figure 4-6: Accuracy formula

It is one of the most commonly used metrics to judge a model and can be a poor indicator of the performance. It will work well only if there are a similar number of samples belonging to each class. However, what can happen when classes are not balanced? Imagine the scenario where the 99% elements are class A and the 1% are class B; with a model that always predicts all elements as class A, we would obtain a 99% of accuracy in this case. The real problem arises when the cost of misclassification of the minor class samples is very high. For example, a model for cancer detection. The previous model would always give ‘no cancer’ predictions.

We can also use the Error Rate, which is the opposite measure (ratio of the number of wrong predictions):

$$Err = \frac{FP+FN}{TP+FP+TN+FN}$$

Figure 4-7: Error formula

4.3.2 Negative Predictive Value

Negative predictive value is the ratio of true negatives and total negatives predicted:

$$NPV = \frac{TN}{TN+FN}$$

Figure 4-8: Negative Predictive Value formula

It can be interpreted as ‘*how much the model is right when obtaining negatives’.* A precision score towards 1 will signify that the model did not miss any false negatives and is able to classify well between correct and incorrect labelling. A low precision score (<0.5) means that the model provoques a high number of false negatives which can be an outcome of imbalanced class or untuned model hyperparameters.

4.3.3 Precision

Precision is the ratio of true positives and total positives predicted:

$$Prec = \frac{TP}{TP+FP}$$

Figure 4-9: Precision formula

It can be interpreted as ‘*how much the model is right when obtaining positives’.* A precision score towards 1 will signify that the model did not miss any true positives and is able to classify well between correct and incorrect labelling. A low precision score (<0.5) means that the model provoques a high number of false positives which can be an outcome of imbalanced class or untuned model hyperparameters.

4.3.4 Recall

This metric is also known as ‘Sensitivity’, ‘True Positive Rate’ and ‘Hit Rate’. It measures the percentage of positive instances out of the total actual positive instances:

$$Rec = \frac{TP}{TP+FN}$$

Figure 4-10: Recall formula

It can be interpreted as ‘*how many extra positive ones missed the model when it showed the positive ones’.* A Recall score near 1 will signify -analogously to precision metric- that the model did not miss any true positives and is able to classify well between correctly and incorrectly labelling. The difference with precision metric is when a low score (<0.5) is obtained. It would mean that the model has a high number of false negatives (which can be an outcome of imbalanced class or untuned model hyperparameters).

4.3.5 Specificity

Percentage of negative instances out of the total actual negative instances:

$$Spe = \frac{TN}{TN+FP}$$

Figure 4-11: Specificity formula

It is similar to recall, but the focus is on the negative instances (‘*how many extra negative ones missed the model when it showed the negative ones*’). If the score is near 1 will mean that the model did not miss any true negative and it is able to classify well between correctly and incorrectly labelling. The difference with the recall metric is when a low score (<0.5) is obtained. It would mean that the model has a high number of false positives.

4.4 Machine Learning competitions

Machine Learning competitions have become very popular. Participating in machine learning competitions (whether as a beginner or as an advanced practitioner) can provide interesting benefits. One of the main reasons for their popularity is allowing people to practise what they have learnt; outside formal education or work experience it is complicated. They can also be used for building a portfolio to expose your skills and, of course, to earn money in case of winning a prize.

There are several sites that host machine learning competitions and most of them follow the same structure. Usually, they provide a wide variety of competitions ranging in skill level, including problems for practising (non-rewarded) and rewarded competitions problems. Normally, the competitions consist of a statement to resolve, and they provide a dataset. The majority of them allow unlimited entries which means that it is possible to keep refining the model and improving the score.

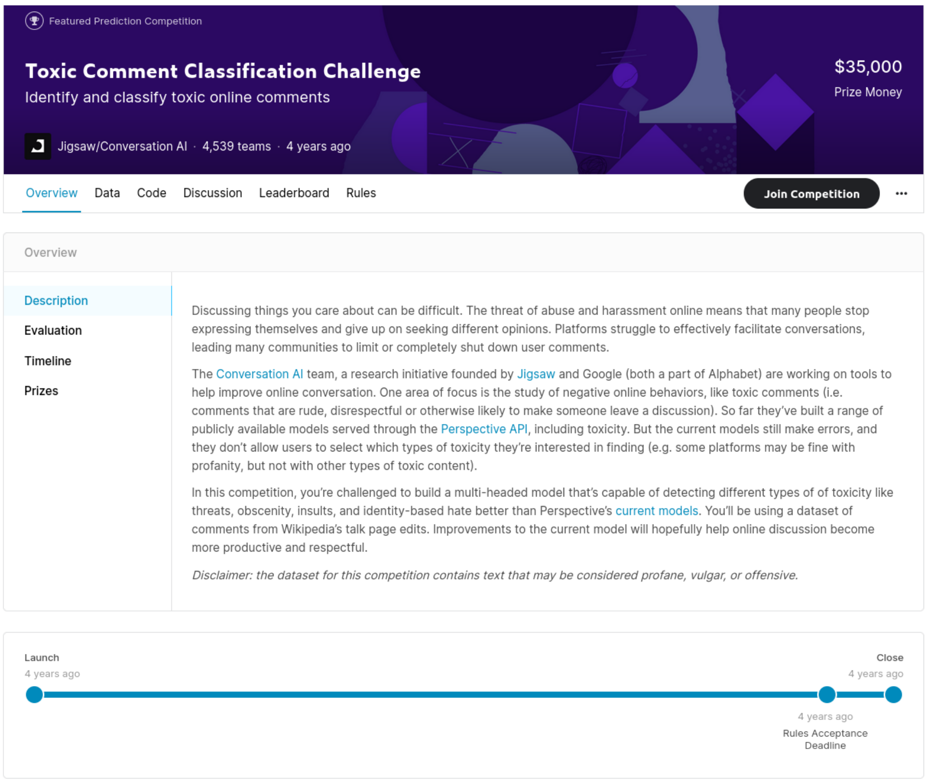
4.4.1 Kaggle

Kaggle is probably the most widely known machine learning competition website. The types of data science problems posted on Kaggle can be anything from attempting to predict cancer occurrence by examining patient records to analysing sentiment to evoke by movie reviews and how this affects audience reaction [100].

The host of the competition is responsible for preparing the data and a detailed description of the problem. The participants submit their models and all the work is shared on the platform with the intention of inspiring new ideas to achieve better benchmarks. Before the deadline expires, the competitors are allowed to make revisions on their submissions as they deem fit (allowing for revisions elevates the level of accuracy and precision). When the deadline for a competition expires, the host pays the prize money to the winner. Hosts have the sole ownership and royalty-free license to use the winning entry with all intellectual property [100].

Lets see an example of a real competition - ‘Toxic Comment Classification Challenge’: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

In the Overview tab we can see the total amount of prizes offered, a detailed description of the issue to resolve and the timeline (when was it launched, the acceptance deadline and the closure):


Featured Prediction Competition
Toxic Comment Classification Challenge
Identify and classify toxic online comments
Jigsaw/Conversation AI · 4,539 teams · 4 years ago
$35,000
Prize Money
Overview
Data
Code
Discussion
Leaderboard
Rules
Join Competition
Overview
Description
Evaluation
Timeline
Prizes

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

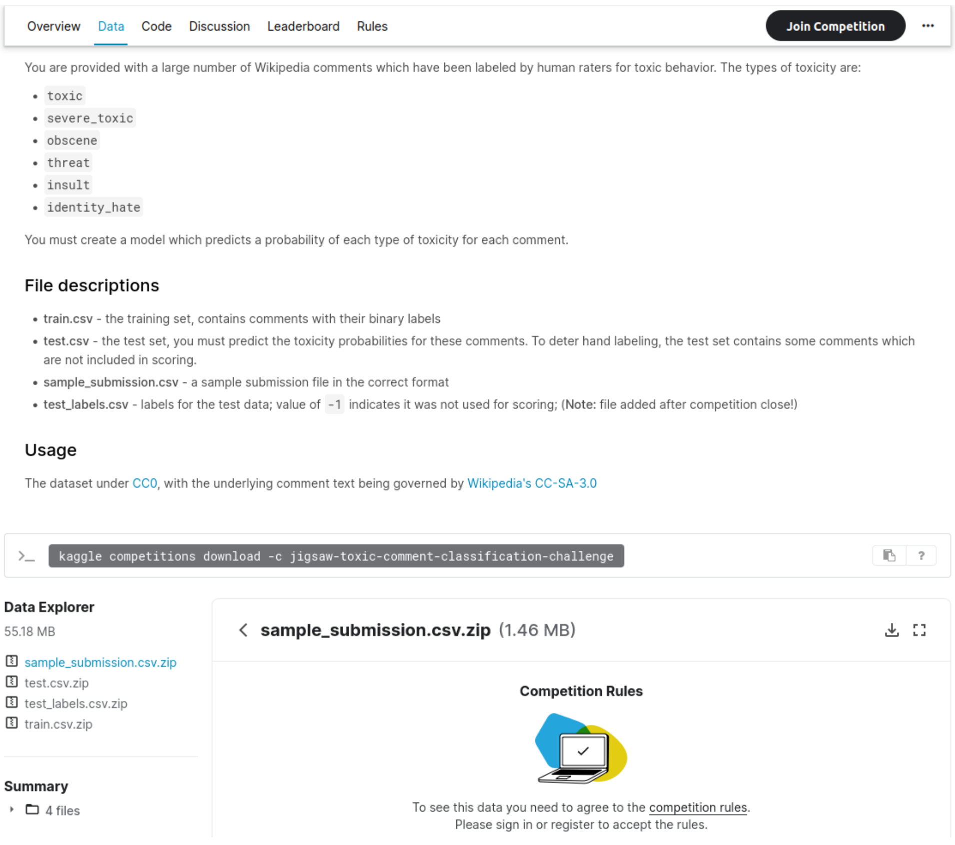
The Conversation AI team, a research initiative founded by Jigsaw and Google (both a part of Alphabet) are working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). So far they've built a range of publicly available models served through the Perspective API, including toxicity. But the current models still make errors, and they don't allow users to select which types of toxicity they're interested in finding (e.g. some platforms may be fine with profanity, but not with other types of toxic content).

In this competition, you're challenged to build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. You'll be using a dataset of comments from Wikipedia's talk page edits. Improvements to the current model will hopefully help online discussion become more productive and respectful.

Disclaimer: the dataset for this competition contains text that may be considered profane, vulgar, or offensive.

Figure 4-12: Kaggle – Contest overview tab

In the Data tab we can read a brief description of the objective and of the necessary files:



You are provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

- `toxic`
- `severe_toxic`
- `obscene`
- `threat`
- `insult`
- `identity_hate`

You must create a model which predicts a probability of each type of toxicity for each comment.

File descriptions

- **train.csv** - the training set, contains comments with their binary labels
- **test.csv** - the test set, you must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.
- **sample_submission.csv** - a sample submission file in the correct format
- **test_labels.csv** - labels for the test data; value of `-1` indicates it was not used for scoring; (**Note:** file added after competition close!)

Usage

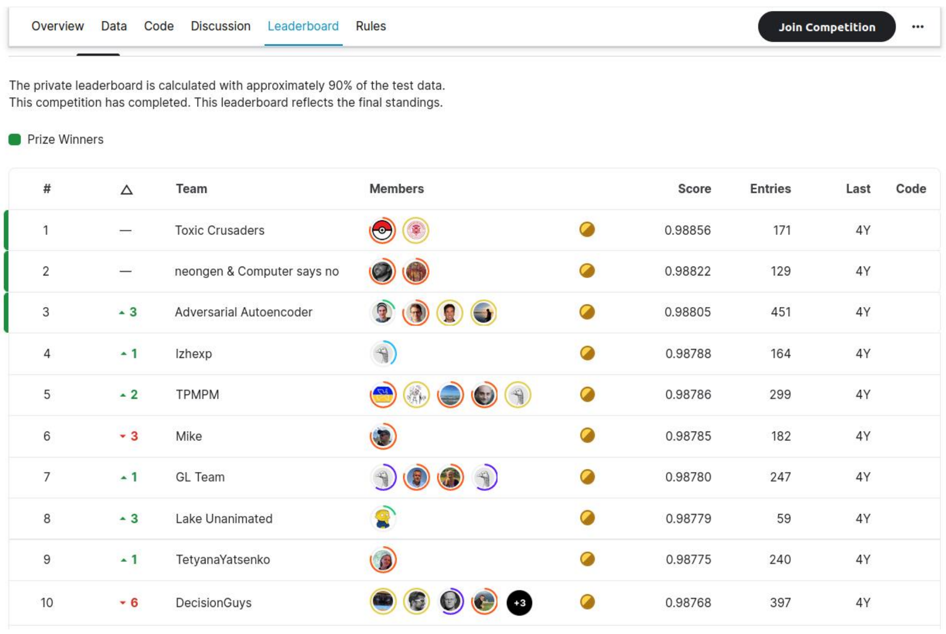
The dataset under CC0, with the underlying comment text being governed by Wikipedia's CC-SA-3.0

```
kaggle competitions download -c jigsaw-toxic-comment-classification-challenge
```

To see this data you need to agree to the competition rules.
Please sign in or register to accept the rules.

Figure 4-13: Kaggle – Contest Data tab

In the Leaderboard tab appears all the participants ordered by score:



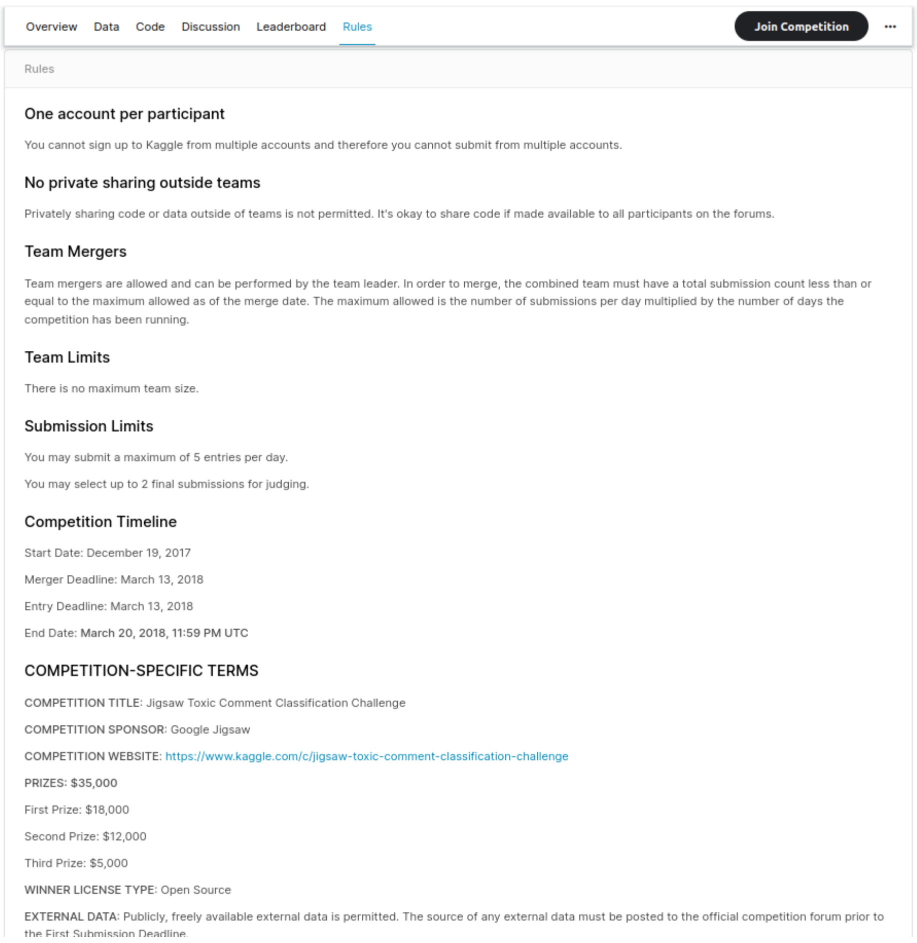
The private leaderboard is calculated with approximately 90% of the test data.
This competition has completed. This leaderboard reflects the final standings.

🟩 Prize Winners

#	△	Team	Members	Score	Entries	Last	Code
1	—	Toxic Crusaders		0.98856	171	4Y	
2	—	neongen & Computer says no		0.98822	129	4Y	
3	▲ 3	Adversarial Autoencoder		0.98805	451	4Y	
4	▲ 1	Izhexp		0.98788	164	4Y	
5	▲ 2	TPMPM		0.98786	299	4Y	
6	▼ 3	Mike		0.98785	182	4Y	
7	▲ 1	GL Team		0.98780	247	4Y	
8	▲ 3	Lake Unanimated		0.98779	59	4Y	
9	▲ 1	TetyanaYatsenko		0.98775	240	4Y	
10	▼ 6	DecisionGuys	+3	0.98768	397	4Y	

Figure 4-14: Kaggle – Contest Leaderboard tab

Lastly, in the Rules tab we can see a detailed description of the competitions terms, acceptance criteria, prizes and legal basis:



Rules

One account per participant

You cannot sign up to Kaggle from multiple accounts and therefore you cannot submit from multiple accounts.

No private sharing outside teams

Privately sharing code or data outside of teams is not permitted. It's okay to share code if made available to all participants on the forums.

Team Mergers

Team mergers are allowed and can be performed by the team leader. In order to merge, the combined team must have a total submission count less than or equal to the maximum allowed as of the merge date. The maximum allowed is the number of submissions per day multiplied by the number of days the competition has been running.

Team Limits

There is no maximum team size.

Submission Limits

You may submit a maximum of 5 entries per day.

You may select up to 2 final submissions for judging.

Competition Timeline

Start Date: December 19, 2017

Merger Deadline: March 13, 2018

Entry Deadline: March 13, 2018

End Date: March 20, 2018, 11:59 PM UTC

COMPETITION-SPECIFIC TERMS

COMPETITION TITLE: Jigsaw Toxic Comment Classification Challenge

COMPETITION SPONSOR: Google Jigsaw

COMPETITION WEBSITE: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

PRIZES: $35,000

First Prize: $18,000

Second Prize: $12,000

Third Prize: $5,000

WINNER LICENSE TYPE: Open Source

EXTERNAL DATA: Publicly, freely available external data is permitted. The source of any external data must be posted to the official competition forum prior to the First Submission Deadline.

Figure 4-15: Kaggle – Competition detailed description tab

4.4.1.1 Types of Kaggle competitions

Kaggle Competitions are designed to provide challenges for competitors at all different stages of their machine learning careers. As a result, they are very diverse, with a range of broad types [101].

4.4.1.1.1 Getting Started

Getting Started competitions are the easiest, most approachable competitions on Kaggle. These are semi-permanent competitions that are meant to be used by new users just getting their foot in the door in the field of machine learning. They offer no prizes or points. Once a submission is more than two months old, it will be invalidated and no longer count towards the leaderboard. This gives new Kagglers the opportunity to see how their scores stack up against a cohort of competitors rather than many tens of thousands of users.

Some examples of this contests are:

- **Digit Recognizer:** The goal is to correctly identify digits from a dataset of tens of thousands of handwritten images.

- **Titanic:** The objective is to use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

4.4.1.1.2 Playground

Playground competitions are a "for fun" type of competition. They are similar to the 'Getting Started' ones but with one step above in difficulty and some of them can have a small cash prize. These are competitions which often provide relatively simple machine learning tasks and are similarly targeted at newcomers.

Some examples are:

- **Dogs versus Cats:** Create an algorithm to distinguish dogs from cats.

- **New York taxi trip duration:** Share code and data to improve ride time predictions.

4.4.1.1.3 Featured

These are the competitions that Kaggle is probably best known for. These are full-scale machine learning challenges which pose difficult, generally commercially-purposed prediction problems. For example, the offer prize pools are currently going as high as a million dollars.

Some examples of these kind of competition:

- **Allstate Claim Prediction Challenge:** Use customers' shopping history to predict which insurance policy they have purchased.
- **Jigsaw Toxic Comment Classification Challenge:** Predict the existence and type of toxic comments on Wikipedia

4.4.1.1.4 Research

Research competitions feature problems which are more experimental than featured competition problems. Due to their experimental nature, these competitions do not usually offer prizes or points. However, they offer an opportunity to work on problems which may not have a clean or easy solution and which are integral to a specific domain or area in a slightly less competitive environment.

Some examples of research competitions are:

- **Large Scale Hierarchical Text Classification:** Classify Wikipedia documents into one of ~300,000 categories

- **Google Landmark Retrieval Challenge:** Given an image, the objective is to find all the same landmarks in a dataset.

5 Project Description

The thing that I often ask startups on top of Ethereum is, 'Can you please tell me why using the Ethereum blockchain is better than using Excel?' And if they can come up with a good answer, that's when you know you've got something really interesting.

- Vitalik Buterin -

The current procedure on most of the Machine Learning competitions is: Participants upload their results to the competition's site and those results are internally reviewed. Ethereum's Smart Contracts could play a useful role to make transparent the review of the results. Ethereum transactions will remain registered and any person can check them; Smarts Contracts are public and they can automate when and how to review the results of the participants.

5.1 Project objective/goal

The objective of this project is to test if Smart Contracts can be a good alternative for the reviewing process on Machine Learning contests. This would imply the next benefits and drawbacks:

- On the user side:
    - Full transparency
    - Sureness on receiving the prize.
    - Fees should be payed for sending results
- On the administrator's side:
    - It would allow them to manage the contest without maintaining a -potentially powerful- server for the storage and review of the results.
    - Fees should be payed for storing results and for the computational cost of the reviewing.

The administrator would be in charge of creating (defining solution and prize) and closing (calculate a winner and sending the prize) the contests, and the user would only need to send their results.

To test this, we have thought in a Proof of Concept. We have developed an Ethereum Dapp that allows us to manage two mock Machine Learning contests as administrators and to participate in them as contesters. The first of them is *Text from Image*, where the contesters will receive an image and the objective is to obtain the text as accurate as possible. The second contest is *Dog or Cat*, where the contesters will receive a set of cats and dogs images and they will have to determine if the image contains a cat or a dog (on each image).

In order to guarantee the proper functioning, contests will have to follow these steps:

1. Creation → Setting the solution and the prize and marking the contest as active.
2. User participation → During a certain amount of time, users will be allowed to send their results. We have decided that the Admin will end the period manually, but it would have been possible to use a different criteria (like finishing on a specific date, or when there are a chosen number of contesters)
3. Winner/s calculation → That implies no more user participation. There is also the ability of expelling a winner in exceptional cases (f.e. fraud)
4. Sending prize to winner/s → Due to the nature of these contests -the winner is the contester with the biggest score, if there is more than one winner is because they obtained the same score-, the prize will be equally divided among the winners.
5. Reset/Delete contest → It will delete all stored information and mark the contest as inactive.

These phases need to be in that order. That means that, for example, it is not allowed to reset a contest before sending the prize to the winner/s, nor user participation after the winner/s calculation. The own Smart Contract will check all cases and will reject improper transactions.

