Spring 2020

A Code Reputation System Using AI and Blockchain Technology

Jeremy Chau
San Jose State University

A CODE REPUTATION SYSTEM USING AI AND BLOCKCHAIN TECHNOLOGY

A Thesis

Presented to

The Faculty of the Department of Computer Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Jeremy N. Chau

May 2020

The Designated Thesis Committee Approves the Thesis Titled

A CODE REPUTATION SYSTEM USING AI AND BLOCKCHAIN TECHNOLOGY

by

Jeremy N. Chau

APPROVED FOR THE DEPARTMENT OF COMPUTER ENGINEERING

SAN JOSÉ STATE UNIVERSITY

MAY 2020

Wencen Wu, Ph.D.	Department of Computer Engineering
Mahima Suresh, Ph.D.	Department of Computer Engineering
Nima Karimian, Ph.D.	Department of Computer Engineering

ABSTRACT

A CODE REPUTATION SYSTEM USING AI AND BLOCKCHAIN TECHNOLOGY

by Jeremy N. Chau

Open source development is a method of development in which source code is developed using the combined skills of the public. A few examples of this sort of development would be the Chromium and Linux kernel projects hosted on GitHub. Open source development offers a wide variety of benefits and disadvantages. A main concern is when many people blindly trust developers without second thought. There is a lack of a rating system that judges those who develop code through open source means. This can call into question the quality of the code being written by the individual as well as any projects he or she has worked on. This project seeks to fill in this gap by developing a rating system using artificial intelligence (AI) to create ratings for GitHub profiles and store them safely and securely onto a blockchain to prevent unwanted manipulation of the ratings. The AI system was able to create ratings for profiles not currently on the blockchain and store the newly created rating onto the blockchain. The user can then access the records through a web application. Despite the success, the algorithm can be improved in the future.

ACKNOWLEDGEMENTS

I would like to acknowledge my advisors, Dr. Kaladhar Voruganti and Dr. Wencen Wu, for their guidance in helping me complete this project. I stumbled upon many obstacles and they were able to help me navigate on how to overcome them. Their knowledge on the subject allowed me to worry about other aspects of the work that needed to be done. It was a pleasure working alongside them on this endeavor.

I would also like to thank my thesis committee members, Dr. Mahima Suresh and Dr. Nima Karimian, for taking time out of their busy schedule to be on the committee and help provide any helpful suggestions for the paper. I would also like to thank all my fellow students and friends, both undergraduate and graduate, for encouraging me to push on forward when challenges came up.

Finally, I would like to thank my family for supporting me during this time as I know it was not easy dealing with my busy schedule as I worked on this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Artificial Intelligence - AI
Application Binary Interface – ABI
American Standard Code for Information Interchange – ASCII
Application Programming Interface – API
Comma Separated Values – CSV
JavaScript Object Notation – JSON
Megabyte – MB
Stochastic gradient descent – SGD

1 INTRODUCTION

Open source development is a great way of creating code, but it is also a great way to find loopholes in the code. Open source development is the process in which source code is crowd sourced and made it publicly available for use. One of the many benefits to open source development is the increase in skill for those who help develop the code base and those who use the code base. With the increase in popularity of open source development, many people who have no experience with coding have begun to experiment with using open source code and discover libraries that could aid their projects, both personal and professional. One of the few drawbacks to open source is that the code is online for anyone to see. Anyone can look at the code and see how it works, for better or worse. Most of the time open source development relies on eliciting aid from the public, which theoretically allows for a much faster development of code and a higher quality due to the number of developers working on the code.

For example, the internet browser Mozilla Firefox was developed using open source means. It is a great example of how a company can ask for aid from the public in the creation of an innovative product. On the flip side, an example of how vulnerabilities can be found and exploited is the recent Equifax breach in 2017. Pittenger [1] explains that Equifax used the open source platform Apache Struts to host their web application. In March 2017, a patch was available for users to update their software, but Equifax would not update their software until after the attack happened. The reason that the attack was so successful was because the platform had publicly posted online about the vulnerability. Attackers simply needed to read about the vulnerability and use it against

all websites using the platform and find the ones that had not updated their software. Users who updated their software beforehand would be safe, but those who did not would be affected by the vulnerability. However, this is not the only way an attacker can find vulnerabilities in a code base.