5.2 Tutorial/Explanation of the App

5.2.1 Application download and Start-Up

The app is stored in this github repository: https://github.com/pedgc/challenge

You will only need to clone the repository and then execute (with python3) the file:

- ChallengeApp.py → If you are an user
- ChallengeAppAdmin.py → If you are the administrator of the contests

Example of the commands for a Linux environment:

```
git clone git@github.com:pedgc/challenge.git

cd challenge/Main

python3 ChallengeApp.py
```

Then, the App will request the secret key of your account and you will be able to use it:

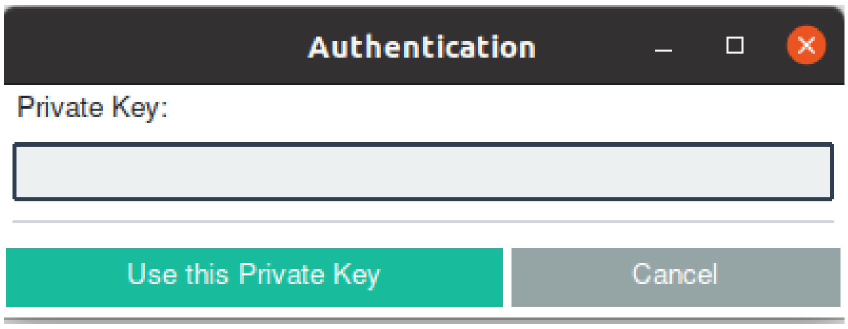
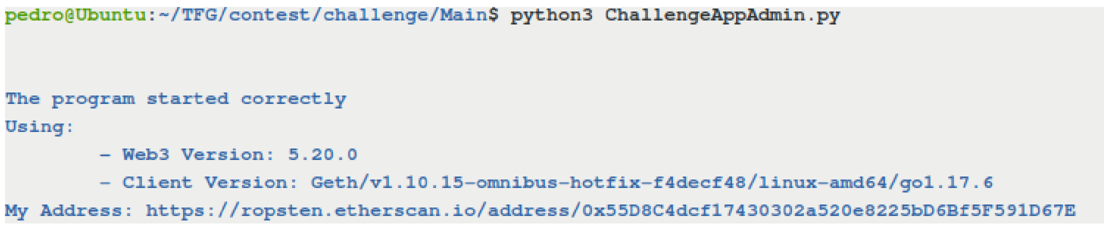


Figure 5-1: Authentication Window

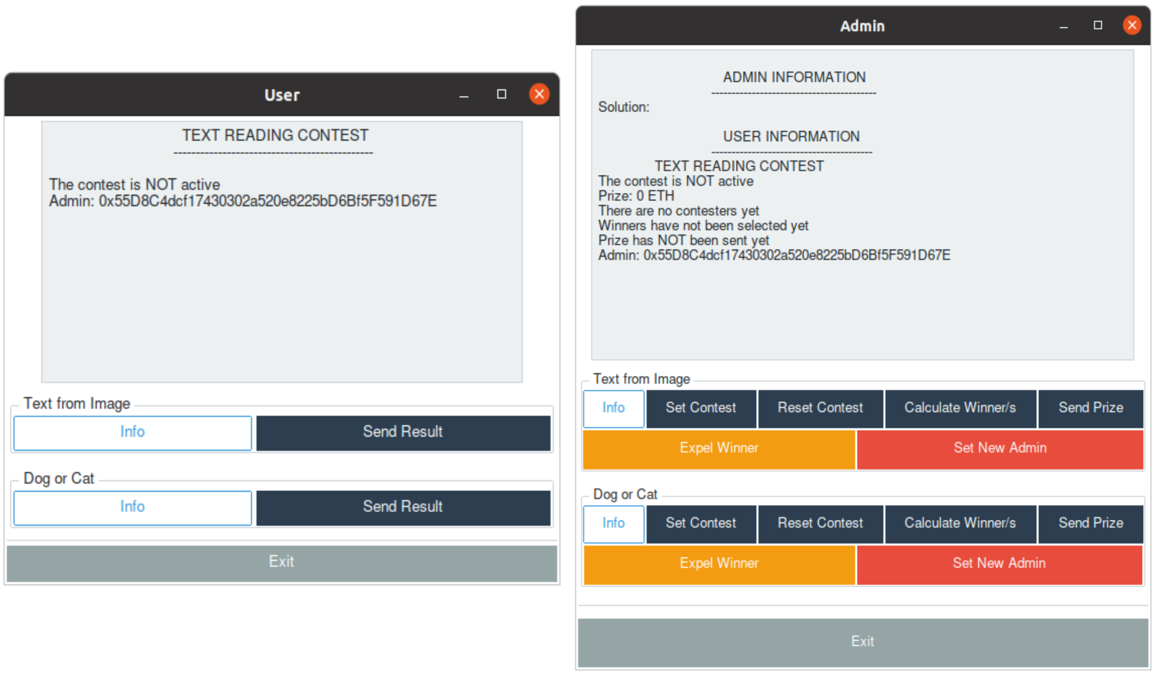
As soon as you provide your private key, you will be able to see in the terminal useful information like the Web3 and Client versions used, and a link to your ethereum account:



```
pedro@Ubuntu:~/TFG/contest/challenge/Main$ python3 ChallengeAppAdmin.py


The program started correctly
Using:
        - Web3 Version: 5.20.0
        - Client Version: Geth/v1.10.15-omnibus-hotfix-f4decf48/linux-amd64/go1.17.6
My Address: https://ropsten.etherscan.io/address/0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-2: Info prompted in console after authentication

Also, a window will pop-up. Depending on if you decided to open the user or de admin app, you will see something like this:



```
TEXT READING CONTEST
-----------------------------------------
The contest is NOT active
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

```
ADMIN INFORMATION
-----------------------------------------
Solution:

USER INFORMATION
-----------------------------------------
TEXT READING CONTEST
The contest is NOT active
Prize: 0 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-3: Main Window: User's App (left) and Admin's App (right)

5.2.1.1 Actions that consume gas

Like with any other Smart Contract, all actions that modify the status of the Contract will require a gas fee. In this case, the only action that requires no gas is *Info* - that will provide the current status of the contest - for both Apps, *User* and *Admin*.

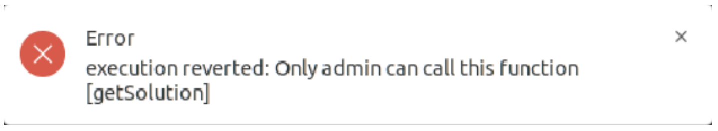
Gas-consuming actions:

- User App
  - Send Result
- Admin App
  - Set Contest
  - Reset Contest
  - Calculate Winner/s
  - Send Prize
  - Expel Winner
  - Set New Admin

5.2.1.2 Security Note

A hacker mind could think of taking advantage of the possibility of using the Admin App to see the solution or to modify something in the contest. This is not possible unless the hacker has the private key of the administrator's account; this Smart Contract checks (in sensitive functions) that the requester's address is the admin address.

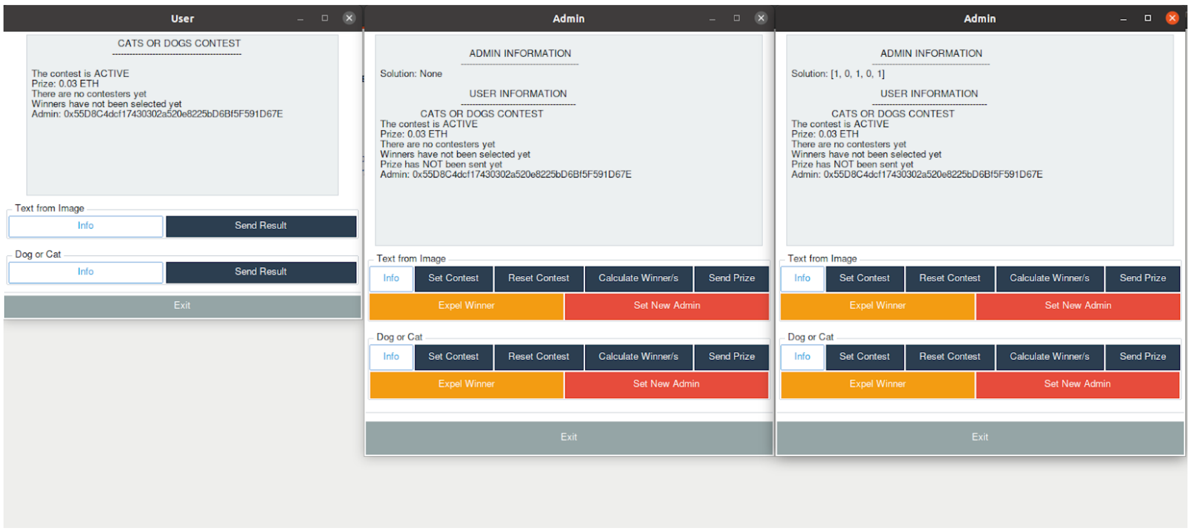
If, for example, the hacker tries to see the solution by clicking the *Info* button, an error message will appear:



```
Error
execution reverted: Only admin can call this function
[getSolution]
```

Figure 5-5: Only Admin notification message

It would not obtain more info than using the User App:



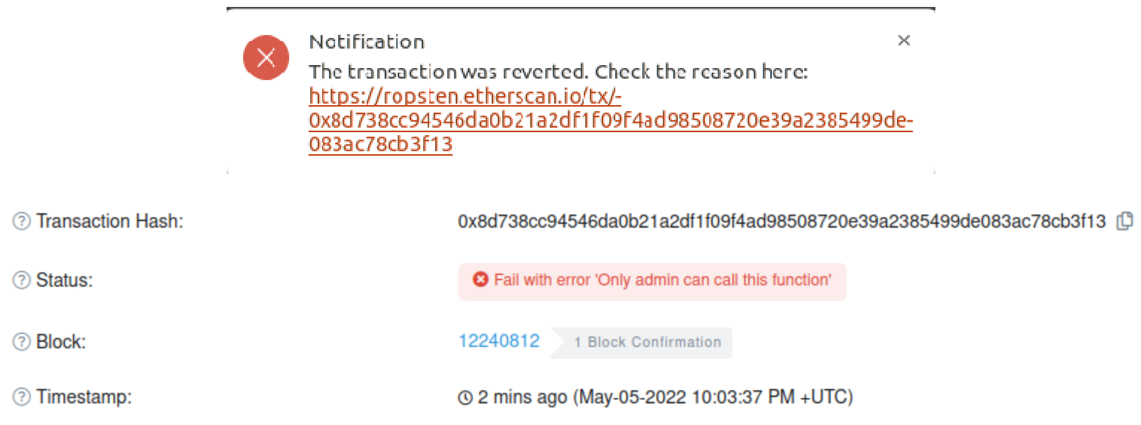
```
CATS OR DOGS CONTEST
--------------------
The contest is ACTIVE
Prize: 0.03 ETH
There are no contesters yet
Winners have not been selected yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

```
ADMIN INFORMATION
-----------------
Solution: None
USER INFORMATION
----------------
CATS OR DOGS CONTEST
The contest is ACTIVE
Prize: 0.03 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

```
ADMIN INFORMATION
-----------------
Solution: [1, 0, 1, 0, 1]
USER INFORMATION
----------------
CATS OR DOGS CONTEST
The contest is ACTIVE
Prize: 0.03 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-6: Obtained info in User app (left) and in Admin windows (center and right)

Something similar would happen if the hacker tries to modify something in the contract:



Notification
The transaction was reverted. Check the reason here: https://ropsten.etherscan.io/tx/-0x8d738cc94546da0b21a2df1f09f4ad98508720e39a2385499de-083ac78cb3f13

Transaction Hash:	0x8d738cc94546da0b21a2df1f09f4ad98508720e39a2385499de083ac78cb3f13
Status:	Fail with error 'Only admin can call this function'
Block:	12240812 1 Block Confirmation
Timestamp:	2 mins ago (May-05-2022 10:03:37 PM +UTC)

Figure 5-7: Reverted transaction (notification and transaction scan)

https://ropsten.etherscan.io/tx/0x8d738cc94546da0b21a2df1f09f4ad98508720e39a2385499de083ac78cb3f13

5.2.2 Application Usage

5.2.2.1 As a User – Participating in contests

This is the menu of the User App. It allows the user to fully participate in both contests (*Text from Image* and *Dog or Cat*):

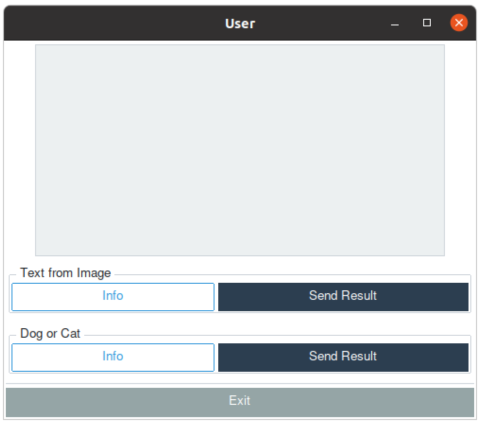
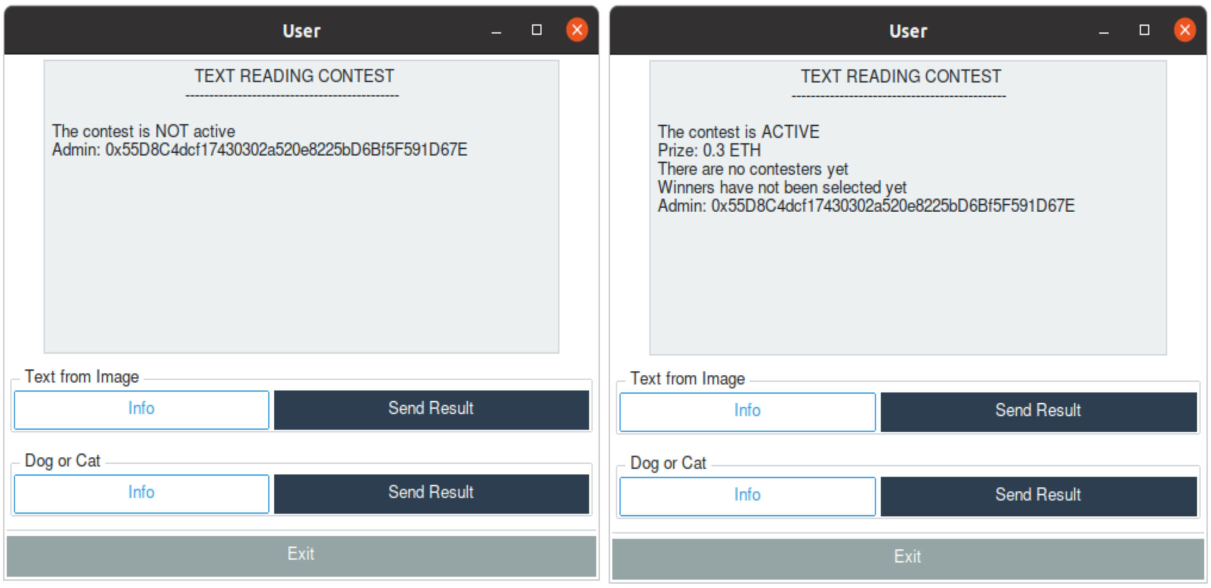


Figure 5-8: User menu

See contest Info

To see the current status of the contest, the user just needs to click on the *Info* button and relevant contest information (like if the contest is active, the prize, how many contesters are etc.) will be displayed.



```
TEXT READING CONTEST
------------------------------------------

The contest is NOT active
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

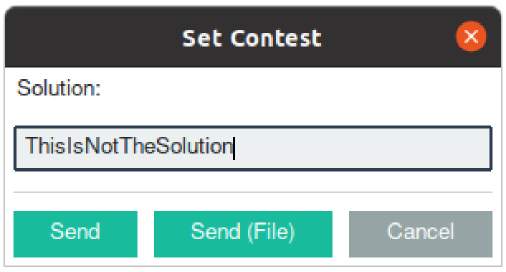
```
TEXT READING CONTEST
------------------------------------------

The contest is ACTIVE
Prize: 0.3 ETH
There are no contesters yet
Winners have not been selected yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-9: Contest info – Not active (left) and active (right)

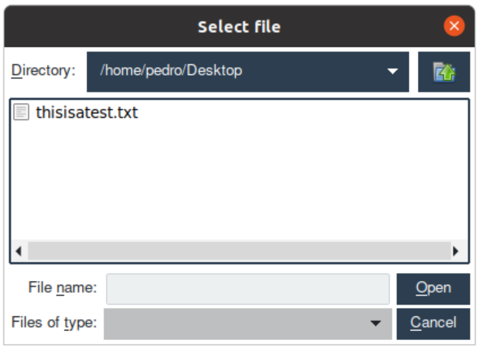
Send Result

To send the solution, the user has to click on the *Send Result* button and then, there are two options:

- To directly write the solution in the box
- To import a txt file (the whole content of the file will be considered as the result)



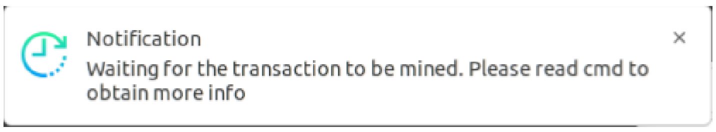
Solution: ThisIsNotTheSolution



Directory: /home/pedro/Desktop

Figure 5-10: Sending solution – Direct input (left) and through text file (right)

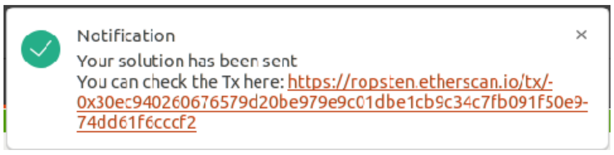
Once the solution is sent, you will receive a notification saying that the transaction needs to be mined:



Notification
Waiting for the transaction to be mined. Please read cmd to obtain more info

Figure 5-11: Waiting for transaction to be mined notification

And once the transaction is mined, you will receive a notification saying that your solution is stored in the Smart Contract:

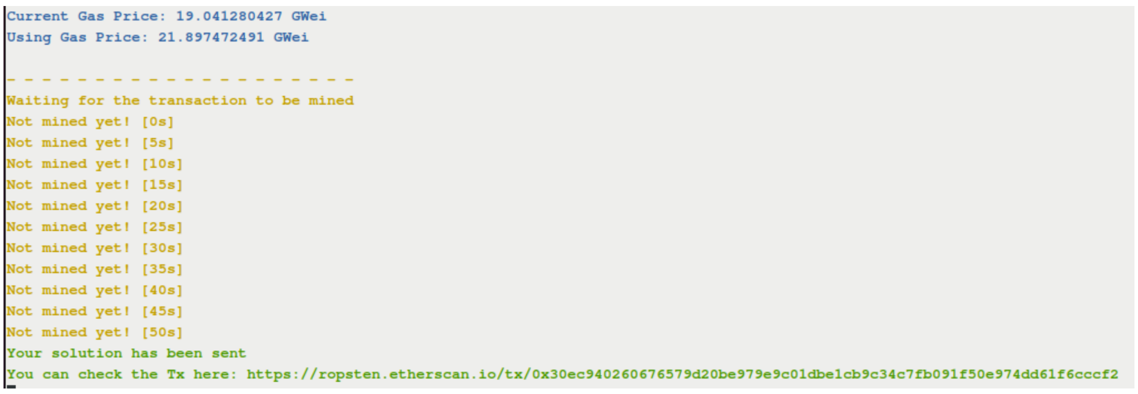


Notification
Your solution has been sent
You can check the Tx here: https://ropsten.etherscan.io/tx/-0x30ec940260676579d20be979e9c01dbe1cb9c34c7fb091f50e9-74dd61f6cccf2

Figure 5-12: Solution has been sent notification

https://ropsten.etherscan.io/tx/0x30ec940260676579d20be979e9c01dbe1cb9c34c7fb091f50e974dd61f6cccf2

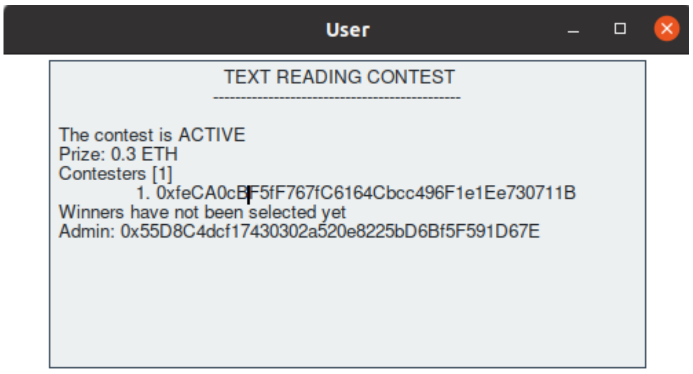
In the console we have useful information like the Gas Price that we are using, how long is it taking to mine the transaction and a link to the transaction itself.



```
Current Gas Price: 19.041280427 GWei
Using Gas Price: 21.897472491 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]
Not mined yet! [20s]
Not mined yet! [25s]
Not mined yet! [30s]
Not mined yet! [35s]
Not mined yet! [40s]
Not mined yet! [45s]
Not mined yet! [50s]
Your solution has been sent
You can check the Tx here: https://ropsten.etherscan.io/tx/0x30ec940260676579d20be979e9c01dbe1cb9c34c7fb091f50e974dd61f6cccf2
```

Figure 5-13: Printed information in console

Finally, we can check that our public address has been included into contesters:



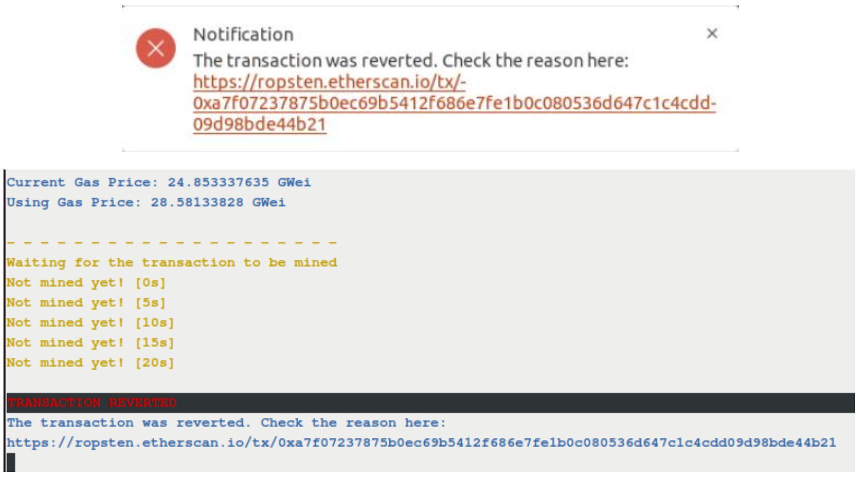
TEXT READING CONTEST

The contest is ACTIVE
Prize: 0.3 ETH
Contesters [1]
1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
Winners have not been selected yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E

Figure 5-14: Checking contesters address information

5.2.2.1.1 Bad transaction case example

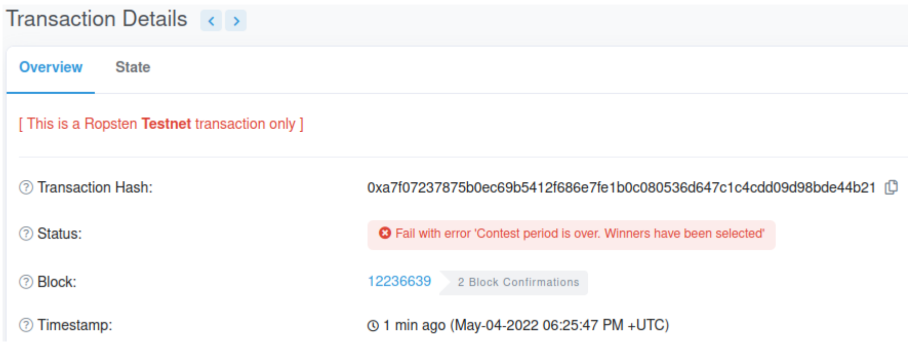
Here is an example of a bad transaction: Somebody has tried to participate after the winners have been selected. Once the transaction is mined, the application would prompt a notification:


Notification
The transaction was reverted. Check the reason here:
https://ropsten.etherscan.io/tx/-0xa7f07237875b0ec69b5412f686e7fe1b0c080536d647c1c4cdd-09d98bde44b21

```
Current Gas Price: 24.853337635 GWei
Using Gas Price: 28.58133828 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]
Not mined yet! [20s]

TRANSACTION REVERTED
The transaction was reverted. Check the reason here:
https://ropsten.etherscan.io/tx/0xa7f07237875b0ec69b5412f686e7fe1b0c080536d647c1c4cdd09d98bde44b21
```

Figure 5-15: Bad transaction – Application notification

To see more details, the user can inspect the provided URL:


Transaction Details

Overview State

[This is a Ropsten **Testnet** transaction only]

Transaction Hash:	0xa7f07237875b0ec69b5412f686e7fe1b0c080536d647c1c4cdd09d98bde44b21
Status:	Fail with error 'Contest period is over. Winners have been selected'
Block:	12236639 2 Block Confirmations
Timestamp:	1 min ago (May-04-2022 06:25:47 PM +UTC)

Figure 5-16: Bad transaction – failure reason

https://ropsten.etherscan.io/tx/0xa7f07237875b0ec69b5412f686e7fe1b0c080536d647c1c4cdd09d98bde44b21

5.2.2.2 As an Admin – Managing Contest

This is the menu of the Admin App. It allows to fully manage both contests (*Text from Image* and *Dog or Cat*):

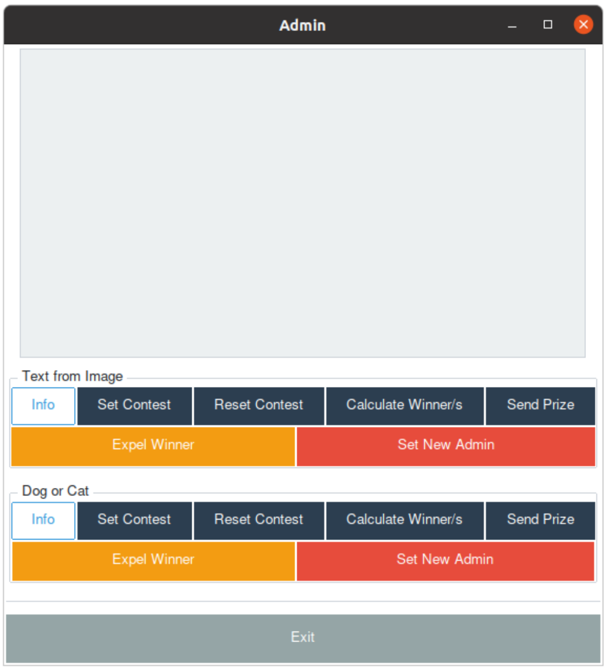
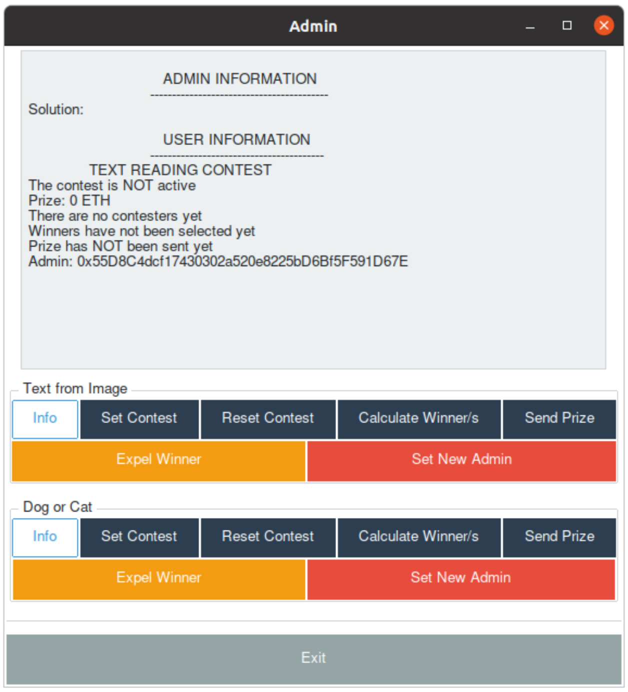


Figure 5-17: Admin's App – Main menu

See contest Info

To see the current status of the contest, the admin just needs to click on the *Info* button and relevant contest information (like if the contest is active, the prize, the solution, how many contesters are etc.) will be displayed.



```
              ADMIN INFORMATION
            ----------------------------------------
Solution:

              USER INFORMATION
            ----------------------------------------
        TEXT READING CONTEST
The contest is NOT active
Prize: 0 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

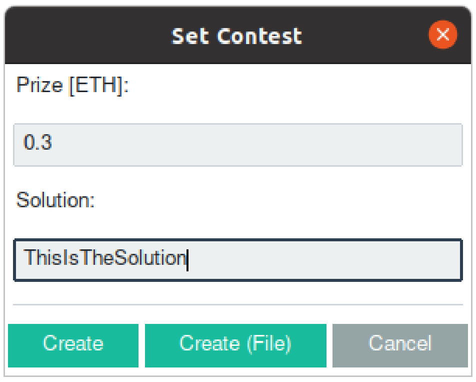
Figure 5-18: Contest Info

Creating a Contest

To set a contest the Admin has to click on the button *Set Contest*. Then he will have to set the prize amount (in Ethereum) and the solution.

To set the solution, there are two options:

- To directly write the solution in the box
- To import a txt file (the whole content of the file will be considered as the solution)



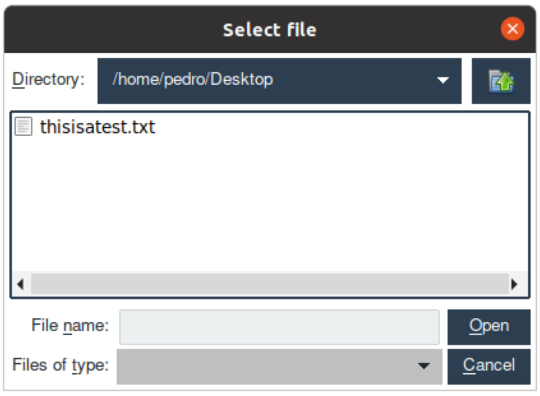
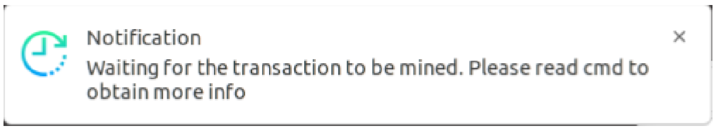


Figure 5-19: Setting contest solution: Direct input (left) and through a text file (right)

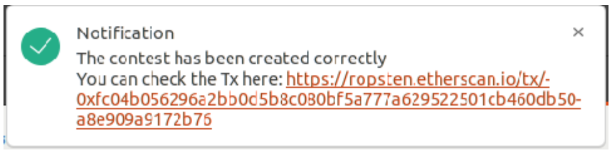
After clicking in *create*, this notification will be displayed:



Notification
Waiting for the transaction to be mined. Please read cmd to obtain more info

Figure 5-20: Waiting for transaction to be mined notification

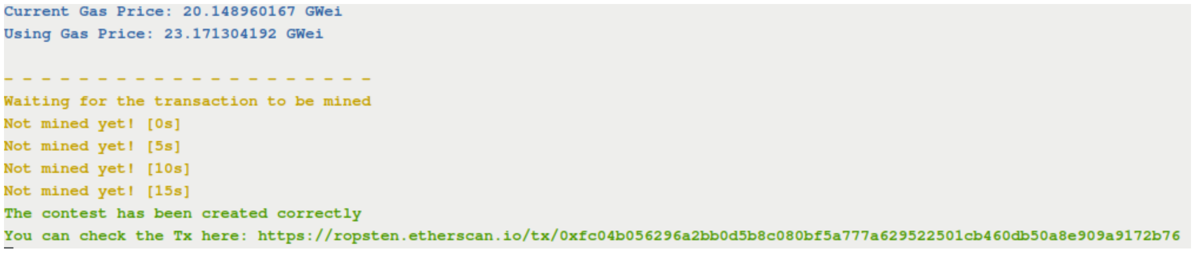
We will receive this notification when the transaction is mined:



Notification
The contest has been created correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/-0xfc04b056296a2bb0d5b8c080bf5a777a629522501cb460db50-a8e909a9172b76

Figure 5-21: Transaction mined correctly notification

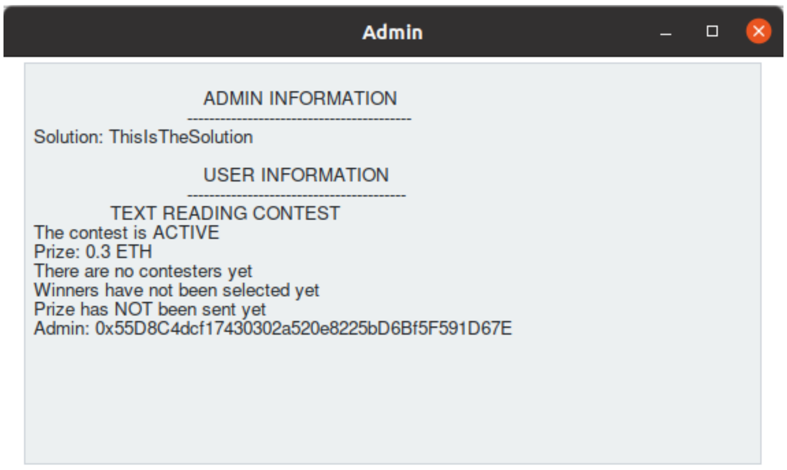
If we look at the cmd, we will find useful information like the gas price, how long it took the transaction to be mined and a link to the transaction itself.



```
Current Gas Price: 20.148960167 GWei
Using Gas Price: 23.171304192 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]
The contest has been created correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/0xfc04b056296a2bb0d5b8c080bf5a777a629522501cb460db50a8e909a9172b76
_
```

Figure 5-22: Useful information displayed in console

By clicking on *Info* button we can check that everything is ok:

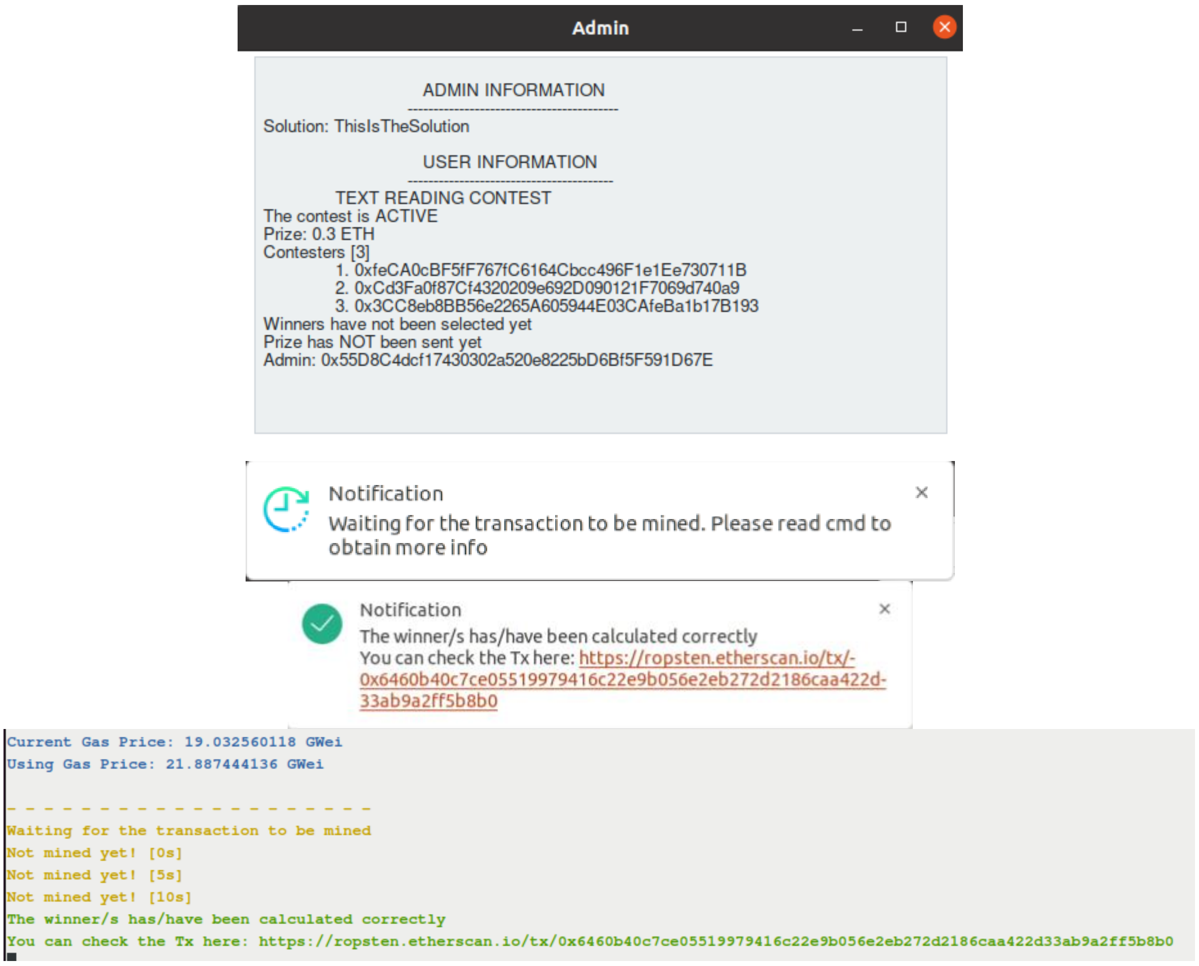


```
Admin

ADMIN INFORMATION
-----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
-----------------------------------------
TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.3 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-23: Info Button

Calculating Winner/s

To calculate the winner/s, the admin has to click on *Calculate Winner/s* button. The only restriction for it is that the contest needs to have, at least, one contester. The rest of it is similar as setting the contest:



```
Admin

ADMIN INFORMATION
-----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
-----------------------------------------
TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.3 ETH
Contesters [3]
        1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
        2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
        3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

```
Notification
Waiting for the transaction to be mined. Please read cmd to obtain more info
```

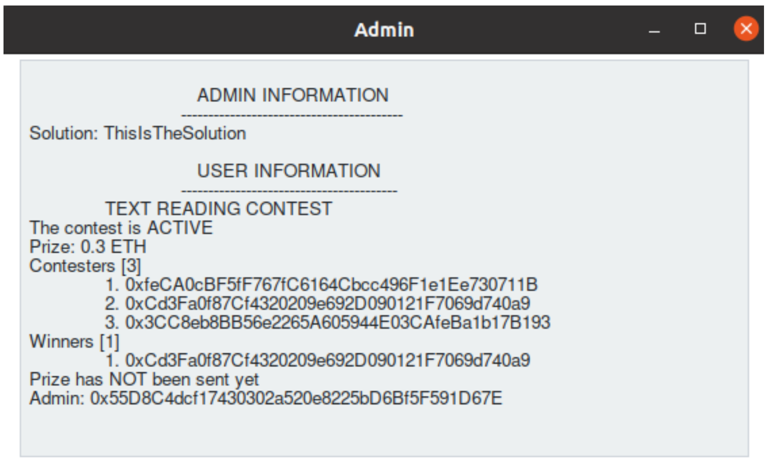
```
Notification
The winner/s has/have been calculated correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/-0x6460b40c7ce05519979416c22e9b056e2eb272d2186caa422d-33ab9a2ff5b8b0
```

```
Current Gas Price: 19.032560118 GWei
Using Gas Price: 21.887444136 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
The winner/s has/have been calculated correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/0x6460b40c7ce05519979416c22e9b056e2eb272d2186caa422d33ab9a2ff5b8b0
```

Figure 5-24: Calculating winner/s sequence

https://ropsten.etherscan.io/tx/0x6460b40c7ce05519979416c22e9b056e2eb272d2186caa422d33ab9a2ff5b8b0

We can check if everything is ok by clicking on the *Info* button.

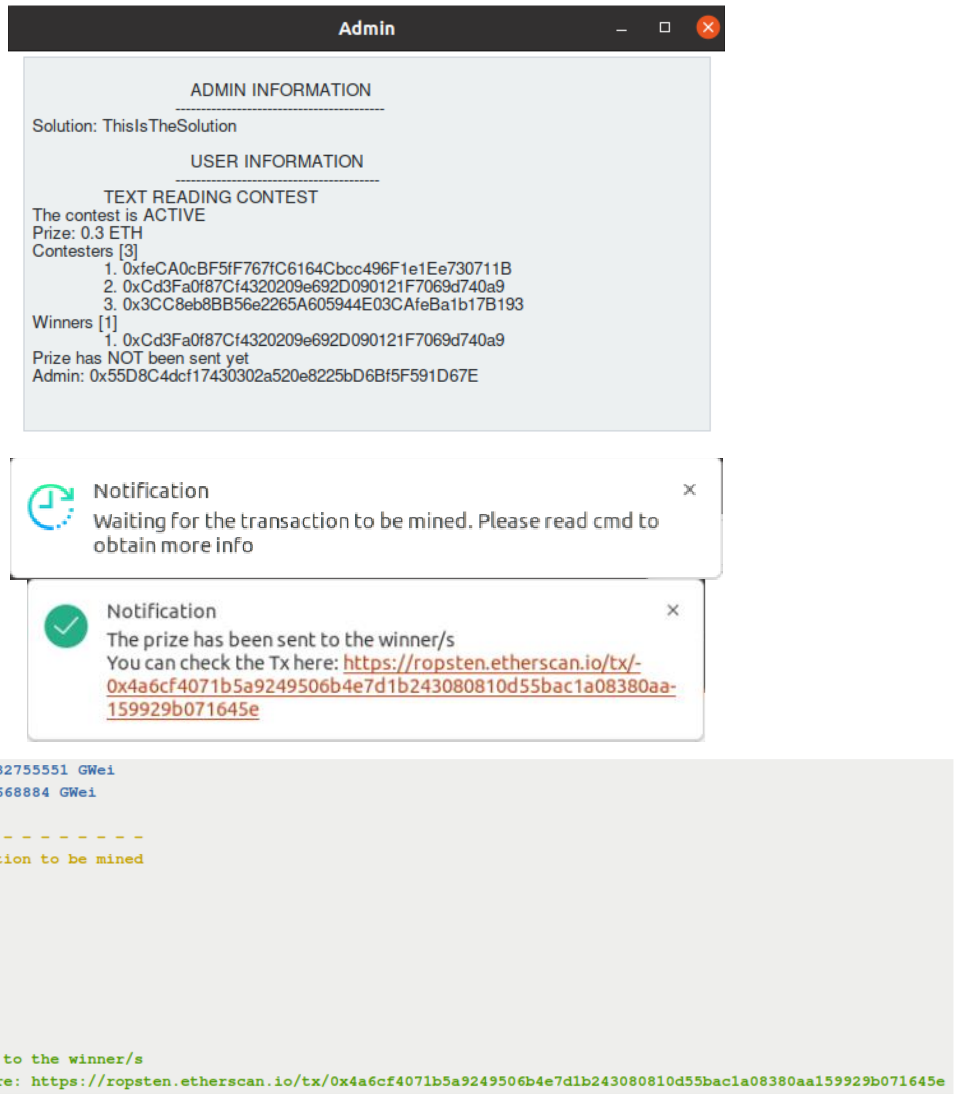


```
ADMIN INFORMATION
----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
----------------------------------------
            TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.3 ETH
Contesters [3]
            1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
            2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
            3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
Winners [1]
            1. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-25: Info Button check

Send Prize

To send the prize to the winner/s, the admin has to click on *Send Prize* button. The only restriction for it is that the contest needs to have, at least, one winner. The rest of it is similar as setting the contest or calculating winners:



```
ADMIN INFORMATION
----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
----------------------------------------
            TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.3 ETH
Contesters [3]
            1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
            2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
            3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
Winners [1]
            1. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Notification
Waiting for the transaction to be mined. Please read cmd to obtain more info

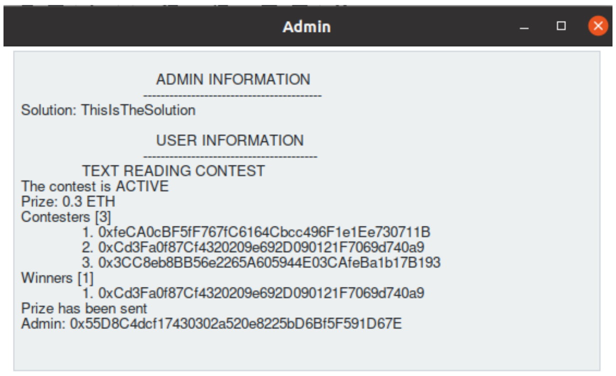
Notification
The prize has been sent to the winner/s
You can check the Tx here: https://ropsten.etherscan.io/tx/-0x4a6cf4071b5a9249506b4e7d1b243080810d55bac1a08380aa-159929b071645e

```
Current Gas Price: 20.332755551 GWei
Using Gas Price: 23.382668884 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]
Not mined yet! [20s]
Not mined yet! [25s]
Not mined yet! [30s]
Not mined yet! [35s]
The prize has been sent to the winner/s
You can check the Tx here: https://ropsten.etherscan.io/tx/0x4a6cf4071b5a9249506b4e7d1b243080810d55bac1a08380aa159929b071645e
```

Figure 5-26: Sending the prize sequence

https://ropsten.etherscan.io/tx/0x4a6cf4071b5a9249506b4e7d1b243080810d55bac1a08380aa159929b071645e

By clicking on *Info* button we can check that everything is ok

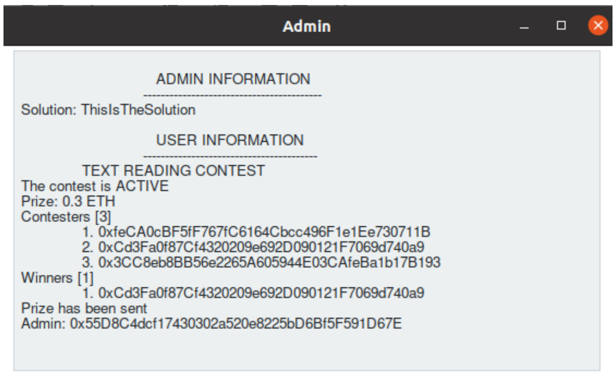


```
Admin

ADMIN INFORMATION
-----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
-----------------------------------------
TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.3 ETH
Contesters [3]
        1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
        2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
        3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
Winners [1]
        1. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
Prize has been sent
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

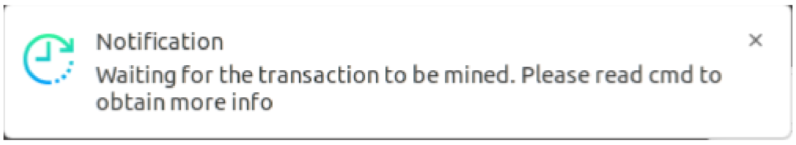
Figure 5-27: Info Button check

Resetting Contest

To reset the contest, the Admin has to click on *Reset Contest* button. The only restriction for it is that the prize must be sent to the winner/s. The rest of it is similar as setting the contest, calculating winners or sending the prize:

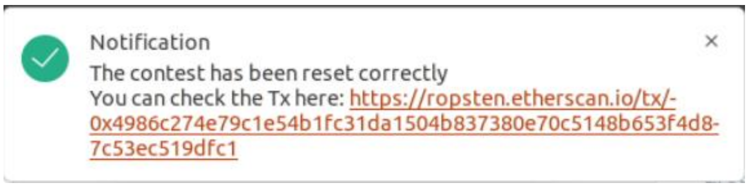


```
Admin

ADMIN INFORMATION
-----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
-----------------------------------------
TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.3 ETH
Contesters [3]
        1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
        2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
        3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
Winners [1]
        1. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
Prize has been sent
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```



Notification

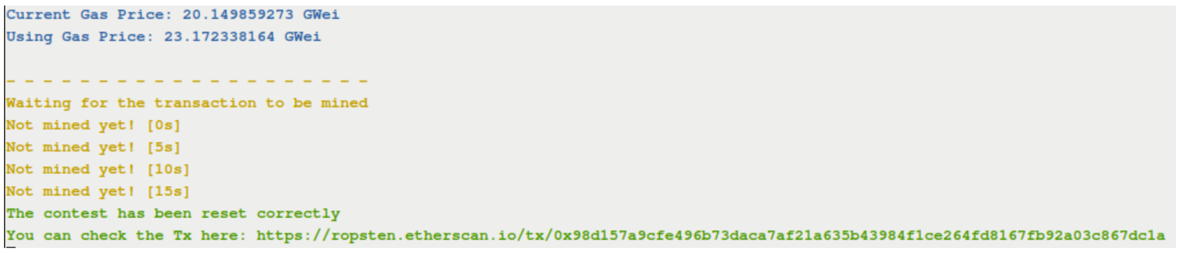
Waiting for the transaction to be mined. Please read cmd to obtain more info



Notification

The contest has been reset correctly

You can check the Tx here: https://ropsten.etherscan.io/tx/-0x4986c274e79c1e54b1fc31da1504b837380e70c5148b653f4d8-7c53ec519dfc1

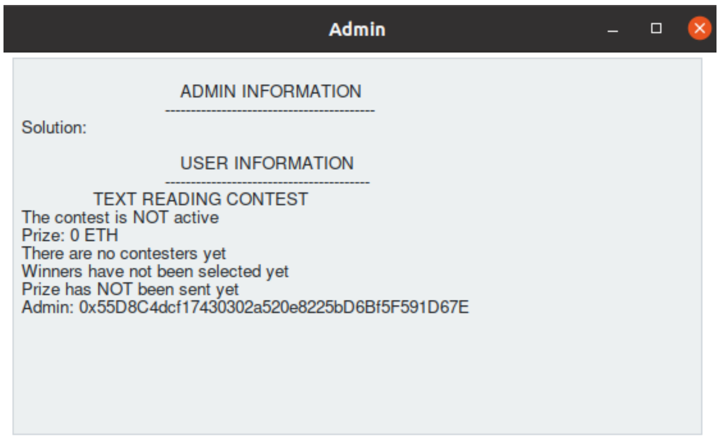


```
Current Gas Price: 20.149859273 GWei
Using Gas Price: 23.172338164 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]
The contest has been reset correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/0x98d157a9cfe496b73daca7af21a635b43984f1ce264fd8167fb92a03c867dc1a
```

Figure 5-28: Resetting Contest sequence

https://ropsten.etherscan.io/tx/0x98d157a9cfe496b73daca7af21a635b43984f1ce264fd8167fb92a03c867dc1a

By clicking on the *Info* button we can check that everything is ok:


Admin

ADMIN INFORMATION

Solution:

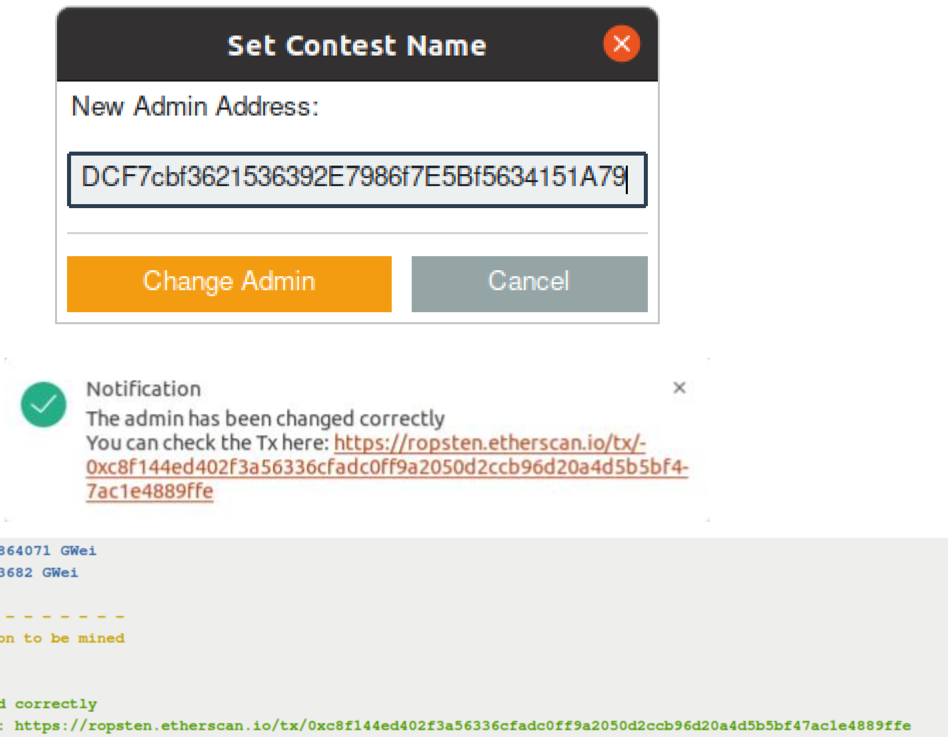
USER INFORMATION

TEXT READING CONTEST
The contest is NOT active
Prize: 0 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E

Figure 5-29: Info button check

Setting New Admin

To set a new administrator of a contest, the admin must click on *Set New Admin* button. It is similar as setting the contest, calculating winners or sending the prize:



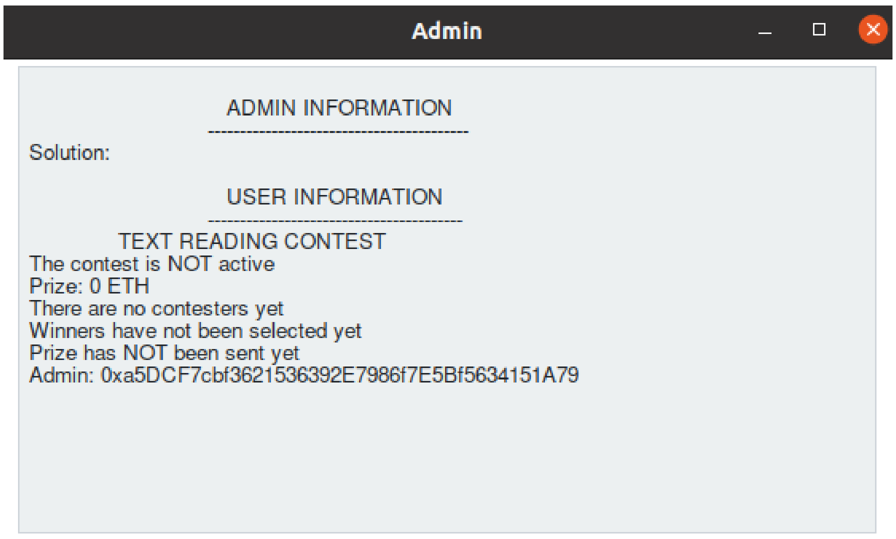
Notification
The admin has been changed correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/-0xc8f144ed402f3a56336cfadc0ff9a2050d2ccb96d20a4d5b5bf4-7ac1e4889ffe