Often, there is an unspoken trust between developers and users that the code is made with the utmost integrity. Many users do not bother looking through the entire library and perform extra research on the repository before considering the implementation of the library. They would rather search for the desired parts of the code and ignore the rest. If there were any flaws or malicious pieces of code in the code base, they would never know. The damage caused by an unrevealed malicious library is immeasurable. If the developers wanted to cause some sort of damage, the user would be unable to prevent the attack from happening until it was too late. This sort of attack would be known as an insider attack. The attack is very dangerous and difficult to prevent and track. There should be a system that can create some sort of trust rating that can help users make the decision to trust the developers or not.

Machine learning is a great way of creating these ratings. Machine learning is a method of performing statistical analysis on a set of data to obtain a pattern. Using this method, users can perform statistical analysis on the data of developers and give them a rating. According to Schnebly et al. [2], an artificial intelligence (AI) system has been shown it can detect Twitter bots by simply looking into a few different statistics of a Twitter account. Schnebly's model considered profile statistics like the age of the account, the followers and following ratio, the tweets to days since creation ratio, and a

few others. Based on these statistics, their model was able to detect bots about 90% of the time. Given that their model had to guess whether an account was a bot based solely on the statistics of the account and was able to obtain 90% accuracy shows what machine learning can do.

Machine learning is a well-suited tool to tackling problems like classifying bot accounts; however, there are pitfalls to their design. One pitfall is that the learning algorithms need access to unstained data. Machine learning is only as good as the data it is being given. In addition, users need to be able to trust the results given by the algorithm. If the results produced by the algorithm are tampered with where they are being stored, then the results will become useless. Using blockchain technology could be a good solution to this problem as it allows managers in charge to keep tabs on who accesses the data or the results and aids in securing the data. Blockchain use could prove that the collected data are from reliable sources and that none of the raw data have been tampered with. In addition, its use could show that only authorized accounts have accessed the results of the AI.

1.1 Contributions of the Thesis

The contributions of our work are as follows:

- We propose a rating system web application that calculates ratings for GitHub profiles and displays them for the user and to store the rating onto a blockchain for secure storage.
- We used GitHub's API to gather profile statistics to create a dataset for the purpose of training and testing the AI.

- Two finalized models have been developed that can be used for the application: a random forest implementation and a neural network. Extensive testing went into determining the better model and random forest won out.
- A smart contract was used to direct the flow of data and interactions between the web application and the blockchain.

1.2 LITERATURE REVIEW

The works in [3]-[5] gave us confidence that creating a rating system using an AI system was possible. According to Valecha et al. [3], while the data wildly varied between different customers and reasons for a purchase, the random forest model was able to make highly accurate guesses. Hu [4] reinforces the idea of using an AI controlled rating system. Hu developed a rating system to determine how well a post-graduate student would be able to pay back bank loans. This was a difficult task as there are a wide variety of reasons as to why a payment on a loan was made. In addition, the system considered that the person would have to be making recurring payments when paying off the loan. The system was able to identify if the student would have any trouble making the recurring payments based off the record of the student. Chen et al. [5] developed a rating system that would use text reviews as part of their dataset. This has some interesting implementations because our solution could possibly do the same in that we could analyze and review the actual code uploaded on GitHub to supplement the rating system and make the ratings more accurate. These findings pushed us to consider neural networks a stronger candidate than before.

Using blockchain as a storage medium is not a new idea. Both [6,7] discuss using a log storage system that implements blockchain, as a way of creating logs to allow others to view any activity involving a vast amount of stored data. The application for the version demonstrated by Wang et al. [6] is intended more for commercial or governmental purposes whereas the implementation from Kumar et al. [7] is meant for more private use as companies aim to track access to sensitive data. Wang et al. [8] discusses how smart contracts could be used to interact with the blockchain and what could be done to improve smart contracts in their current form.

When it comes to using the blockchain and figuring out how to interact with it, Gao et al. [9] and Xu et al. [10] help show how blockchain can be used to view information passed between entities. Gao et al. [9] discusses how their work in the financial sector relies on having accurate data. The separate entities need to get information from each other without needing to directly communicate with each other. Retrieving information off the blockchain and to know where the information came from is invaluable and allows for accountability as the person in charge can determine the source of suspicious activity. Xu et al. [10] gives the reader the concept that an electric company, a bank, and customers can be connected through the blockchain. Having the end users interact with a smart contract to pay the electric company though their bank account is one of the various ways blockchain technology can change our lifestyle. Finally, Xu et al. does note that the security of the chain must have priority above all else due to how sensitive the data stored within the chain are. The motivation for security is part of the reason for

choosing to use the technology. Having a rating system means nothing if developers can change their own scores without letting the system know about it.