```
Current Gas Price: 19.874864071 GWei
Using Gas Price: 22.856093682 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
The admin has been changed correctly
You can check the Tx here: https://ropsten.etherscan.io/tx/0xc8f144ed402f3a56336cfadc0ff9a2050d2ccb96d20a4d5b5bf47ac1e4889ffe
```

Figure 5-30: Setting new admin sequence

https://ropsten.etherscan.io/tx/0xc8f144ed402f3a56336cfadc0ff9a2050d2ccb96d20a4d5b5bf47ac1e4889ffe

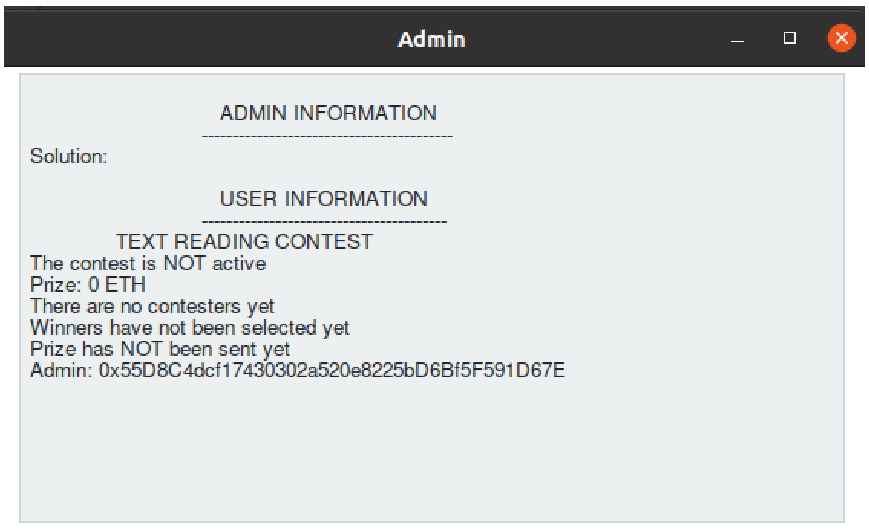
By clicking on *Info* button we can check that everything is ok:



```
                ADMIN INFORMATION
            ----------------------------------------
Solution:
                USER INFORMATION
            ----------------------------------------
        TEXT READING CONTEST
The contest is NOT active
Prize: 0 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0xa5DCF7cbf3621536392E7986f7E5Bf5634151A79
```

Figure 5-31: Info button check

Remember that initially we had:

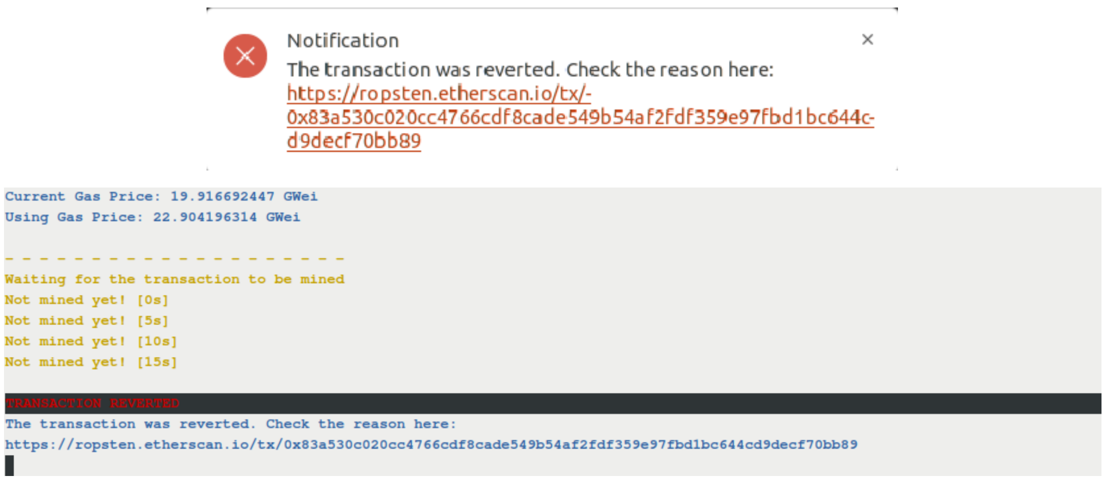


```
                ADMIN INFORMATION
            ----------------------------------------
Solution:
                USER INFORMATION
            ----------------------------------------
        TEXT READING CONTEST
The contest is NOT active
Prize: 0 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-32: Previous info button information

5.2.2.2.1 Bad case example

Here is an example of a bad transaction on the admin's side: he tried to send the prize without having elected the winners. Once the transaction is mined, the application would prompt a notification:



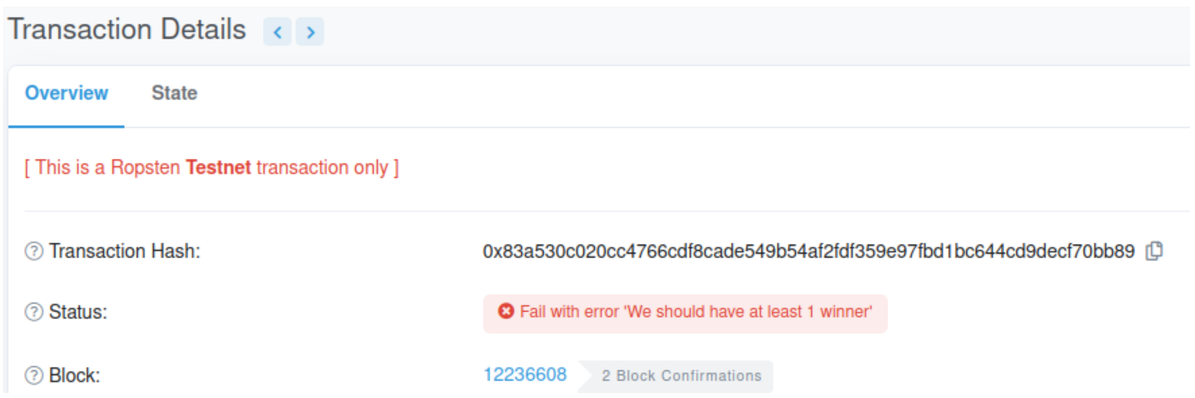
Notification

The transaction was reverted. Check the reason here: https://ropsten.etherscan.io/tx/-0x83a530c020cc4766cdf8cade549b54af2fdf359e97fbd1bc644c-d9decf70bb89

```
Current Gas Price: 19.916692447 GWei
Using Gas Price: 22.904196314 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]

TRANSACTION REVERTED
The transaction was reverted. Check the reason here:
https://ropsten.etherscan.io/tx/0x83a530c020cc4766cdf8cade549b54af2fdf359e97fbd1bc644cd9decf70bb89
```

Figure 5-33: Bad transaction – Application notification

To see more details, the user can inspect the provided URL:



Transaction Details

Overview | State

[This is a Ropsten **Testnet** transaction only]

Transaction Hash:	0x83a530c020cc4766cdf8cade549b54af2fdf359e97fbd1bc644cd9decf70bb89
Status:	Fail with error 'We should have at least 1 winner'
Block:	12236608 2 Block Confirmations

https://ropsten.etherscan.io/tx/0x83a530c020cc4766cdf8cade549b54af2fdf359e97fbd1bc644cd9decf70bb89

5.3 Project Design and Programming

5.3.1 Project Architecture/Design

We can divide the application structure in two different sides, local (Python) and remote (Smart Contract, Solidity).

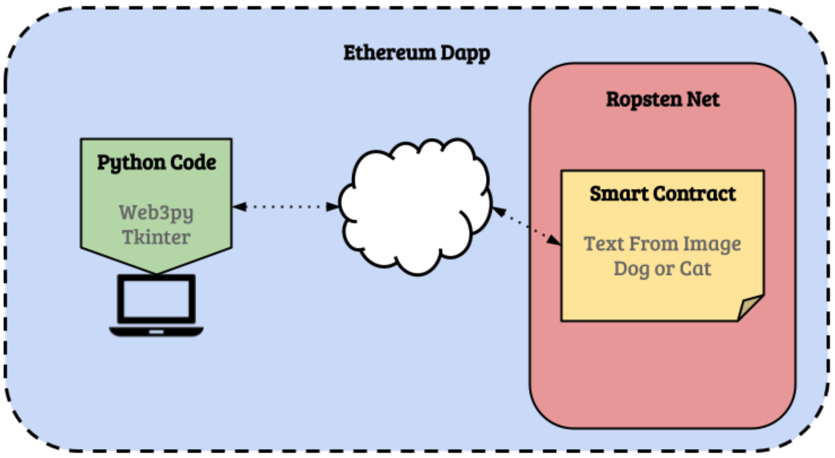


Figure 5-34: Application Structure

5.3.1.1 Local side

It is the Python code. Its purpose is to offer a graphical user interface to interact with the Smart Contract. It uses two main libraries:

- **Web3.py:** Library for interacting with Ethereum (sending transactions, interacting with smart contracts, reading block data, and a variety of other use cases). Here is the link for its documentation: https://web3py.readthedocs.io/en/stable/
- **Tkinter:** Library for the design and generation of the graphical user interface (different windows, buttons etc.). Here is the link for its documentation: https://docs.python.org/3/library/tkinter.html#module-tkinter

5.3.1.2 Remote side

It is the Ethereum Smart Contract. It contains both contests (*Text From Image* and *Dog or Cat*) and it is deployed in Ropsten, one of the test nets for Ethereum blockchain.

5.3.2 File Structure

5.3.2.1 Solidity Files

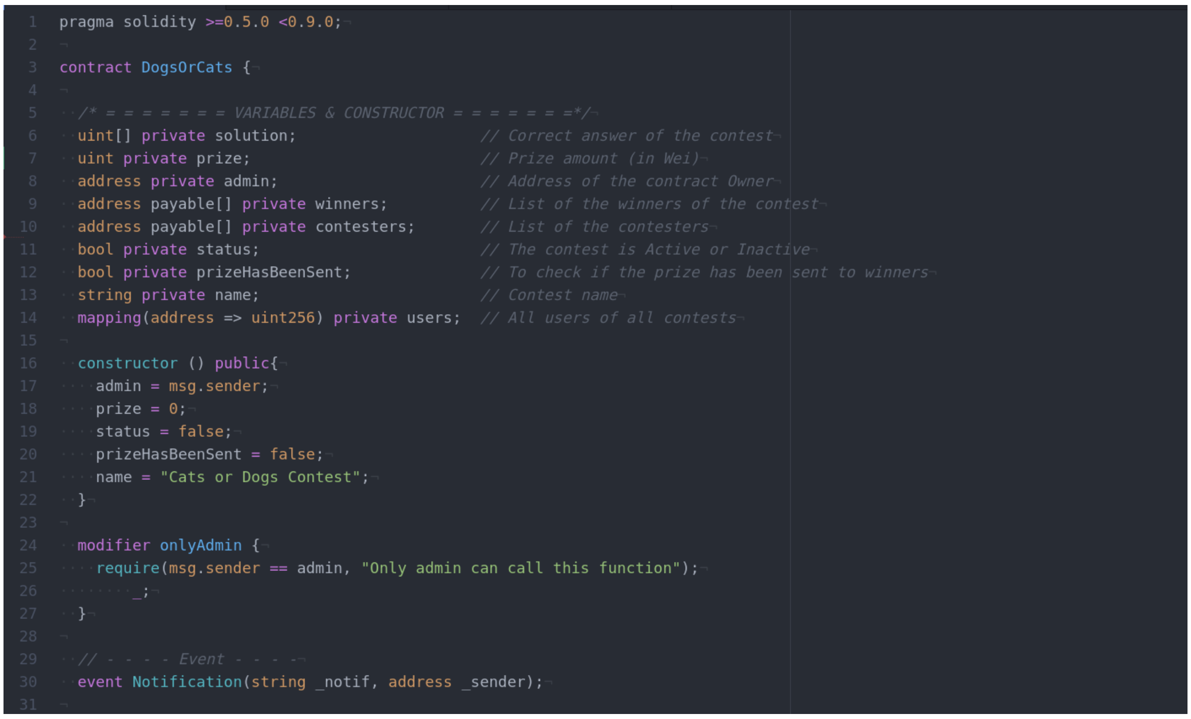
Solidity files are for defining Smart Contracts. This language is similar to Java, contracts are like classes. They both can have constructors, private and public methods, global and local variables, and can be instantiated. However, Solidity contracts also have public addresses in the blockchain (after being deployed) and can store and send value. There is also a special variable, *msg*. It contains properties which allow access to the blockchain's contracts, their functions, and their values. For example, *msg.sender* is always the address where a current function call came from, and *msg.value* is the amount of Ethereum sent in the transaction.

I have developed two Solidity files on this Ethereum Dapp: *DogsOrCats.sol* and *TextImage.sol*. Both of them follow a similar structure:

1. Global variables & constructor are defined
2. *OnlyAdmin* modifier and *Notification* event are defined
3. Public Getters & Setters are defined
4. Functions used by the Administrator of the contract
5. Functions used by contesters

5.3.2.1.1 DogsOrCats.sol

This file is for defining the *Dogs or Cats* Smart Contract. If we have a look at the first part of the file:



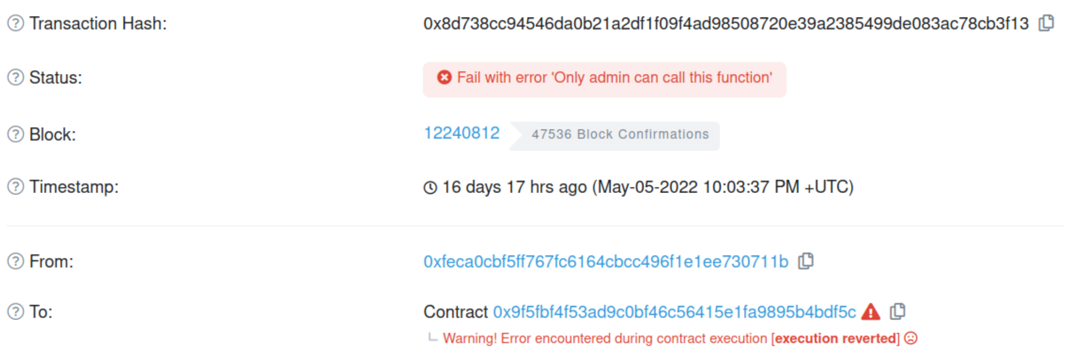
```
pragma solidity >=0.5.0 <0.9.0;

contract DogsOrCats {

  /* = = = = = = = VARIABLES & CONSTRUCTOR = = = = = =*/
  uint[] private solution;                      // Correct answer of the contest
  uint private prize;                           // Prize amount (in Wei)
  address private admin;                        // Address of the contract Owner
  address payable[] private winners;            // List of the winners of the contest
  address payable[] private contesters;         // List of the contesters
  bool private status;                          // The contest is Active or Inactive
  bool private prizeHasBeenSent;                // To check if the prize has been sent to winners
  string private name;                          // Contest name
  mapping(address => uint256) private users;    // All users of all contests

  constructor () public{
    admin = msg.sender;
    prize = 0;
    status = false;
    prizeHasBeenSent = false;
    name = "Cats or Dogs Contest";
  }

  modifier onlyAdmin {
    require(msg.sender == admin, "Only admin can call this function");
        _;
  }

  // - - - - Event - - - -
  event Notification(string _notif, address _sender);

```

Figure 5-35: DogsOrCats.sol: variables and constructor

The first line (*pragma solidity*) is to specify the valid version/s of the compiler. Solidity is a new language and it is continuously being improved. Whenever a new feature or improvement is introduced, it comes out with a new version.

Then we have the variables and the constructor. The constructor is executed just once, during the deployment of the contract. We have used it for initialising some variables like the admin's address (with *msg.sender* in the constructor we set that the admin's address is the address that deploys the contract) or the name of the contract. Regarding variables there are two aspects that are peculiar to this language. One is the *payable* modifier, which is required for addresses that are likely to receive money from the contract. The other aspect is the *mapping* kind of variable that associates two kinds of variables. For example, we use the *users* variable for storing all contesters and their scores.

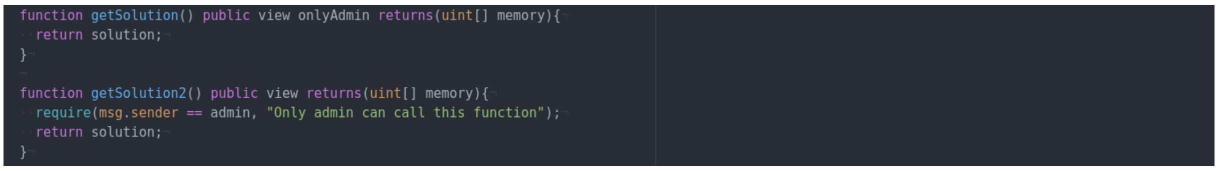
Before describing the purpose of the *onlyAdmin* modifier, it is needed to explain the use of the *require* method. Sometimes you need a function to be executed only under specific conditions; *require* will check if a condition is true and will allow code to flow only if the condition is true. If the condition is false, *require* will throw an error, the rest of the code will not be executed, and the transaction will revert. This will keep gas costs to a minimum (the rest of the function will not be executed) and the user will be able to receive a message with the cause of the error (in this case the message is 'Only admin can call this function'). Please note that this message will be stored in the transaction summary. Here is one example, https://ropsten.etherscan.io/tx/0x8d738cc94546da0b21a2df1f09f4ad98508720e39a2385499de083ac78cb3f13 :



```
Transaction Hash:   0x8d738cc94546da0b21a2df1f09f4ad98508720e39a2385499de083ac78cb3f13
Status:             Fail with error 'Only admin can call this function'
Block:              12240812  47536 Block Confirmations
Timestamp:          16 days 17 hrs ago (May-05-2022 10:03:37 PM +UTC)
From:               0xfeca0cbf5ff767fc6164cbcc496f1e1ee730711b
To:                 Contract 0x9f5fbf4f53ad9c0bf46c56415e1fa9895b4bdf5c
                    └ Warning! Error encountered during contract execution [execution reverted]
```

Figure 5-36: require condition not met

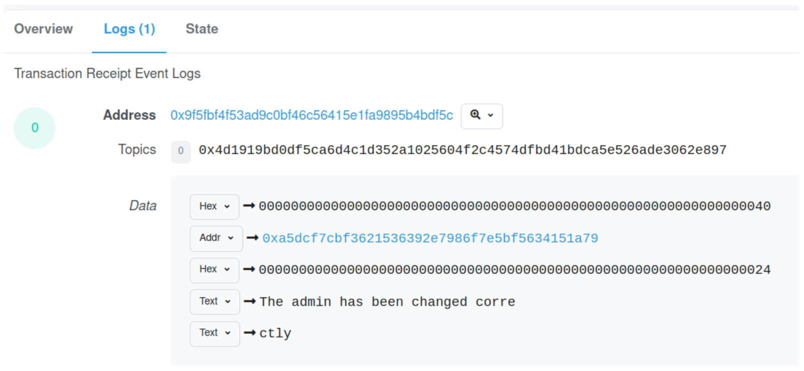
If we detect that we have a certain condition/s that we want to check in many functions, we can create a modifier to make the code shorter. These two functions are equivalent:



```
function getSolution() public view onlyAdmin returns(uint[] memory){
  return solution;
}

function getSolution2() public view returns(uint[] memory){
  require(msg.sender == admin, "Only admin can call this function");
  return solution;
}
```

Figure 5-37: Modifier example

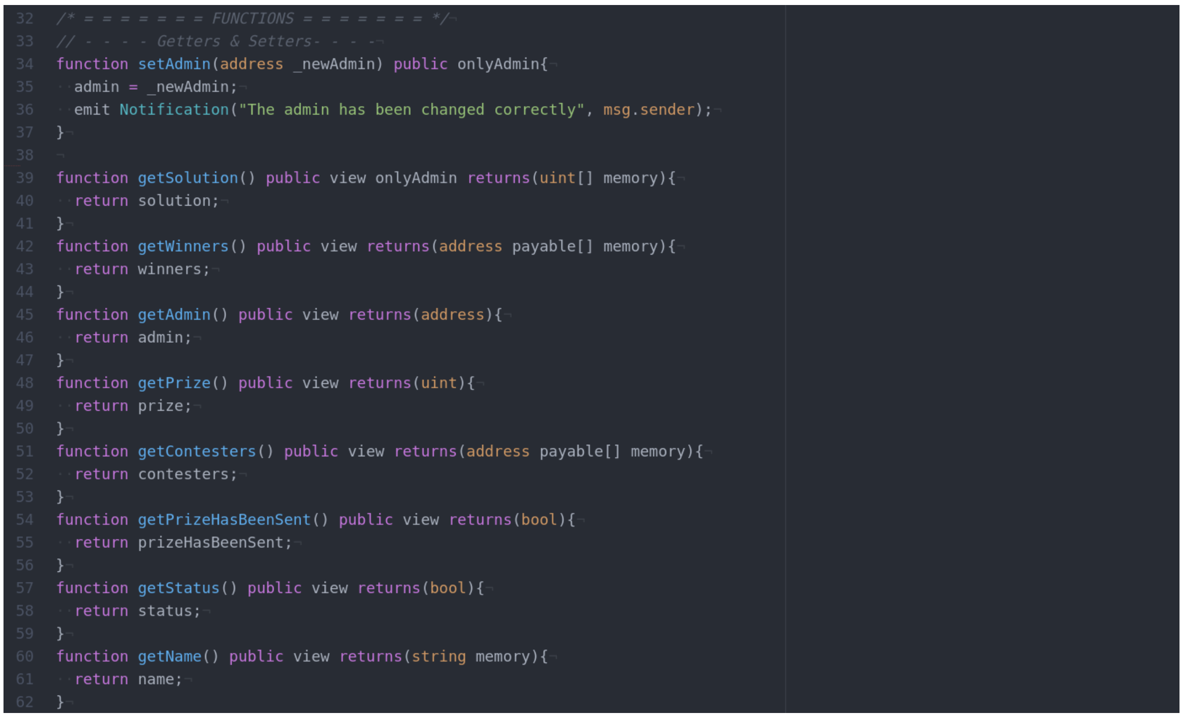
Events are for storing the arguments passed in the transaction or, in other words, for logging. This information is usually helpful for the user that initiated the transaction. The *Notification* event will store a string and the address of the user. Here is an example, https://ropsten.etherscan.io/tx/0x64f0b9d70b94888fb4dd3e0fcc243dbd9ee99aab9a7ec42c989d1964a3ffb917#eventlog :



```
Transaction Receipt Event Logs

0   Address  0x9f5fbf4f53ad9c0bf46c56415e1fa9895b4bdf5c
    Topics   0  0x4d1919bd0df5ca6d4c1d352a1025604f2c4574dfbd41bdca5e526ade3062e897
    Data     Hex  → 0000000000000000000000000000000000000000000000000000000000000040
             Addr → 0xa5dcf7cbf3621536392e7986f7e5bf5634151a79
             Hex  → 0000000000000000000000000000000000000000000000000000000000000024
             Text → The admin has been changed corre
             Text → ctly
```

Figure 5-38: Transaction logs example

Now, let's look at the Getters and Setters:

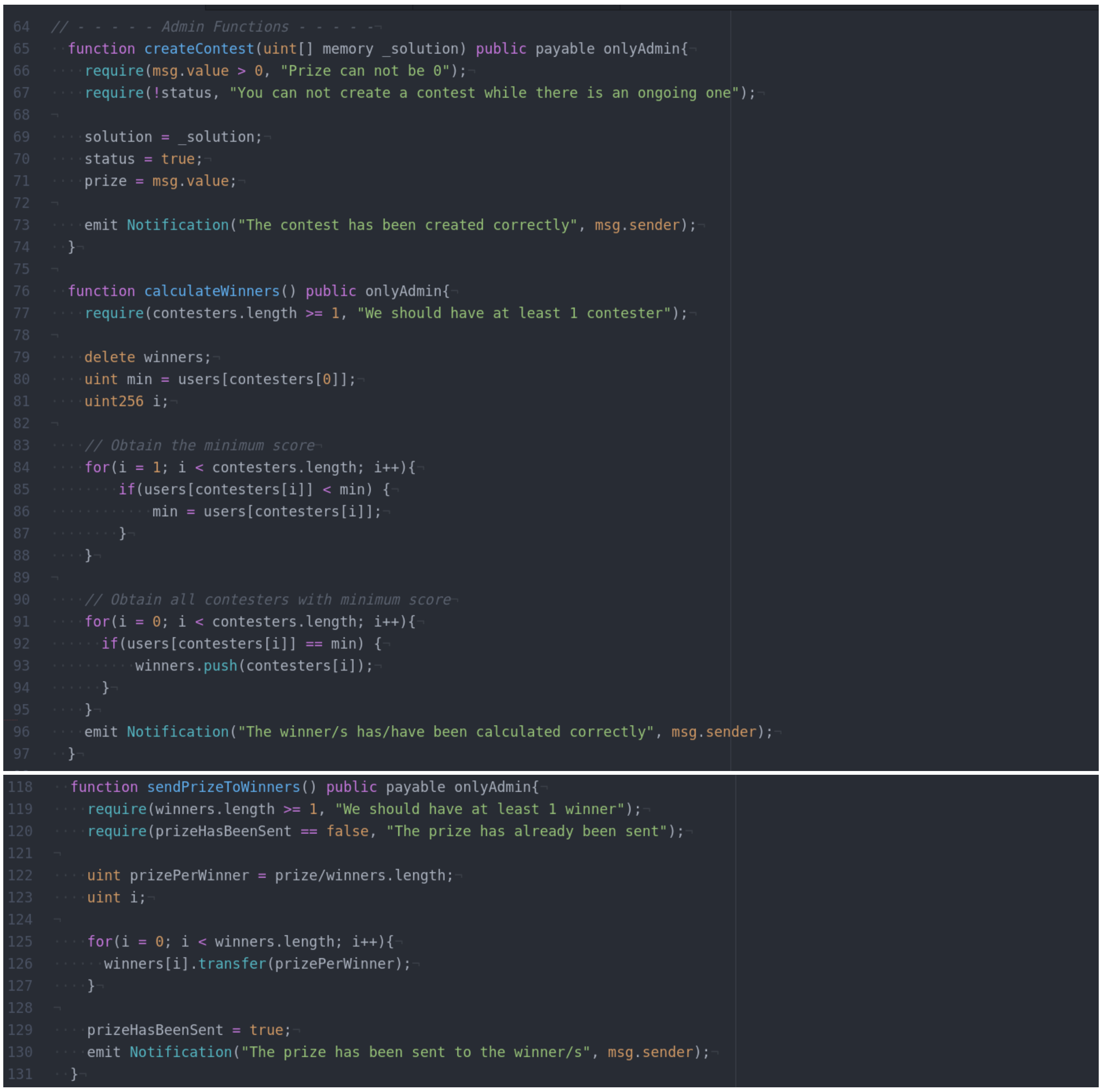


```
/* = = = = = = = FUNCTIONS = = = = = = = */
// - - - - Getters & Setters- - - -
function setAdmin(address _newAdmin) public onlyAdmin{
  admin = _newAdmin;
  emit Notification("The admin has been changed correctly", msg.sender);
}

function getSolution() public view onlyAdmin returns(uint[] memory){
  return solution;
}
function getWinners() public view returns(address payable[] memory){
  return winners;
}
function getAdmin() public view returns(address){
  return admin;
}
function getPrize() public view returns(uint){
  return prize;
}
function getContesters() public view returns(address payable[] memory){
  return contesters;
}
function getPrizeHasBeenSent() public view returns(bool){
  return prizeHasBeenSent;
}
function getStatus() public view returns(bool){
  return status;
}
function getName() public view returns(string memory){
  return name;
}
```

Figure 5-39: DogsOrCats.sol – Getters and Setters

You can see how to emit events (*setAdmin* method) and how some of the methods have the *onlyAdmin* modifier (*setAdmin* and *getSolution*).

Regarding the Admin methods, we are not going to cover them all, but we can see the most interesting ones.



```
// - - - - - Admin Functions - - - - -
  function createContest(uint[] memory _solution) public payable onlyAdmin{
    require(msg.value > 0, "Prize can not be 0");
    require(!status, "You can not create a contest while there is an ongoing one");

    solution = _solution;
    status = true;
    prize = msg.value;

    emit Notification("The contest has been created correctly", msg.sender);
  }

  function calculateWinners() public onlyAdmin{
    require(contesters.length >= 1, "We should have at least 1 contester");

    delete winners;
    uint min = users[contesters[0]];
    uint256 i;

    // Obtain the minimum score
    for(i = 1; i < contesters.length; i++){
        if(users[contesters[i]] < min) {
            min = users[contesters[i]];
        }
    }

    // Obtain all contesters with minimum score
    for(i = 0; i < contesters.length; i++){
      if(users[contesters[i]] == min) {
          winners.push(contesters[i]);
      }
    }
    emit Notification("The winner/s has/have been calculated correctly", msg.sender);
  }
```

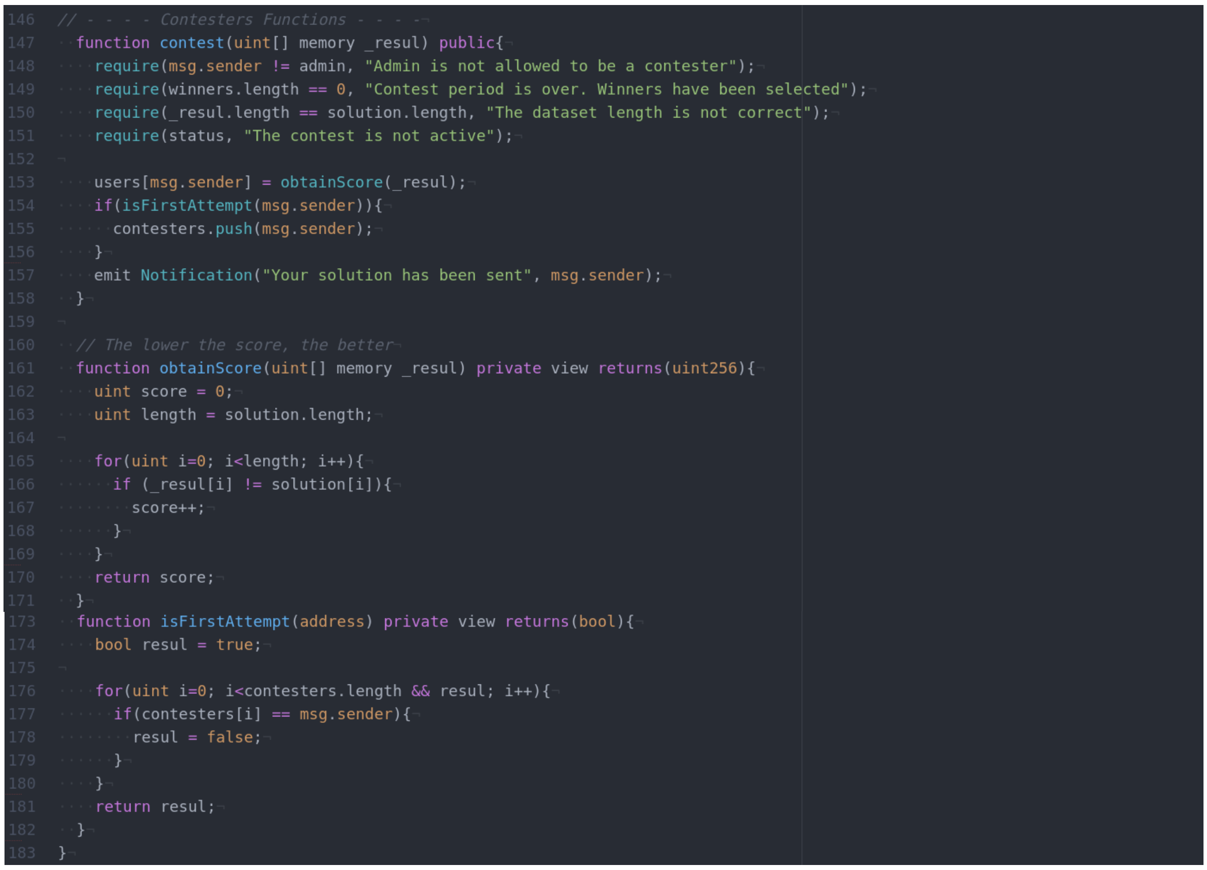
```
  function sendPrizeToWinners() public payable onlyAdmin{
    require(winners.length >= 1, "We should have at least 1 winner");
    require(prizeHasBeenSent == false, "The prize has already been sent");

    uint prizePerWinner = prize/winners.length;
    uint i;

    for(i = 0; i < winners.length; i++){
      winners[i].transfer(prizePerWinner);
    }

    prizeHasBeenSent = true;
    emit Notification("The prize has been sent to the winner/s", msg.sender);
  }
```

Figure 5-40: DogsOrCats.sol – createContest, calculateWinners and sendPrizeToWinners methods

The method *createContest* requires (besides being the admin and that there is not an ongoing contest) the transaction to contain some Ether, which is going to be the prize of the contest. Please note that the *prize* variable is just to store the amount of the prize, not the prize itself. The prize will pass from the Admin's Ethereum account to the Smart Contract account. To send the prize to the winner/s the Admin will use the method *sendPrizeToWinners*, which will equally divide the prize for all the winners and will transfer it to them.

With the method *calculateWinners* we will obtain the contester/s with the best (lowest) result. The result of each contester is calculated when they send their result.

Regarding participants functions they will participate with the *contest* function:



```
146 // - - - - Contesters Functions - - - -
147   function contest(uint[] memory _resul) public{
148     require(msg.sender != admin, "Admin is not allowed to be a contester");
149     require(winners.length == 0, "Contest period is over. Winners have been selected");
150     require(_resul.length == solution.length, "The dataset length is not correct");
151     require(status, "The contest is not active");
152 
153     users[msg.sender] = obtainScore(_resul);
154     if(isFirstAttempt(msg.sender)){
155       contesters.push(msg.sender);
156     }
157     emit Notification("Your solution has been sent", msg.sender);
158   }
159 
160   // The lower the score, the better
161   function obtainScore(uint[] memory _resul) private view returns(uint256){
162     uint score = 0;
163     uint length = solution.length;
164 
165     for(uint i=0; i<length; i++){
166       if (_resul[i] != solution[i]){
167         score++;
168       }
169     }
170     return score;
171   }
173   function isFirstAttempt(address) private view returns(bool){
174     bool resul = true;
175 
176     for(uint i=0; i<contesters.length && resul; i++){
177       if(contesters[i] == msg.sender){
178         resul = false;
179       }
180     }
181     return resul;
182   }
183 }
```

Figure 5-41: DogsOrCats.sol – contest, obtainScore and isFirstAttempt methods

This function is:

1. Mapping the user (its address) with the obtained score (line 153)
2. Storing the user address in a list of the participants (lines 154 and 155). As any user can participate as many times as desired (the valid score will be the last one), it is only required to store its address in the list just the first time.

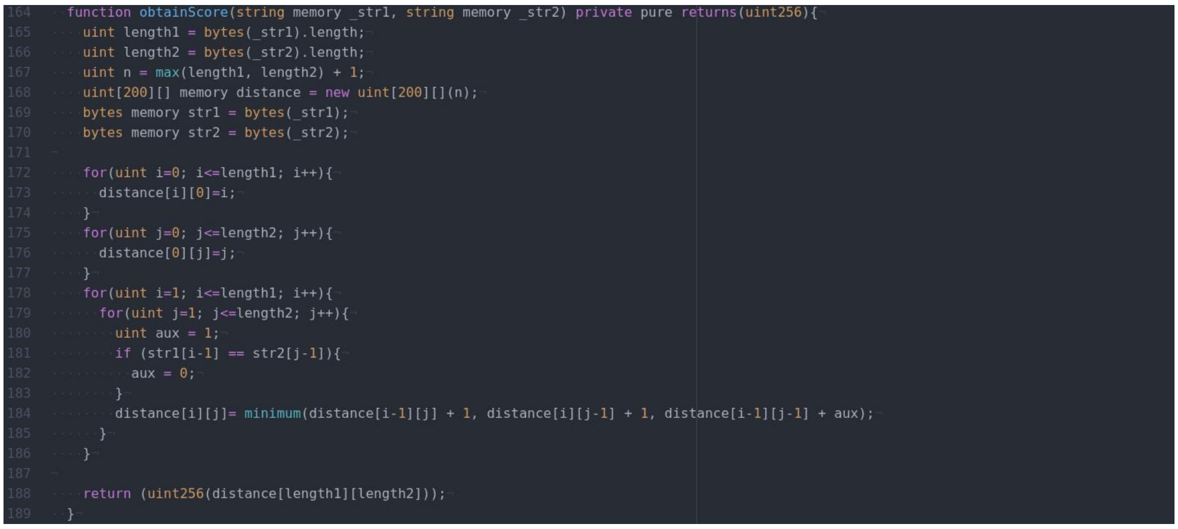
There is one more thing to explain, the *view* modificator (in methods *obtainScore* and *isFirstAttempt*). It is related to gas consumption. If we see the *contest* method, it is:

- Having computational cost
- Reading the state of the blockchain
- Modifying the state of the blockchain

Each one of those items consume gas and, by default, Solidity will assume that all functions are potentially capable of doing all of them. However, *obtainScore* and *isFirstAttempt* methods do not modify the state of the blockchain. Using the *view* modifier we let the compiler know that the function is not going to modify the state of the blockchain and it will make it more efficient in terms of gas consumption or, in case it is also a public method (like the *getters*), it will not require a transaction (no gas to be paid).

5.3.2.1.2 TextOrImage.sol

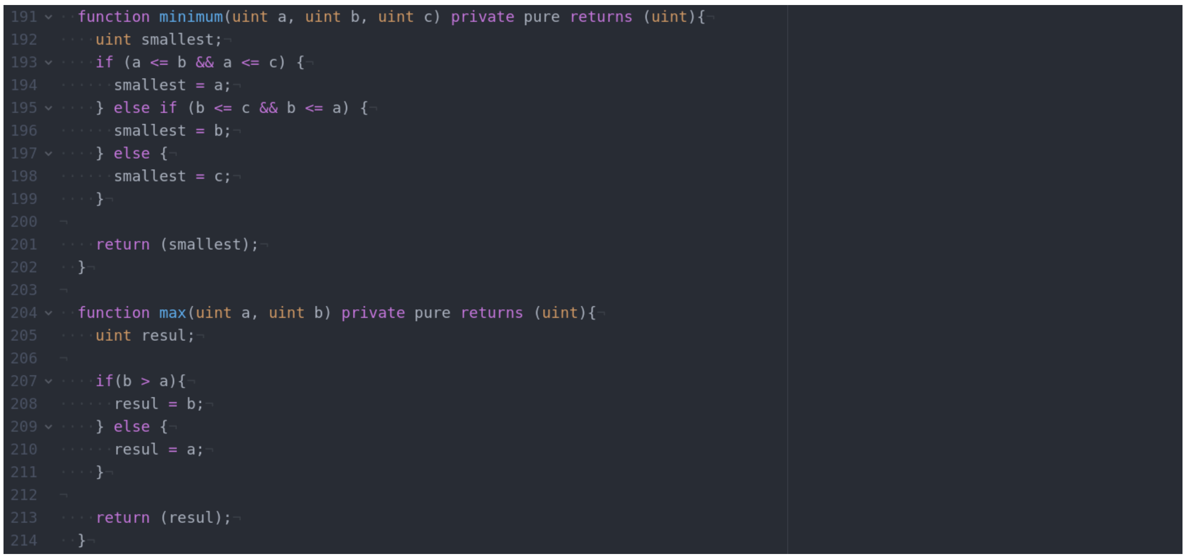
This file is for defining the *Text from Image* Smart Contract. As the structure is very similar to *DogsOrCats.sol*, we will only focus on its main difference which is the *obtainScore* method:



```solidity
function obtainScore(string memory _str1, string memory _str2) private pure returns(uint256){
  uint length1 = bytes(_str1).length;
  uint length2 = bytes(_str2).length;
  uint n = max(length1, length2) + 1;
  uint[200][] memory distance = new uint[200][](n);
  bytes memory str1 = bytes(_str1);
  bytes memory str2 = bytes(_str2);

  for(uint i=0; i<=length1; i++){
    distance[i][0]=i;
  }
  for(uint j=0; j<=length2; j++){
    distance[0][j]=j;
  }
  for(uint i=1; i<=length1; i++){
    for(uint j=1; j<=length2; j++){
      uint aux = 1;
      if (str1[i-1] == str2[j-1]){
        aux = 0;
      }
      distance[i][j]= minimum(distance[i-1][j] + 1, distance[i][j-1] + 1, distance[i-1][j-1] + aux);
    }
  }

  return (uint256(distance[length1][length2]));
}
```

Figure 5-42: TextOrImage.sol – obtainScore method

We want to compare the obtained string of the contester with the string of the real solution. For that we are using the Levenshtein distance algorithm, which counts the minimum number of modifications that must be made to a string to make it equal to the other string. A relevant aspect of Solidity is that there are no libraries available. I had to code the *max* and *minimum* methods:



```solidity
function minimum(uint a, uint b, uint c) private pure returns (uint){
  uint smallest;
  if (a <= b && a <= c) {
    smallest = a;
  } else if (b <= c && b <= a) {
    smallest = b;
  } else {
    smallest = c;
  }

  return (smallest);
}

function max(uint a, uint b) private pure returns (uint){
  uint resul;

  if(b > a){
    resul = b;
  } else {
    resul = a;
  }

  return (resul);
}
```

Figure 5-43: TextOrImage.sol – minimum and max methods

The last comment about this is the *pure* modificator (it is being used in the three methods; *obtainScore*, *minimum* and *max*). If the *view* modificator is for letting the compiler know that the function will only have computational cost and will need to read the status of the blockchain, the *pure* modificator is to indicate that the function will not need to read the status of the blockchain and, therefore, make it more efficient in terms of gas consumption.

5.3.2.2 Python Files

There are eight Python files in this project. Nevertheless, we are not going to focus on all of them but the ones related with the communication with the Smarts Contracts (*TextImageContract.py, Notifications.py* and *DogsOrCatsContract.py*). Here is a diagram about the relation of these files:

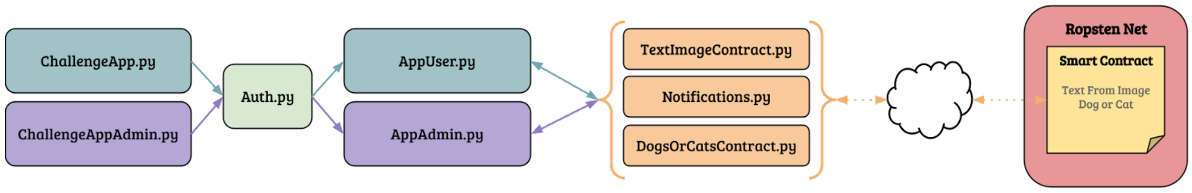


Figure 5-44: Pyhton Files diagram

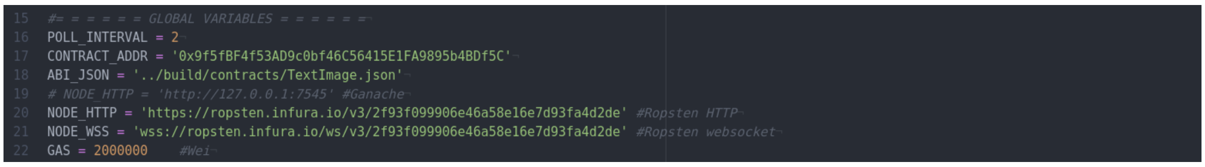
The purpose of *ChallengeApp.py* and *AppUser.py* is to create the graphical user interface for the contesters. In the same way, *ChallengeAppAdmin.py* and *AppAdmin.py* are for the generation of the graphical user interface for the Admin.

The purpose of *Auth.py* is to ask, check the format, store and use the private key of the user or admin.

5.3.2.2.1 TextImageContract.py

This file is for communicating with *Text from Image* Smart Contract. The whole content of the file is not going to be displayed, we will focus on the most interesting parts of it.

Global Variables

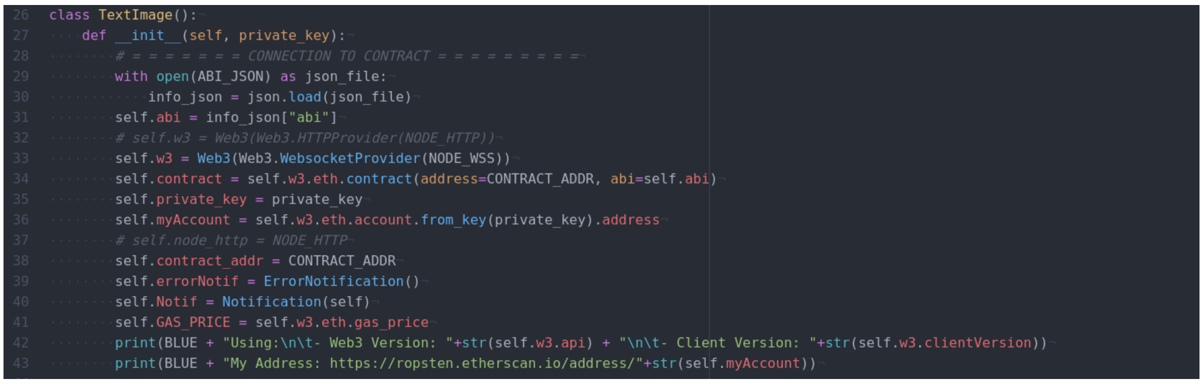


```
#= = = = = = GLOBAL VARIABLES = = = = = =
POLL_INTERVAL = 2
CONTRACT_ADDR = '0x9f5fBF4f53AD9c0bf46C56415E1FA9895b4BDf5C'
ABI_JSON = '../build/contracts/TextImage.json'
# NODE_HTTP = 'http://127.0.0.1:7545' #Ganache
NODE_HTTP = 'https://ropsten.infura.io/v3/2f93f099906e46a58e16e7d93fa4d2de' #Ropsten HTTP
NODE_WSS = 'wss://ropsten.infura.io/ws/v3/2f93f099906e46a58e16e7d93fa4d2de' #Ropsten websocket
GAS = 2000000     #Wei
```

Figure 5-45: TextImageContract.py – Global variables

The address of the contract is in the *CONTRACT_ADDR* variable. You will obtain this address when deploying it (it is explained in the section *Contract Deployment*; it is also explained how to obtain the *ABI_JSON* and why it is important). To establish the connection with the Smart Contract you can use HTTP or a websocket (*NODE_HTTP* and *NODE_WSS*). It is recommended to use a websocket if possible. Lastly, we set the maximum amount of gas to consume per transaction in the *GAS* variable.

Connection to the contract



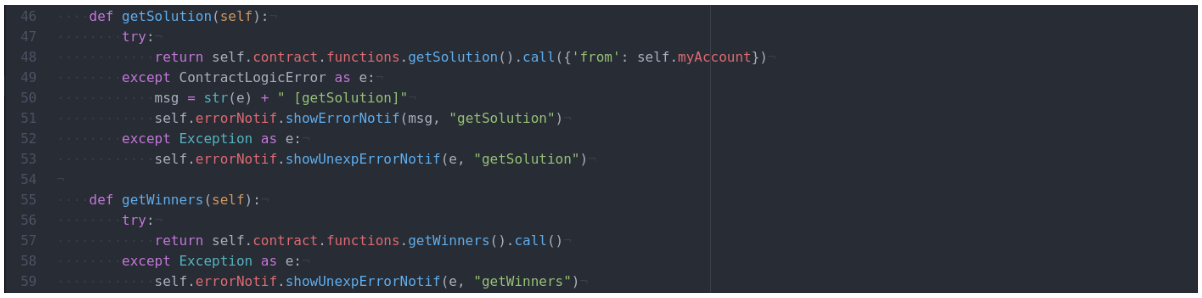
```
26  class TextImage():
27      def __init__(self, private_key):
28          # = = = = = = = = CONNECTION TO CONTRACT = = = = = = = = =
29          with open(ABI_JSON) as json_file:
30              info_json = json.load(json_file)
31          self.abi = info_json["abi"]
32          # self.w3 = Web3(Web3.HTTPProvider(NODE_HTTP))
33          self.w3 = Web3(Web3.WebsocketProvider(NODE_WSS))
34          self.contract = self.w3.eth.contract(address=CONTRACT_ADDR, abi=self.abi)
35          self.private_key = private_key
36          self.myAccount = self.w3.eth.account.from_key(private_key).address
37          # self.node_http = NODE_HTTP
38          self.contract_addr = CONTRACT_ADDR
39          self.errorNotif = ErrorNotification()
40          self.Notif = Notification(self)
41          self.GAS_PRICE = self.w3.eth.gas_price
42          print(BLUE + "Using:\n\t- Web3 Version: "+str(self.w3.api) + "\n\t- Client Version: "+str(self.w3.clientVersion))
43          print(BLUE + "My Address: https://ropsten.etherscan.io/address/"+str(self.myAccount))
```

Figure 5-46: TextImageContract.py – Connection to the contract

To establish the connection to the contract (line 34) we need the websocket, the contract address and the ABI of the contract. We obtain the *contract* variable that will be mainly utilised for using the methods of the contract.

Also *ErrorNotification* and *Notification* objects are instantiated to deal with the notifications of the *Text from Image* Smart Contract.

Functions that do not require a transaction

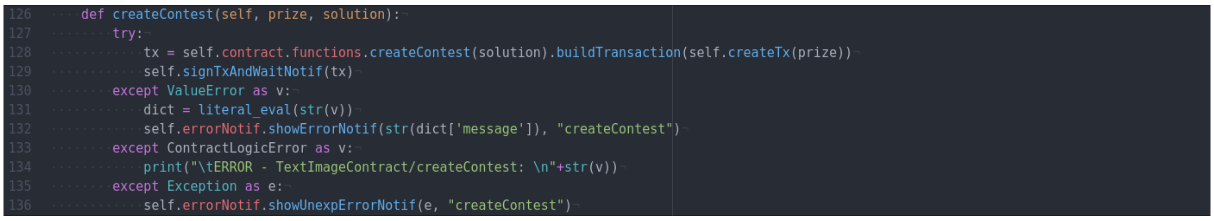


```
46     def getSolution(self):
47         try:
48             return self.contract.functions.getSolution().call({'from': self.myAccount})
49         except ContractLogicError as e:
50             msg = str(e) + " [getSolution]"
51             self.errorNotif.showErrorNotif(msg, "getSolution")
52         except Exception as e:
53             self.errorNotif.showUnexpErrorNotif(e, "getSolution")
54 
55     def getWinners(self):
56         try:
57             return self.contract.functions.getWinners().call()
58         except Exception as e:
59             self.errorNotif.showUnexpErrorNotif(e, "getWinners")
```

Figure 5-47: TextImageContract.py – Functions that do not need a transaction

These methods will not modify the state of the blockchain, therefore no gas will be consumed and no transaction will be needed. To execute them, you will have to use the *contract* variable and use the *call* method adding parameters if required (lines 48 and 57).

Functions that require a transaction



```
    def createContest(self, prize, solution):
        try:
            tx = self.contract.functions.createContest(solution).buildTransaction(self.createTx(prize))
            self.signTxAndWaitNotif(tx)
        except ValueError as v:
            dict = literal_eval(str(v))
            self.errorNotif.showErrorNotif(str(dict['message']), "createContest")
        except ContractLogicError as v:
            print("\tERROR - TextImageContract/createContest: \n"+str(v))
        except Exception as e:
            self.errorNotif.showUnexpErrorNotif(e, "createContest")
```

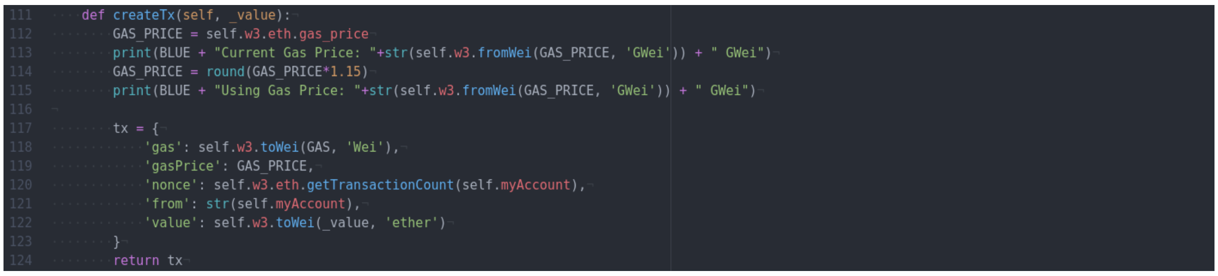
Figure 5-48: TextImageContract.py – Functions that need a transaction

These methods will modify the status of the blockchain; therefore gas will be consumed and a transaction will be needed. To execute them, you will have to use the *contract* variable and use the *buildTransaction* method, that will need a transaction as a parameter.

Generating a transaction

Defining a transaction is simple in Web3py, it is just a json object with the next fields:

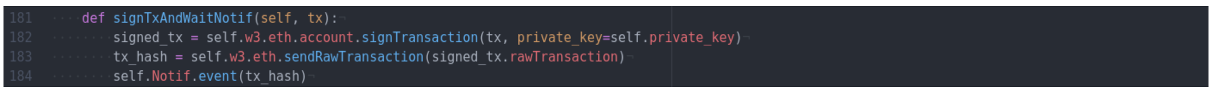
- gas: Maximum amount of gas willing to consume for this transaction.
- gasPrice: Amount of Ether (in Wei) to pay per unit of gas. As the gas price is not stable and the priority (for this project) is to get the transaction mined as soon as possible, I set it to be 15% higher than the gas price of the last transaction in the blockchain.
- nonce: The nonce is the number of transactions sent from a given address (each time you send a transaction, the nonce increases by one). This field prevents double-spends, as the nonce is the order in which transactions are sorted.
- from: The origin account of the transaction.
- value: Amount of Ether (in Wei) to transfer in the transaction (it can be zero).



```
    def createTx(self, _value):
        GAS_PRICE = self.w3.eth.gas_price
        print(BLUE + "Current Gas Price: "+str(self.w3.fromWei(GAS_PRICE, 'GWei')) + " GWei")
        GAS_PRICE = round(GAS_PRICE*1.15)
        print(BLUE + "Using Gas Price: "+str(self.w3.fromWei(GAS_PRICE, 'GWei')) + " GWei")

        tx = {
            'gas': self.w3.toWei(GAS, 'Wei'),
            'gasPrice': GAS_PRICE,
            'nonce': self.w3.eth.getTransactionCount(self.myAccount),
            'from': str(self.myAccount),
            'value': self.w3.toWei(_value, 'ether')
        }
        return tx
```

Figure 5-49: TextImageContract.py – Creating a transaction

Signing a Transaction

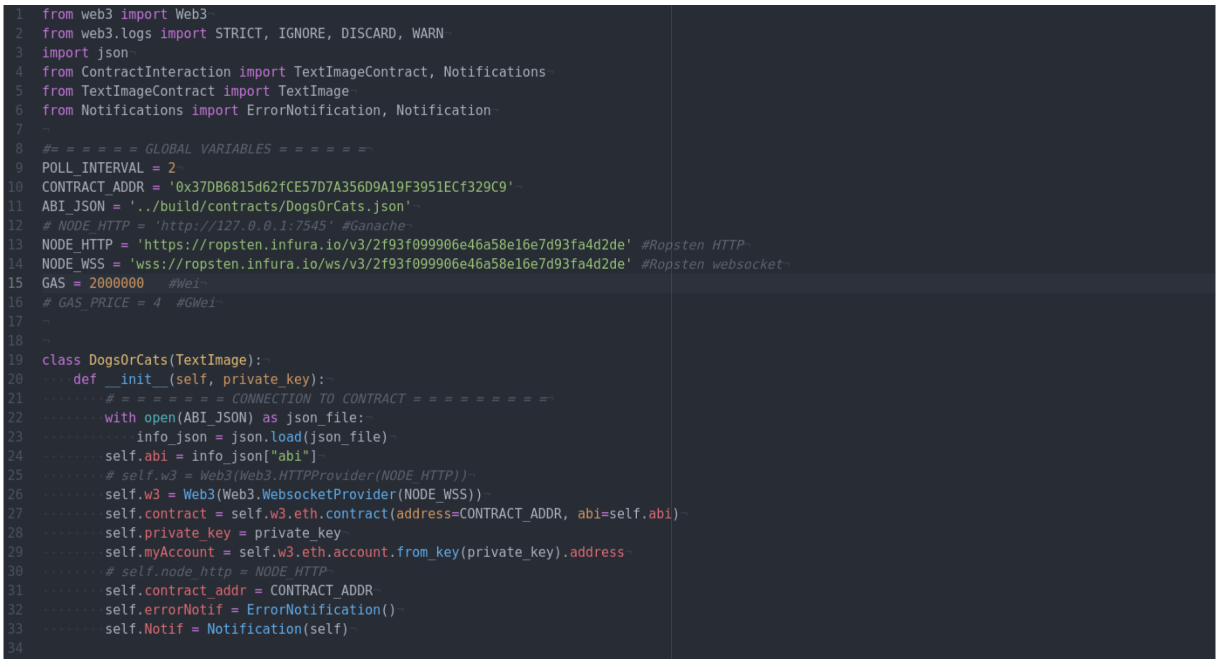


```
    def signTxAndWaitNotif(self, tx):
        signed_tx = self.w3.eth.account.signTransaction(tx, private_key=self.private_key)
        tx_hash = self.w3.eth.sendRawTransaction(signed_tx.rawTransaction)
        self.Notif.event(tx_hash)
```

Figure 5-50: TextImageContract.py – Signing a transaction

5.3.2.2.2 DogsOrCatsContract.py

As DogsOrCats class inherits all methods from TextImage class, the file content is much shorter. This is the whole file:



```python
from web3 import Web3
from web3.logs import STRICT, IGNORE, DISCARD, WARN
import json
from ContractInteraction import TextImageContract, Notifications
from TextImageContract import TextImage
from Notifications import ErrorNotification, Notification

#= = = = = = GLOBAL VARIABLES = = = = = =
POLL_INTERVAL = 2
CONTRACT_ADDR = '0x37DB6815d62fCE57D7A356D9A19F3951ECf329C9'
ABI_JSON = '../build/contracts/DogsOrCats.json'
# NODE_HTTP = 'http://127.0.0.1:7545' #Ganache
NODE_HTTP = 'https://ropsten.infura.io/v3/2f93f099906e46a58e16e7d93fa4d2de' #Ropsten HTTP
NODE_WSS = 'wss://ropsten.infura.io/ws/v3/2f93f099906e46a58e16e7d93fa4d2de' #Ropsten websocket
GAS = 2000000   #Wei
# GAS_PRICE = 4  #GWei


class DogsOrCats(TextImage):
    def __init__(self, private_key):
        # = = = = = = = = CONNECTION TO CONTRACT = = = = = = = = =
        with open(ABI_JSON) as json_file:
            info_json = json.load(json_file)
        self.abi = info_json["abi"]
        # self.w3 = Web3(Web3.HTTPProvider(NODE_HTTP))
        self.w3 = Web3(Web3.WebsocketProvider(NODE_WSS))
        self.contract = self.w3.eth.contract(address=CONTRACT_ADDR, abi=self.abi)
        self.private_key = private_key
        self.myAccount = self.w3.eth.account.from_key(private_key).address
        # self.node_http = NODE_HTTP
        self.contract_addr = CONTRACT_ADDR
        self.errorNotif = ErrorNotification()
        self.Notif = Notification(self)

```

Figure 5-51: DogsOrCatsContract.py file

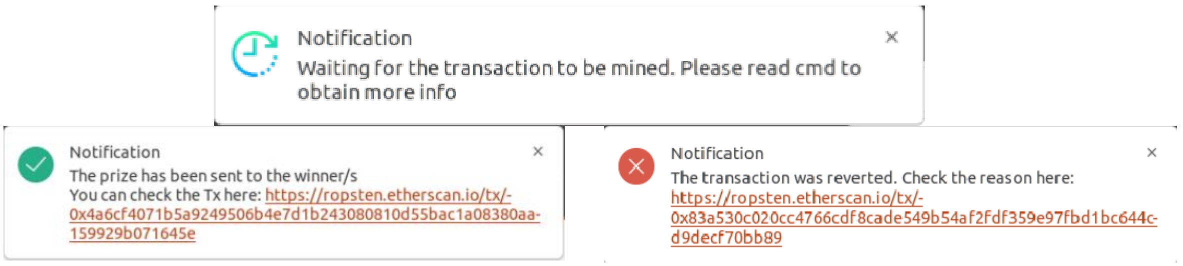
The differences are the contract address and the ABI json.

5.3.2.2.3 Notifications.py

The purpose of this file is to handle all notifications coming from *Text from Image* and *Dogs Or Cats* Smart Contracts. In general, there are two kinds of notifications for Ethereum transactions:

- Events and reverts: The transaction has been mined. If all *require* conditions have been satisfied and the contract is notifying something (emitting an event) we will be able to catch the event. In this case, we only have one kind of event in each contract (*Notification*). But it can also happen that one of the *require* conditions has not been satisfied. Then, the Smart Contract would revert the transaction (and we would notice).
- Errors: The transaction was not mined. This can happen because of an unexpected error (for example network problem).

These notifications will pop up a window with a description of it:



Notification
Waiting for the transaction to be mined. Please read cmd to obtain more info

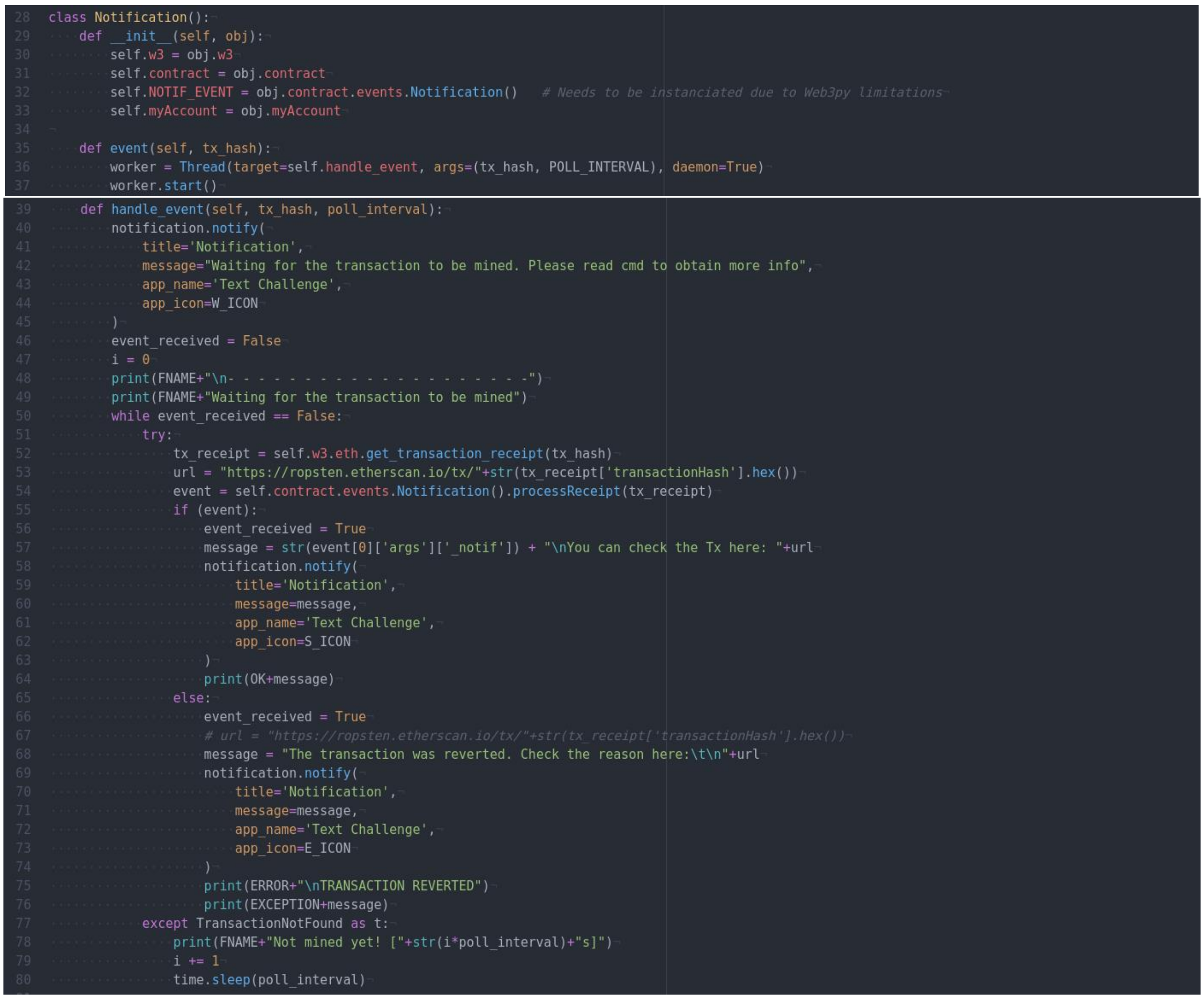
Notification
The prize has been sent to the winner/s
You can check the Tx here: https://ropsten.etherscan.io/tx/-0x4a6cf4071b5a9249506b4e7d1b243080810d55bac1a08380aa-159929b071645e

Notification
The transaction was reverted. Check the reason here:
https://ropsten.etherscan.io/tx/-0x83a530c020cc4766cdf8cade549b54af2fdf359e97fbd1bc644c-d9decf70bb89

Figure 5-52: Notifications Examples

Class Notification

This class is for handling events and reverts:



```
28 class Notification():
29     def __init__(self, obj):
30         self.w3 = obj.w3
31         self.contract = obj.contract
32         self.NOTIF_EVENT = obj.contract.events.Notification()   # Needs to be instanciated due to Web3py limitations
33         self.myAccount = obj.myAccount
34 
35     def event(self, tx_hash):
36         worker = Thread(target=self.handle_event, args=(tx_hash, POLL_INTERVAL), daemon=True)
37         worker.start()
39     def handle_event(self, tx_hash, poll_interval):
40         notification.notify(
41             title='Notification',
42             message="Waiting for the transaction to be mined. Please read cmd to obtain more info",
43             app_name='Text Challenge',
44             app_icon=W_ICON
45         )
46         event_received = False
47         i = 0
48         print(FNAME+"\n- - - - - - - - - - - - - - - - - - - - -")
49         print(FNAME+"Waiting for the transaction to be mined")
50         while event_received == False:
51             try:
52                 tx_receipt = self.w3.eth.get_transaction_receipt(tx_hash)
53                 url = "https://ropsten.etherscan.io/tx/"+str(tx_receipt['transactionHash'].hex())
54                 event = self.contract.events.Notification().processReceipt(tx_receipt)
55                 if (event):
56                     event_received = True
57                     message = str(event[0]['args']['_notif']) + "\nYou can check the Tx here: "+url
58                     notification.notify(
59                         title='Notification',
60                         message=message,
61                         app_name='Text Challenge',
62                         app_icon=S_ICON
63                     )
64                     print(OK+message)
65                 else:
66                     event_received = True
67                     # url = "https://ropsten.etherscan.io/tx/"+str(tx_receipt['transactionHash'].hex())
68                     message = "The transaction was reverted. Check the reason here:\t\n"+url
69                     notification.notify(
70                         title='Notification',
71                         message=message,
72                         app_name='Text Challenge',
73                         app_icon=E_ICON
74                     )
75                     print(ERROR+"\nTRANSACTION REVERTED")
76                     print(EXCEPTION+message)
77             except TransactionNotFound as t:
78                 print(FNAME+"Not mined yet! ["+str(i*poll_interval)+"s]")
79                 i += 1
80                 time.sleep(poll_interval)
```

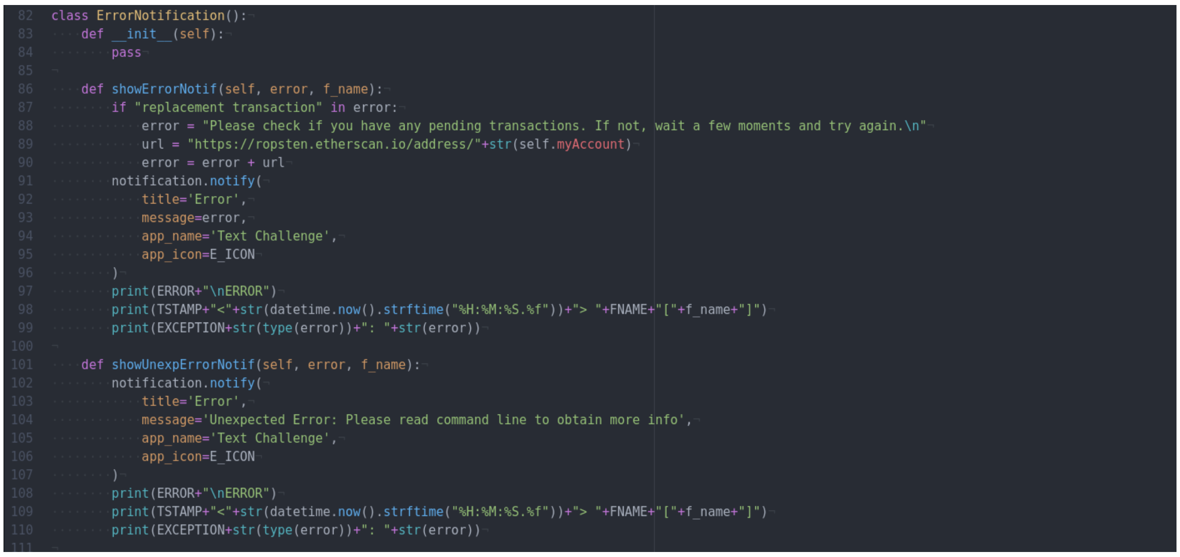
Figure 5-53: Notifications.py – Notification class

Whenever a transaction is started, this object will be constantly checking its status:

1. **Obtaining the receipt (line 52):** When a transaction is initiated we have its transaction hash and, when it is mined we can obtain its receipt (with the hash). If we are not able to obtain the receipt, that means that the transaction has not been mined yet. A *TransactionNotFound* exception will trigger (line 77) and we will need to wait some time.
2. **Obtaining the event (line 54):** Once we obtain the receipt, we will be able to search if there is an event on it:
    1. **If there is an event (line 55):** That means the called method has been successfully called and the event is displayed in the pop-up window (line 57).
    2. **If there is not an event (line 65):** In this case, it means that there transaction has been reverted because I have designed the contract to emit a *Notification* event in all methods that require a transaction. The url of the transaction is printed to check the cause (lines 68-74).

Class ErrorNotification

This class is for notifying errors on the application:

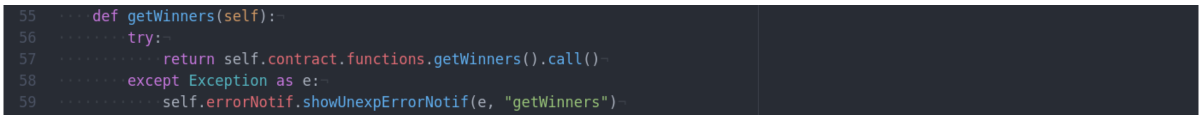


```
class ErrorNotification():
    def __init__(self):
        pass

    def showErrorNotif(self, error, f_name):
        if "replacement transaction" in error:
            error = "Please check if you have any pending transactions. If not, wait a few moments and try again.\n"
            url = "https://ropsten.etherscan.io/address/"+str(self.myAccount)
            error = error + url
        notification.notify(
            title='Error',
            message=error,
            app_name='Text Challenge',
            app_icon=E_ICON
        )
        print(ERROR+"\nERROR")
        print(TSTAMP+"<"+str(datetime.now().strftime("%H:%M:%S.%f"))+"> "+FNAME+"["+f_name+"]")
        print(EXCEPTION+str(type(error))+": "+str(error))

    def showUnexpErrorNotif(self, error, f_name):
        notification.notify(
            title='Error',
            message='Unexpected Error: Please read command line to obtain more info',
            app_name='Text Challenge',
            app_icon=E_ICON
        )
        print(ERROR+"\nERROR")
        print(TSTAMP+"<"+str(datetime.now().strftime("%H:%M:%S.%f"))+"> "+FNAME+"["+f_name+"]")
        print(EXCEPTION+str(type(error))+": "+str(error))
```

Figure 5-54: Notifications.py – ErrorNotification class

The Ropsten network does not allow you to try to mine a transaction when you still have a pending one, *showErrorNotif* method is for detecting that issue.

The *showUnexpErrorNotif* method is for printing any possible exception that can happen during the use of the App. All methods will send the exception to this function, here is an example:



```
    def getWinners(self):
        try:
            return self.contract.functions.getWinners().call()
        except Exception as e:
            self.errorNotif.showUnexpErrorNotif(e, "getWinners")
```

Figure 5-55: Sending notification example

5.3.2.3 Contest Example

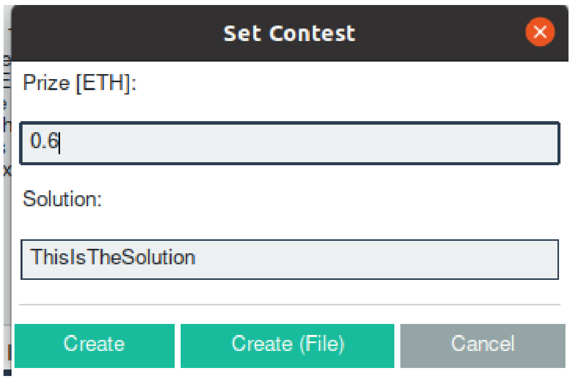
In this section we are going to see the normal development of a contest.

5.3.2.3.1 Contract Deployment

On the Admin's side, we are going to create a new contest. For example, a 'Text from Image' one. This will be the parameters of the contest:

- Solution: ThisIsTheSolution
- Prize: 0.6 ETH

We click in the 'Set Contest' button and fulfil the fields:



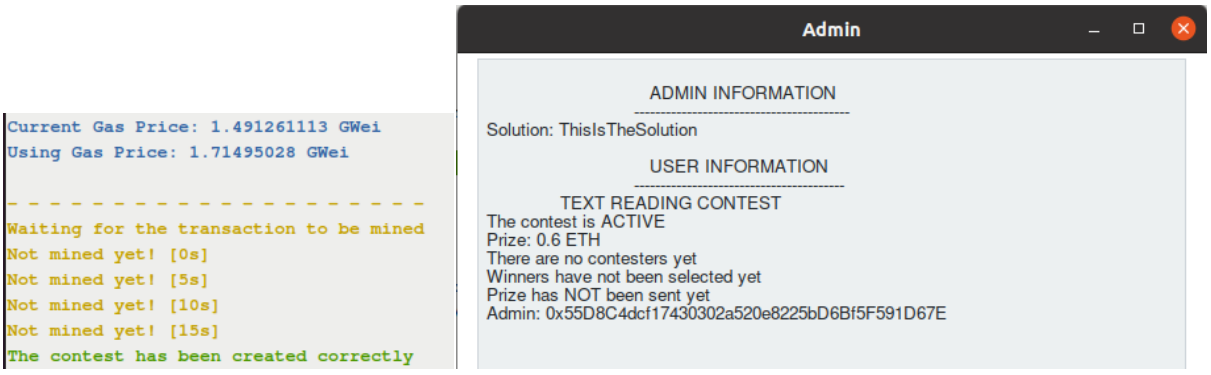
Set Contest

Prize [ETH]: 0.6

Solution: ThisIsTheSolution

Figure 5-56: Setting contests prize and solution

After clicking in 'Create' button, we only need to wait until the transaction is mined to see that the contest has been created:



```
Current Gas Price: 1.491261113 GWei
Using Gas Price: 1.71495028 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Not mined yet! [15s]
The contest has been created correctly
```

Admin

ADMIN INFORMATION

Solution: ThisIsTheSolution

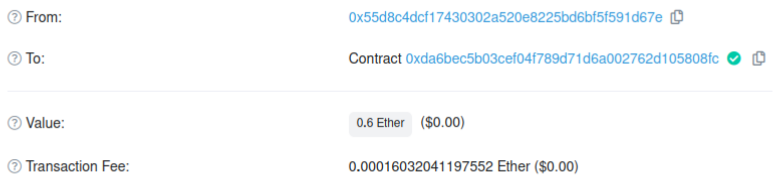
USER INFORMATION

TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.6 ETH
There are no contesters yet
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E

Figure 5-57: Checking price and solutions are correct after transaction has been mined

If we analyze the transaction:
https://ropsten.etherscan.io/tx/0x7dbbc206befa8f986bc548e437237fb80432c8944254922e1418aee96a053822

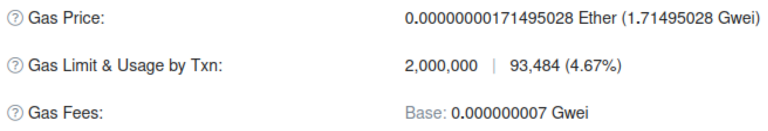
- The prize (0.6 ETH) has been transferred from Admin's account to the Smart Contract's account:



From:	0x55d8c4dcf17430302a520e8225bd6bf5f591d67e
To:	Contract 0xda6bec5b03cef04f789d71d6a002762d105808fc
Value:	0.6 Ether ($0.00)
Transaction Fee:	0.00016032041197552 Ether ($0.00)

Figure 5-58: Price on transaction

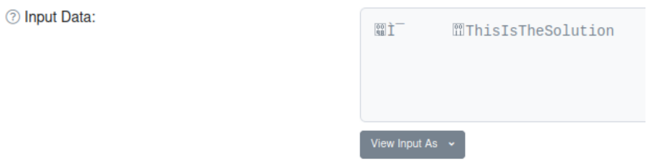
- Gas information; 93484 Gwei has been consumed for mining the transaction:



Gas Price:	0.00000000171495028 Ether (1.71495028 Gwei)
Gas Limit & Usage by Txn:	2,000,000 | 93,484 (4.67%)
Gas Fees:	Base: 0.000000007 Gwei

Figure 5-59: Gas information

- We can see the "attached" solution in the input data:

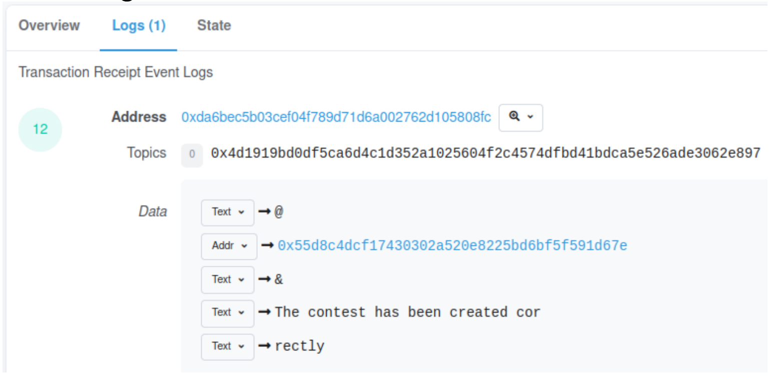


Input Data:

```
ThisIsTheSolution
```

Figure 5-60: Solution on transaction

- In the 'Logs' section we can see the notification from the contract:



Transaction Receipt Event Logs

12

Address 0xda6bec5b03cef04f789d71d6a002762d105808fc

Topics 0 0x4d1919bd0df5ca6d4c1d352a1025604f2c4574dfbd41bdca5e526ade3062e897

Data
```
Text → @
Addr → 0x55d8c4dcf17430302a520e8225bd6bf5f591d67e
Text → &
Text → The contest has been created cor
Text → rectly
```

Figure 5-61: Notification from contract

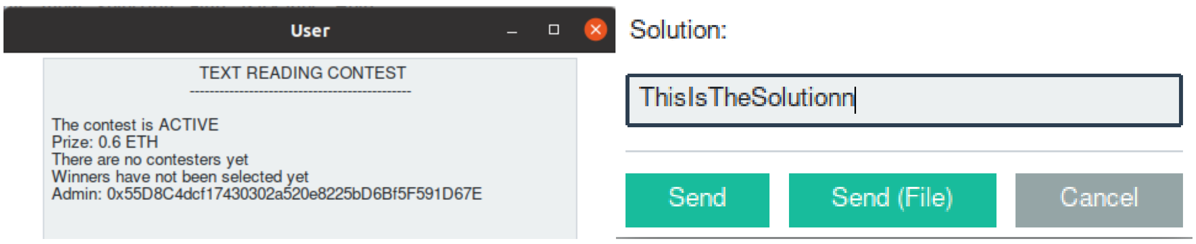
5.3.2.3.2 Participating in the contest

There will be four participants and their respectives solutions will be:

- Participant 1: *ThisIsTheSolutionn*
- Participant 2: *ThisIsTheSolution*
- Participant 3: *ThisIsTheSolution*
- Participant 4: *asdf*

We expect to have the prize half-splitted between Participant 2 and Participant 3.

For Participant 1 we can see that the contest is active and there are no contesters yet:



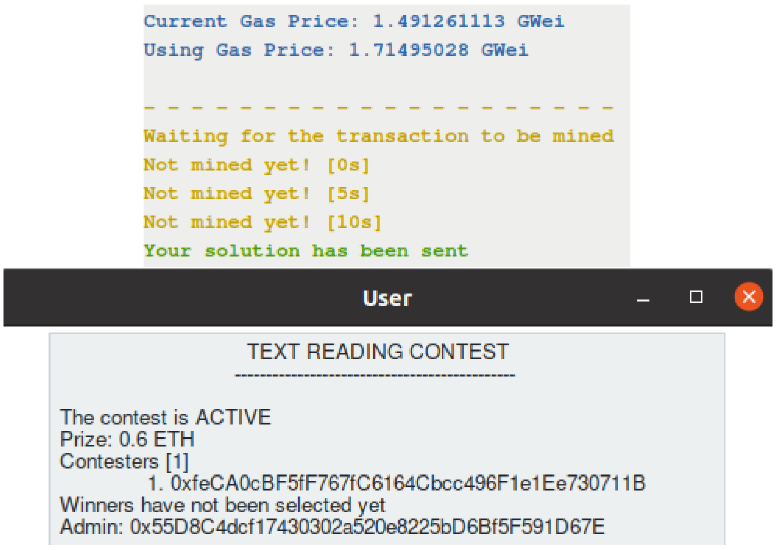
```
TEXT READING CONTEST
--------------------------------------------

The contest is ACTIVE
Prize: 0.6 ETH
There are no contesters yet
Winners have not been selected yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Solution:

ThisIsTheSolutionn

Figure 5-62: Contest status before sending solution

After sending the solution, we can check that the info has been updated:



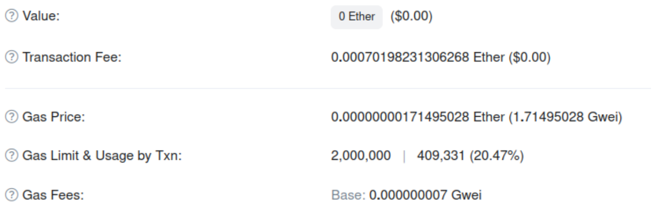
```
Current Gas Price: 1.491261113 GWei
Using Gas Price: 1.71495028 GWei

- - - - - - - - - - - - - - - - - - - -
Waiting for the transaction to be mined
Not mined yet! [0s]
Not mined yet! [5s]
Not mined yet! [10s]
Your solution has been sent
```

```
TEXT READING CONTEST
--------------------------------------------

The contest is ACTIVE
Prize: 0.6 ETH
Contesters [1]
        1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
Winners have not been selected yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-63: Checking participant address has been added to the ‘Contesters’ section

If we analyze the transaction:
https://ropsten.etherscan.io/tx/0x347f0cde6742717a4a4284291c841cf834b899ba01452b64c0936999de5ae8d1

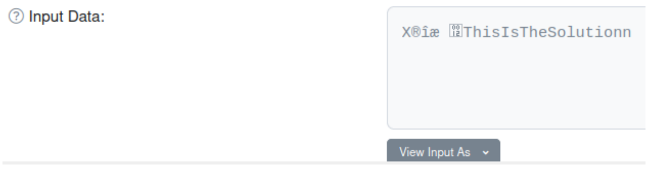
- We can see that the only spent ether is for covering the gas cost:



Value:	0 Ether ($0.00)
Transaction Fee:	0.00070198231306268 Ether ($0.00)
Gas Price:	0.00000000171495028 Ether (1.71495028 Gwei)
Gas Limit & Usage by Txn:	2,000,000 | 409,331 (20.47%)
Gas Fees:	Base: 0.000000007 Gwei

Figure 5-64: Transaction cost

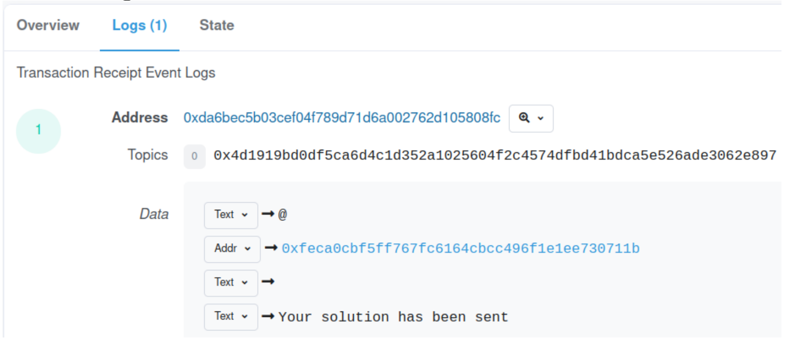
- The solution is "attached" in the input data:



```
Input Data:
X®iæ ThisIsTheSolutionn
```

Figure 5-65: Solution in transaction

- In the 'Logs' section we can see the notification from the contract:



```
Transaction Receipt Event Logs

1   Address  0xda6bec5b03cef04f789d71d6a002762d105808fc
    Topics   0  0x4d1919bd0df5ca6d4c1d352a1025604f2c4574dfbd41bdca5e526ade3062e897
    Data     Text → @
             Addr → 0xfeca0cbf5ff767fc6164cbcc496f1e1ee730711b
             Text →
             Text → Your solution has been sent
```

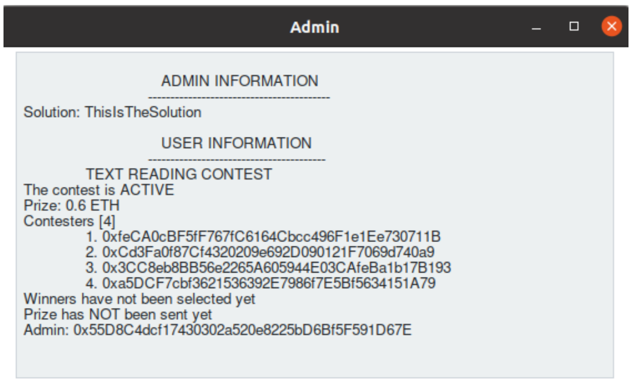
Figure 5-66: Notification from contract

The details from the rest of contesters transactions can be checked here:

- Participant 2: https://ropsten.etherscan.io/tx/0xdc47ce30af0c2b6a6d183808fbae38062943251984b01365c5a86d824bbff17a
- Participant 3: https://ropsten.etherscan.io/tx/0x6037a0607eadf3fd416aec0b0a193f0985672350a8db8e0b3af700f8e5180aa5
- Participant 4: https://ropsten.etherscan.io/tx/0x44e55b54b1d244a944b1be127e125b194759c358ff07a6b7b47ec16d3a2ba155

5.3.2.3.3 Contest resolution

We can see that all participants have sent their solutions by clicking in the 'Info' button:

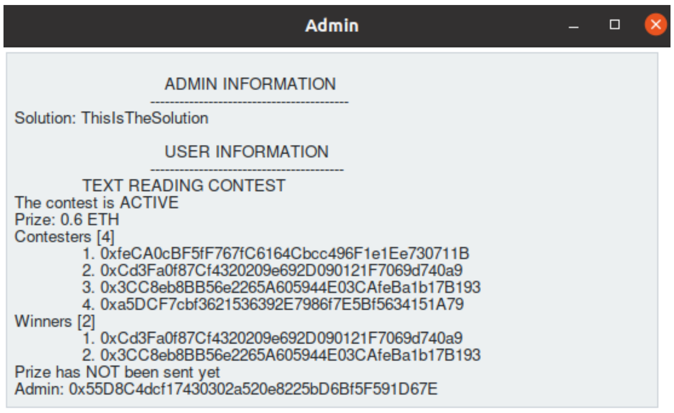


```
Admin

                ADMIN INFORMATION
              ------------------------
Solution: ThisIsTheSolution

                USER INFORMATION
              ------------------------
         TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.6 ETH
Contesters [4]
         1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
         2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
         3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
         4. 0xa5DCF7cbf3621536392E7986f7E5Bf5634151A79
Winners have not been selected yet
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-67: Checking participants addresses in 'Contesters' section

To obtain the winners, we click in the 'Calculate Winner/s' button:

Once the transaction (https://ropsten.etherscan.io/tx/0x5075f7c5ad746713f090ab7fba8680a6d0b7a560f7825d718d887eedb6c69bef) has been mined, we can see that the winners have been selected (but the prize has not being sent yet):

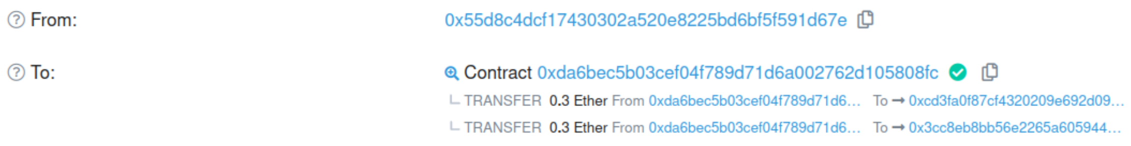


```
Admin

ADMIN INFORMATION
----------------------------------------
Solution: ThisIsTheSolution

USER INFORMATION
---------------------------------------
TEXT READING CONTEST
The contest is ACTIVE
Prize: 0.6 ETH
Contesters [4]
        1. 0xfeCA0cBF5fF767fC6164Cbcc496F1e1Ee730711B
        2. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
        3. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
        4. 0xa5DCF7cbf3621536392E7986f7E5Bf5634151A79
Winners [2]
        1. 0xCd3Fa0f87Cf4320209e692D090121F7069d740a9
        2. 0x3CC8eb8BB56e2265A605944E03CAfeBa1b17B193
Prize has NOT been sent yet
Admin: 0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
```

Figure 5-68: Checking winners addresses

Please note that the addresses match with the addresses of the Participants 2 and 3.

To send the prize to the winners we click on the button 'Send Prize'. Once the transaction is mined, the winners will obtain the prize. If we analyze the transaction: https://ropsten.etherscan.io/tx/0x333a379f0f2b0543fad3a3224b08dad38f39d5886c4a714023a84e5e9a02b85a



```
From:   0x55d8c4dcf17430302a520e8225bd6bf5f591d67e
To:     Contract 0xda6bec5b03cef04f789d71d6a002762d105808fc
          └ TRANSFER 0.3 Ether From 0xda6bec5b03cef04f789d71d6... To → 0xcd3fa0f87cf4320209e692d09...
          └ TRANSFER 0.3 Ether From 0xda6bec5b03cef04f789d71d6... To → 0x3cc8eb8bb56e2265a605944...
```

Figure 5-69: Prize transfer to winners

We can see that 0.3 ETH were transferred to each winner.

5.3.2.3.4 Gas consumption and cost analysis

There are 3 main parameters for calculating how much are we going to spend in the fees for the transaction:

- **Gas Usage:** Required gas for mining the transaction. It can not be modified.
- **Gas Price:** The amount of Gwei for each unit of gas. It can be set; the higher the gas price, the sooner the transaction is going to be mined. If the gas price is too low, the transaction will (potentially) never get mined. It fluctuates widely; for this analysis we are going to consider it as 50Gwei. You can consult the historical gas price here: https://ycharts.com/indicators/ethereum_average_gas_price

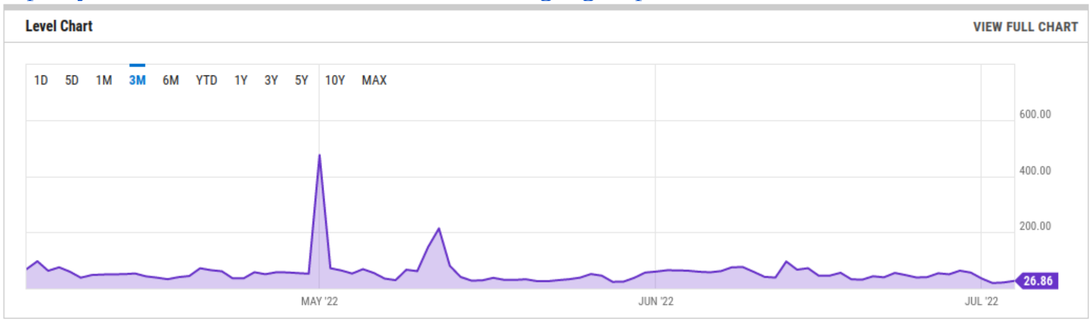


Figure 5-70: Average gas price

Please note that the previous contest example was done in a testnet, not the real network of Ethereum (much less people using the testnet, gas price is much lower).

  - You can also choose the maximum amount of gas you are willing to spend, with the Gas Limit parameter. The maximum allowed Gas Limit is approximately 30.000.000 Gwei.

- **ETH Price:** To be fair, the ETH price has no influence on the cost (in ETH) of the fee. However, it is good to have an idea on how much would be in Euro. Again, ETH price fluctuates widely; for this analysis we are going to consider it as 1ETH = 1000€. You can consult the historical ETH

price here:
https://coinmarketcap.com/es/currencies/ethereum/eth/eur/

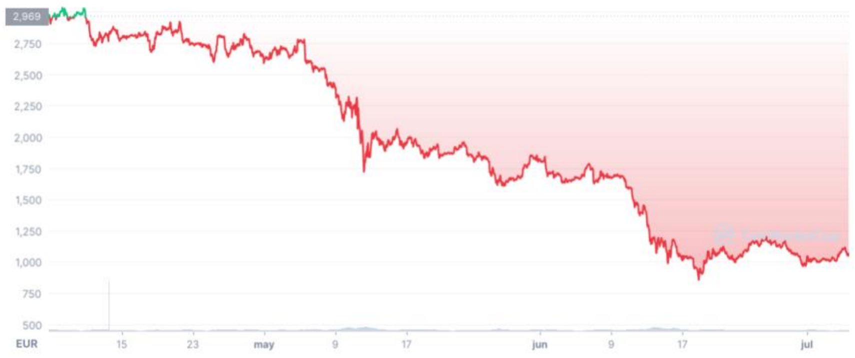


Figure 5-71: Historical ETH Price (€)

In these contests, the transaction that will require more gas is obtaining the score as it is where the biggest computational effort is. Continuing with the 'Text from Image' contest, the larger the solution, the more computational effort is required.

Now, we are going to estimate how much would cost to create and participate in the example contest if it was deployed in the Ethereum main network:

- **Contest creation:** 93484 gas * 50 Gwei/gas = 0'010788 ETH → 10'788€
- **Contest participation:** 409331 gas * 50 Gwei/gas = 0'047218 ETH → 47'218€

If we decide the solution to be larger, we will see how the fees cost is incremented. In fact, we have seen that there is a limitation on the number of characters of the solution (185 characters). To obtain the score of a 185 character string needs more than the maximum amount of gas consumption allowed per transaction.

5.3.2.3.4.1 175 character solution

- **Contest creation:** 229115 gas * 50 Gwei/gas = 0'026338 ETH → 26'338€

Gas Limit & Usage by Txn: 30,000,000 | 229,115 (0.76%)

https://ropsten.etherscan.io/tx/0xc8b73876ab0c1c29204eff3baf4475e655597050df5706b5de5a503b783a6214

- **Participation:** 27857652 gas * 50 Gwei/gas = 3'1963 ETH → 3196'3€

Gas Limit & Usage by Txn: 30,000,000 | 27,857,652 (92.86%)

https://ropsten.etherscan.io/tx/0xfe80de17e4b8a626068ceb5dd48d99cefb11a84e2e480159921c2ed7ab027532

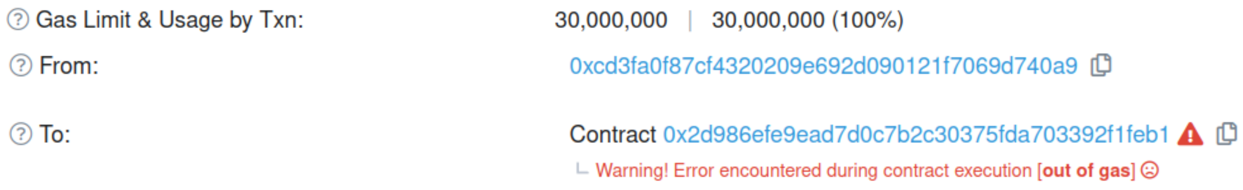
5.3.2.3.4.2 185 character solution

- **Creation:** 229235 gas * 50 Gwei/gas = 0'026294 ETH → 26'294€

Gas Limit & Usage by Txn: 30,000,000 | 229,235 (0.76%)

https://ropsten.etherscan.io/tx/0xbe3b059a64ac6f8770eddb592ce7a22b8da9c0fe224a70610670775506bb501f

- **Participation:** 30000000 * 50 Gwei/gas = 3'4377 ETH → 3437'7€



Gas Limit & Usage by Txn: 30,000,000 | 30,000,000 (100%)

From: 0xcd3fa0f87cf4320209e692d090121f7069d740a9

To: Contract 0x2d986efe9ead7d0c7b2c30375fda703392f1feb1

Warning! Error encountered during contract execution [**out of gas**]

Note that, as the transaction has been reverted, the contester was not able to submit its solution to the contest (and the fee has been paid).
https://ropsten.etherscan.io/tx/0x4da6bf1c566ce2f83b20026ee0b489bd174bbbd7828f16c9ed9aea8dc04c1b4b

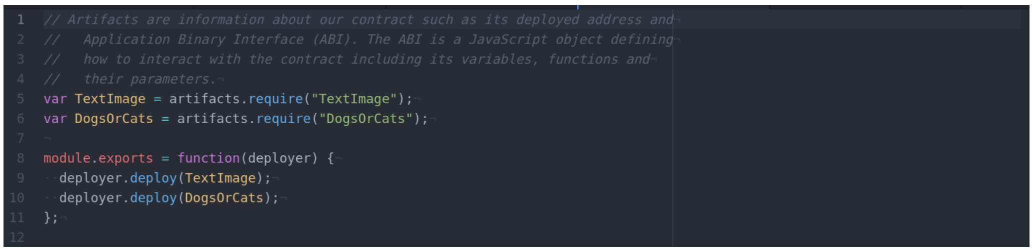
5.3.2.4 Contract Deployment

To deploy the Smart Contract in Ropsten I am using two tools:

- **Infura:** Infura defines itself as “Infura provides the tools and infrastructure that allow developers to easily take their blockchain application from testing to scaled deployment - with simple, reliable access to Ethereum and IPFS”. Basically it provides access to the Ropsten network (and other Ethereum networks). It will give you an endpoint (https or wss) to be used as an API for managing your contract or contracts.
- **Truffle:** The tool that will deploy the Smart Contracts.

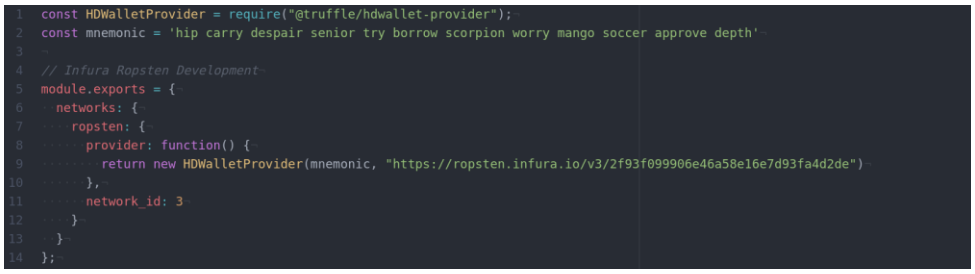
5.3.2.5 Setting the deployment: Configuration files

The configuration for contract deployment is really simple with Truffle (and Infura), it uses JavaScript. It is only needed a file defining which contracts to deploy (deploy_contract in this case) and a file to define the network (truffle-config.js in this case):



```
// Artifacts are information about our contract such as its deployed address and
//   Application Binary Interface (ABI). The ABI is a JavaScript object defining
//   how to interact with the contract including its variables, functions and
//   their parameters.
var TextImage = artifacts.require("TextImage");
var DogsOrCats = artifacts.require("DogsOrCats");

module.exports = function(deployer) {
  deployer.deploy(TextImage);
  deployer.deploy(DogsOrCats);
};
```

Figure 5-72: deploy_contract.js file

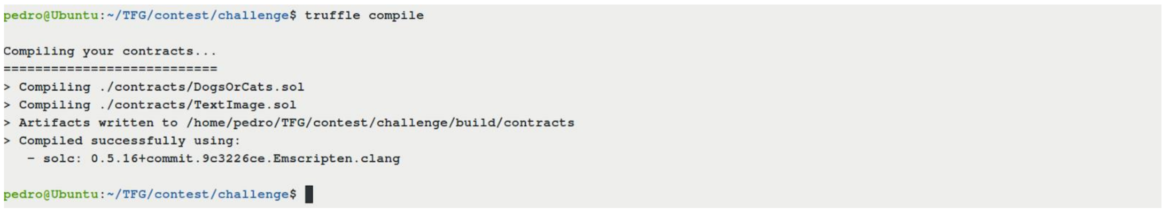


```
const HDWalletProvider = require("@truffle/hdwallet-provider");
const mnemonic = 'hip carry despair senior try borrow scorpion worry mango soccer approve depth'

// Infura Ropsten Development
module.exports = {
  networks: {
    ropsten: {
      provider: function() {
        return new HDWalletProvider(mnemonic, "https://ropsten.infura.io/v3/2f93f099906e46a58e16e7d93fa4d2de")
      },
      network_id: 3
    }
  }
};
```

Figure 5-73: truffle-config.js file

5.3.2.6 Deployment Example

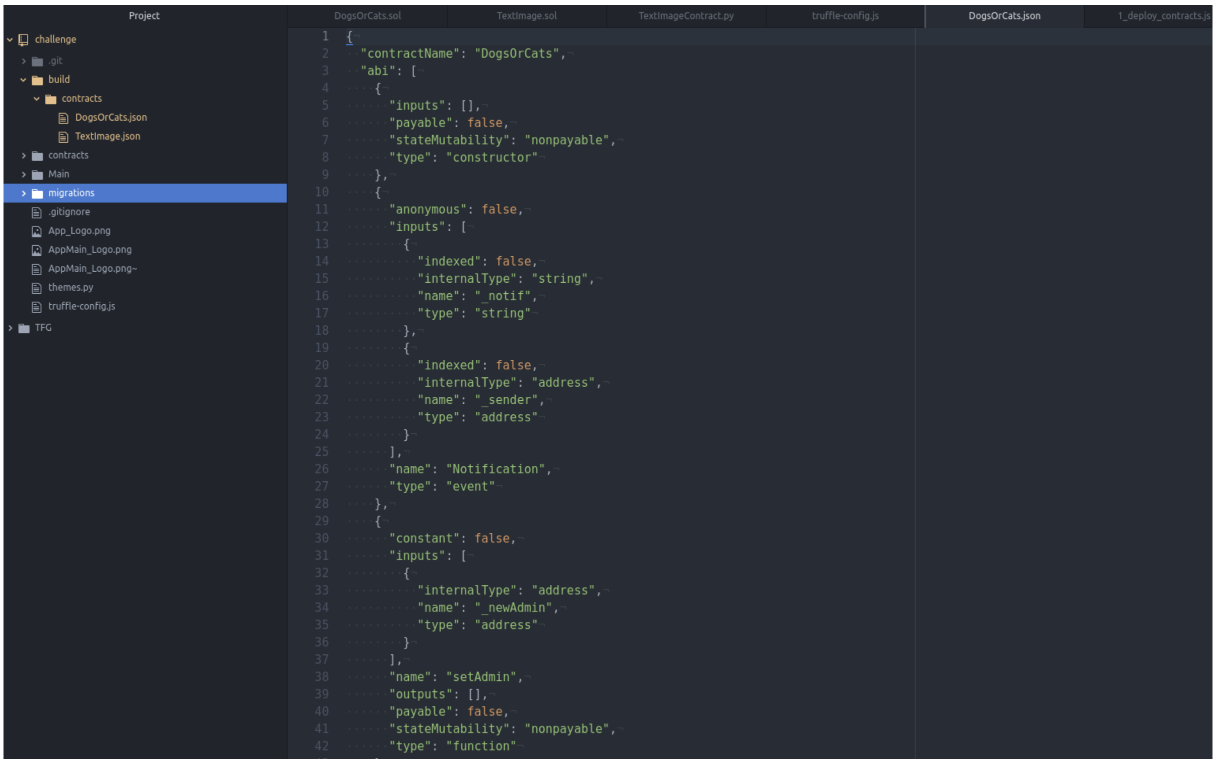
First we have to compile the contracts



```
pedro@Ubuntu:~/TFG/contest/challenge$ truffle compile

Compiling your contracts...
===========================
> Compiling ./contracts/DogsOrCats.sol
> Compiling ./contracts/TextImage.sol
> Artifacts written to /home/pedro/TFG/contest/challenge/build/contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

pedro@Ubuntu:~/TFG/contest/challenge$
```

Figure 5-74: Contracts compilation

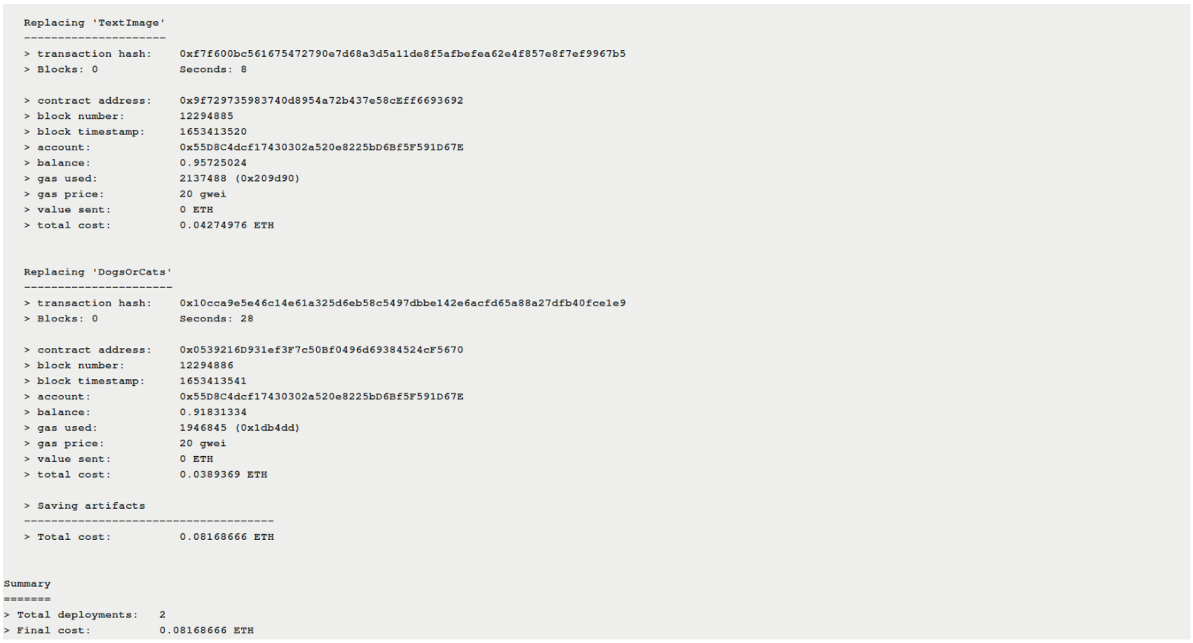
With this, besides obtaining the compilation, we have obtained the ABI_JSON of the contracts:



```
{
  "contractName": "DogsOrCats",
  "abi": [
    {
      "inputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "constructor"
    },
    {
      "anonymous": false,
      "inputs": [
        {
          "indexed": false,
          "internalType": "string",
          "name": "_notif",
          "type": "string"
        },
        {
          "indexed": false,
          "internalType": "address",
          "name": "_sender",
          "type": "address"
        }
      ],
      "name": "Notification",
      "type": "event"
    },
    {
      "constant": false,
      "inputs": [
        {
          "internalType": "address",
          "name": "_newAdmin",
          "type": "address"
        }
      ],
      "name": "setAdmin",
      "outputs": [],
      "payable": false,
      "stateMutability": "nonpayable",
      "type": "function"
```

Figure 5-75: Section of the ABI_JSON of DogsOrCats contract

Then we run *truffle migrate --network ropsten*



```
   Replacing 'TextImage'
   ---------------------
   > transaction hash:    0xf7f600bc561675472790e7d68a3d5a11de8f5afbefea62e4f857e8f7ef9967b5
   > Blocks: 0            Seconds: 8

   > contract address:    0x9f729735983740d8954a72b437e58cEff6693692
   > block number:        12294885
   > block timestamp:     1653413520
   > account:             0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
   > balance:             0.95725024
   > gas used:            2137488 (0x209d90)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.04274976 ETH


   Replacing 'DogsOrCats'
   ----------------------
   > transaction hash:    0x10cca9e5e46c14e61a325d6eb58c5497dbbe142e6acfd65a88a27dfb40fce1e9
   > Blocks: 0            Seconds: 28

   > contract address:    0x0539216D931ef3F7c50Bf0496d69384524cF5670
   > block number:        12294886
   > block timestamp:     1653413541
   > account:             0x55D8C4dcf17430302a520e8225bD6Bf5F591D67E
   > balance:             0.91831334
   > gas used:            1946845 (0x1db4dd)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0389369 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.08168666 ETH


Summary
=======
> Total deployments:   2
> Final cost:          0.08168666 ETH
```

Figure 5-76: Placing contracts in Ropsten Network

5.3.3 Problems and limitations during the development

5.3.3.1 Coding the Levenshtein distance algorithm in Solidity Language

One of the limitations is that Solidity can not compare strings nor characters. The workaround was to compare bytes in sets of 32B; it will work for alphanumeric characters (they are encoded in sets of 32B). It will not work for characters that need more than 32B to be encoded.

The other limitation is that the maximum length of the solution can not be variable. To implement this algorithm, a two dimensional matrix (*distance* variable, line 168 in *TextImage.sol*) is needed. Solidity does not allow this kind of variable to be fully variable and one of the dimensions must be fixed. In this case the decision was to set the maximum length of 200 characters. In case of a future modification, it would be needed to modify the line 168 and deploy a new contract.

5.3.3.2 Moving from local testing environment to real network testing

Local testing was done with Ganache, a software that creates a personal Ethereum blockchain. It is very useful for the initial tests and learning how a blockchain works. I faced 3 main issues when moving to a real network (Ropsten):

- **Building transactions:** Ganache simplifies the transactions, it is configured by default to accept any transaction. In Ropsten (like in MainNet), transactions need to be signed. I had to divide the transaction generation into two functions, *createTx* and *signTxAndWaitNotif* for adding the signing feature with the private key of the user address on every transaction.
- **Notification Handler:** Ganache is configured by default to mine a block for each transaction so transactions are instantly added to the blockchain. In Ropsten transactions take some time to be included in the blockchain. I had to add the feature of checking if a transaction has been included in a mined block or it is still pending.
- **Web3py community:** It is not the most popular library for interacting with Ethereum. Most people prefer to use Javascript as most Dapps use a website as GUI. Most of the time, error descriptions are not very clear, and it is complicated to find information about this error on the Internet as the Web3py community is a small one.

6 Final Conclusions

The main advantage of blockchain technology is supposed to be that it's more secure, but new technologies are generally hard for people to trust, and this paradox can't really be avoided.

- Vitalik Buterin -

After surveying the Ethereum environment, it can be stated that there is a solid technology at the background. The possibility of using a blockchain as the backbone of an application can derive into useful tools. Ethereum has one of the biggest communities of developers and the -increasing- interest from private companies, all indications are that it will continue improving its performance.

Currently, the most popular Ethereum DApps are for exchanges (to swap ETH and tokens), marketplaces (to buy goods or services with ETH) and games. These games are usually focused on having certain assets and use them to obtain better assets from other players. The most popular is Axie Infinity, a game where you can collect and raise fantasy creatures. Besides collecting and raising, you can make a team consisting of 3 Axie's to battle in arena or adventure.

Regarding this project we can conclude that, currently, Ethereum blockchain is not a good solution for calculating the winner/s of Machine Learning contests. This kind of task requires computational power that is a far cry from what the ethereum blockchain can offer (remember the limitation of around 185 characters for the contest, or the high cost of calculating the score of a solution around 175 characters).

6.1 Future Work

Ethereum DApps have high potential, but they have certain limitations as well. As per what we have seen in this project, their main limitation is the computational power. As of today, they are not a good option for applications that require high computational performance.

On this scenario gas cost is expensive and unpredictable; although with the change from Proof of Work to Proof of Stake the gas cost will be predictable, it will probably remain expensive. There are also coding limitations for the 'Text from Image' contest, it is not possible to have a fully dynamic solution length.

That being said, if we are still decided to implement a solution for Machine Learning competitions with Ethereum bockchain, the best option would be to move the backend from being fully decentralized to semi-decentralized. The idea would be to have a "standard" server in charge of obtaining the score of each participant and providing them to the Smart Contract (for example with a REST API).

7 REFERENCES

[1] «Bitcoin Whitepaper,» [En línea]. Available: https://bitcoin.org/bitcoin.pdf.

[2] «CoinMarketCap - Bitcoin,» [En línea]. Available: https://coinmarketcap.com/currencies/bitcoin/.

[3] «CoinMarketCap - Litecoin,» [En línea]. Available: https://coinmarketcap.com/currencies/litecoin/.

[4] «LiteCoin Whitepaper,» [En línea]. Available: https://whitepaper.io/document/683/litecoin-whitepaper.

[5] «CoinMarketCap - Ethereum,» [En línea]. Available: https://coinmarketcap.com/currencies/ethereum/.

[6] «CoinMarketCap - Dragonchain,» [En línea]. Available: https://coinmarketcap.com/currencies/dragonchain/.

[7] «DragonChain Whitepaper,» [En línea]. Available: https://whitepaper.io/document/58/dragonchain-whitepaper.

[8] «Blockchain introduction - medium,» [En línea]. Available: https://medium.com/@4pelumite/blockchain-23d88d1e5832.

[9] «Bitcoin blockchain decentralization - Investopedia,» [En línea]. Available: https://www.investopedia.com/terms/b/blockchain.asp.

[10] «Ripple - Consensus Whitepaper,» [En línea]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[11] «Stellar Consensus Protocol,» [En línea]. Available: https://www.stellar.org/papers/stellar-consensus-protocol.

[12] «Federated Byzantine Agreement,» [En línea]. Available: https://cryptorobin.com/what-is-federated-byzantine-agreement/.

[13] «Ethereum Whitepaper,» [En línea]. Available: https://ethereum.org/en/whitepaper/.

[14] «Ethereum Yellowpaper,» [En línea]. Available: https://ethereum.github.io/yellowpaper/paper.pdf.

[15] «Ethereum Official Documentation,» [En línea]. Available: https://ethereum.org/en/developers/docs/.

[16] «Ethereum Official Documentation - DApps,» [En línea]. Available: https://ethereum.org/en/developers/docs/dapps/.

[17] «Ethereum DApps 2,» [En línea]. Available: https://www.investopedia.com/terms/d/decentralized-applications-dapps.asp.

[18] «Ethereum Dapps 3,» [En línea]. Available: https://ciberninjas.com/cripto-ethereum-dapps/.

[19] «Ethereum DApps 4,» [En línea]. Available: https://www.miethereum.com/smart-contracts/dapps/.

[20] [En línea]. Available: https://www.iebschool.com/blog/dapps-o-aplicaciones-descentralizadas-que-son-y-como-funcionan-finanzas/.

[21] «Ethereum Official Documentation - Data Structures,» [En línea]. Available: https://ethereum.org/en/developers/docs/data-structures-and-encoding/.

[22] «Ethereum Official Documentation - Patricia Merkle Tries,» [En línea]. Available: https://ethereum.org/en/developers/docs/data-structures-and-encoding/patricia-merkle-trie.

[23] «Merkle Patricia Trie,» [En línea]. Available: https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd.

[24] «Merkle in Ethereum,» [En línea]. Available: https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/.

[25] «Etherum Trie,» [En línea]. Available: https://easythereentropy.wordpress.com/2014/06/04/understanding-the-ethereum-trie/.

[26] «Ethereum Data Structure 1,» [En línea]. Available: https://medium.com/hackernoon/getting-deep-into-ethereum-how-data-is-stored-in-ethereum-e3f669d96033.

[27] «Ethereum Data Structure 2,» [En línea]. Available: https://medium.com/codechain/modified-merkle-patricia-trie-how-ethereum-saves-a-state-e6d7555078dd.

[28] «Ethereum Data Structure 3,» [En línea]. Available: https://ethereum.stackexchange.com/questions/55461/how-to-store-storage-trie-in-leveldb.

[29] «Ethereum Data Structure 4,» [En línea]. Available: https://ethereum.stackexchange.com/questions/46068/what-are-the-storage-and-state-tries.

[30] «Ethereum Official Documentation - RLP,» [En línea]. Available: https://ethereum.org/en/developers/docs/data-structures-and-encoding/rlp.

[31] «Ethereum Official Documentation - SSZ,» [En línea]. Available: https://ethereum.org/en/developers/docs/data-structures-and-encoding/ssz/.

[32] «Ethereum Official Documentation - Accounts,» [En línea]. Available: https://ethereum.org/en/developers/docs/accounts/.

[33] «Ethereum Accounts - 1,» [En línea]. Available: https://info.etherscan.com/understanding-ethereum-accounts/.

[34] «Ethereum Accounts - 2,» [En línea]. Available: https://medium.com/coinmonks/ethereum-account-212feb9c4154.

[35] «Ethereum Accounts - 3,» [En línea]. Available: https://www.theengineeringprojects.com/2021/06/ethereum-accounts-definition-types-and-fields.html.

[36] «Ethereum Official Documentation - Transactions,» [En línea]. Available: https://ethereum.org/en/developers/docs/transactions/.

[37] «EtherScan,» [En línea]. Available: https://etherscan.io/.

[38] «Ethereum Transactions 2,» [En línea]. Available: https://info.etherscan.com/understanding-an-ethereum-transaction/.

[39] «Ethereum Transactions 3,» [En línea]. Available: https://www.sofi.com/learn/content/how-do-ethereum-transactions-work/.

[40] «Ethereum Official Documentation - Blocks,» [En línea]. Available: https://ethereum.org/en/developers/docs/blocks/.

[41] «EtherScan - Blocks,» [En línea]. Available: https://etherscan.io/blocks.

[42] «Ethereum Blocks 1,» [En línea]. Available: https://bitsonblocks.net/2016/10/02/gentle-introduction-ethereum/.

[43] «Ethereum Official Documentation - Gas,» [En línea]. Available: https://ethereum.org/en/developers/docs/gas/.

[44] «Ethereum Gas 1,» [En línea]. Available: https://www.investopedia.com/terms/g/gas-ethereum.asp.

[45] «Ethereum Gas 2,» [En línea]. Available: https://www.coindesk.com/learn/what-are-ethereum-gas-fees/.

[46] «Ethereum Official Documentation - EVM,» [En línea]. Available: https://ethereum.org/en/developers/docs/evm/.

[47] «Ethereum EVM 1,» [En línea]. Available: https://coinmarketcap.com/alexandria/glossary/ethereum-virtual-machine-evm.

[48] «Ethereum EVM 2,» [En línea]. Available: https://academy.bit2me.com/que-es-ethereum-virtual-machine-evm/.

[49] «Ethereum EVM 3,» [En línea]. Available: https://www.bitrates.com/guides/ethereum/what-is-the-unstoppable-world-computer.

[50] «Ethereum Official Documentation - Nodes,» [En línea]. Available: https://ethereum.org/en/developers/docs/nodes-and-clients/.

[51] «Ethereum Node 1,» [En línea]. Available: https://ethereum.org/en/run-a-node/.

[52] «Ethereum Node 2,» [En línea]. Available: https://www.quicknode.com/guides/infrastructure/ethereum-full-node-vs-archive-node.

[53] «Ethereum Node 3,» [En línea]. Available: http://www.alchemy.com/overviews/what-is-an-ethereum-node.

[54] «Ethereum Node 4,» [En línea]. Available: https://www.coindesk.com/learn/ethereum-nodes-and-clients-a-complete-guide/.

[55] «Ethereum Official Documentation - Network,» [En línea]. Available:

https://ethereum.org/en/developers/docs/networks/.

[56] «Ethereum Ropsten,» [En línea]. Available: https://github.com/ethereum/pm/issues/460.

[57] «Ethereum Goerli,» [En línea]. Available: https://goerli.net/.

[58] «Ethereum Official Documentation - Consensus Mechanisms,» [En línea]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/.

[59] «Ethereum Consensus Mechanisms 1,» [En línea]. Available: https://www.fool.com/investing/stock-market/market-sectors/financials/cryptocurrency-stocks/proof-of-work/.

[60] «Ethereum Consensus Mechanisms - 2,» [En línea]. Available: https://www.coinbase.com/es/learn/crypto-basics/what-is-proof-of-work-or-proof-of-stake.

[61] «Ethereum Consensus Mechanisms 3,» [En línea]. Available: https://www.technologyreview.com/2022/03/04/1046636/ethereum-blockchain-proof-of-stake/.

[62] «Ethereum Consensus Mechanisms 4,» [En línea]. Available: https://docs.ethhub.io/ethereum-roadmap/ethereum-2.0/proof-of-stake/.

[63] «Ethereum Consensus Mechanisms 5,» [En línea]. Available: https://www.bcbgroup.com/what-is-proof-of-stake-pos/.

[64] «Ethereum Consensus Mechanisms 6,» [En línea]. Available: https://www.investopedia.com/terms/p/proof-stake-pos.asp.

[65] «Ethereum Official Documentation - Token Standards,» [En línea]. Available: https://ethereum.org/en/developers/docs/standards/tokens/.

[66] «Ethereum Tokens - 1,» [En línea]. Available: https://www.investopedia.com/news/what-erc20-and-what-does-it-mean-ethereum/.

[67] «Ethereum Tokens - 2,» [En línea]. Available: https://info.etherscan.com/what-is-erc20-token/.

[68] «Ethereum Tokens - 3,» [En línea]. Available: https://blog.coinbase.com/a-beginners-guide-to-ethereum-tokens-fbd5611fe30b.

[69] «ML - What is 1,» [En línea]. Available: https://www.ibm.com/cloud/learn/machine-learning.

[70] «ML - What is 2,» [En línea]. Available: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML.

[71] «ML - What is 3,» [En línea]. Available: https://www.expert.ai/blog/machine-learning-definition/.

[72] «ML - What is 4,» [En línea]. Available: https://www.geeksforgeeks.org/introduction-machine-learning/?ref=lbp.

[73] «ML - History 1,» [En línea]. Available: https://www.techtarget.com/whatis/A-Timeline-of-Machine-Learning-History.

[74] «ML - History 2,» [En línea]. Available: https://www.techtarget.com/whatis/A-Timeline-of-Machine-

Learning-History.

[75] «ML - History 3,» [En línea]. Available: https://www.lightsondata.com/the-history-of-machine-learning/.

[76] «ML - History 4,» [En línea]. Available: https://www.dataversity.net/a-brief-history-of-machine-learning/#.

[77] «Arthur Samuel,» [En línea]. Available: https://videogamehistorian.wordpress.com/2014/01/22/the-priesthood-at-play-computer-games-in-the-1950s/.

[78] «Rosenblatt's perceptron - TowardsDataScience,» [En línea]. Available: https://towardsdatascience.com/rosenblatts-perceptron-the-very-first-neural-network-37a3ec09038a.

[79] «Neighbor Algorithm,» [En línea]. Available: https://scikit-learn.org/stable/modules/neighbors.html.

[80] «EBL - Gerald Dejong,» [En línea]. Available: https://link.springer.com/content/pdf/10.1007/BF00114116.pdf.

[81] «Hitton,» [En línea]. Available: https://www.cbc.ca/news/technology/turing-award-ai-deep-learning-1.5070415.

[82] «Google AI beats proffesional Go Player,» [En línea]. Available: http://www.bbc.co.uk/news/technology-35761246.

[83] «Waymo Taxi,» [En línea]. Available: https://www.elconfidencial.com/tecnologia/2020-10-14/waymo-coches-autonomos-conductores-google_2787032/.

[84] «GPT-3,» [En línea]. Available: https://www.cnbc.com/2020/07/23/openai-gpt3-explainer.html.

[85] «ML - Basic Concepts 1,» [En línea]. Available: https://becominghuman.ai/an-introduction-to-machine-learning-33a1b5d3a560.

[86] «ML - Basic Concepts 2,» [En línea]. Available: https://en.wikipedia.org/wiki/Data_science.

[87] «ML - Basic Concepts 3,» [En línea]. Available: https://tvst.arvojournals.org/article.aspx?articleid=2762344.

[88] «ML - Basic Concepts 4,» [En línea]. Available: https://tvst.arvojournals.org/article.aspx?articleid=2762344.

[89] «ML - Basic Concepts 5,» [En línea]. Available: https://www.ibm.com/cloud/learn/neural-networks.

[90] «ML - Basic Concepts 6,» [En línea]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network.

[91] «ML, NN & DL classification - IBM,» [En línea]. Available: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks.

[92] «ML - Types 1,» [En línea]. Available: https://en.wikipedia.org/wiki/Supervised_learning.

[93] «ML - Types 2,» [En línea]. Available: https://www.ibm.com/cloud/learn/unsupervised-learning.

[94] «ML - Types 3,» [En línea]. Available: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

[95] «ML - Types 4,» [En línea]. Available: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/.

[96] «ML - Classification vs Regression,» [En línea]. Available: https://prutor.ai/classification-vs-regression/.

[97] «ML - Performance 1,» [En línea]. Available: https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15.

[98] «ML - Performance 2,» [En línea]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

[99] «ML - Performance 3,» [En línea]. Available: https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15.

[100] «Kaggle, Getting Started,» [En línea]. Available: https://www.kaggle.com/getting-started/44916.

[101] «Types of Competitions - Kaggle,» [En línea]. Available: https://www.kaggle.com/docs/competitions.

[102] «Ethereum Transactions 1,» [En línea]. Available: https://www.quicknode.com/guides/knowledge-base/what-are-ethereum-transactions.