The idea of combining AI system and blockchain is an innovative one. Currently, developers have considered these topics to be separate. However, that idea is slowly changing, and new ideas about the combination of these topics appear every day. Salah et al. [11] describes a few flaws AIs and blockchain have and how they could be solved by having them work together and gives readers a glimpse into a few examples of applications. Currently, AI systems are developed with a centralized database holding all the necessary data from which they use to train against and learn to make informed decisions on. But using a centralized database comes with risk as the data can be subjected to hacking and manipulation and allow a multitude of AI systems to produce erroneous results. In today's environment, blockchain's usage primarily lies in the financial sector. Furthermore, companies have started to take notice of how blockchain could be applied to different areas of life if done properly.

An example would be the Ethereum blockchain, which heavily bases their current structure from Bitcoin's structure but has found a way to allow the amount of data per block to be larger than 1 megabyte (MB). Ethereum accomplished this by introducing the concept of smart contracts which allow for not just money but data to be sent and stored on the blockchain. Wehbe et al. [12] proves how blockchain can be used to store medical records. Being able to store these records would allow data to be accessed by anyone who needed information on the patient. Medical professionals would no longer have to jump through red tape to get to their patients records because the patient gives them the access.

However, the storage capability will come at a cost. In Ethereum's case, it costs money to store data on the chain, as the stored data are taking up space on all Ethereum nodes. In addition, many developers in the field agree that creating blockchain applications are not hard, but that getting users for these applications is the hard part. This problem can potentially be addressed with the introduction AIs as users might be more willing to work with the application if an AI system helps keep it working properly. An example of blockchain helping an AI system would be that the algorithms work better when the data fed to the system comes from a reliable source. Blockchain technology has a way of verifying that the data does come from a reliable source and keeps the AI system running efficiently and with little error. The web application presented in this thesis is an example of how blockchain technology can enhance the AI system by securely storing data to solve a problem.

2 PROBLEM STATEMENT

With an influx in the use of open source code, we want to assure users that the developers publishing code online are trustworthy. Open source development is a great way of helping the public get access to a code base without forcing them to create it themselves. Unfortunately, it is also a good way of taking advantage of those who do not know any better. For any given problem with an available solution, there could be a single library or a multitude of libraries that solve the problem for the user. It is a game of random chance for inexperienced users in recognizing which library they should choose. Creating an application to reduce the probability of an unsuspecting developer using a harmful library is the goal of this paper.

To accomplish this goal, we want to create a rating system in which an AI system takes in a GitHub profile and produces a rating that shows how trustworthy the user is. This rating system would allow users to check on how skilled and trusted the user is. We use an AI system to review a user's statistics and give a rating based on what the user has done so far on GitHub. We want to use an AI system because manually reviewing and rating millions of profiles is not feasible, and automation makes the creation of the system easier. The program we develop will be trained to be smart enough to make intelligent choices due to the wide variety of information it will be given by the GitHub profile information. The average skill of the public is always changing and using a static equation to output a rating does not allow for any flexibility for future uses. A static equation is unable to update itself if the average skill of the public rises or falls. The AI

will be able to adapt to new data and have the model retrain itself based on new sets of data.

To build our AI, a dataset needs to be created to train the AI. Unfortunately, datasets on GitHub profiles do not exist, which means we will have to create our own. However, another problem arises from the creation of our own dataset. How do we create the base ratings for the profiles whose data we have collected with minimal bias? It was decided that using an unsupervised clustering algorithm would do the best job as these types of algorithms group similar data points together. These groupings will allow the creation of the base ratings. Outliers such as accounts with a large number of points in every category or none at all needed to be culled from the dataset because this rating system is not geared towards these accounts as they do not need the rating. After obtaining the groupings and assigning scores, the AI system can be trained. And yet, using an AI system to produce ratings may not be good enough.

Developers who score low may want to change it without putting forth the effort required to raise the score. To prevent such manipulation, we have chosen to employ blockchain technology. We can use blockchain to securely store the rating information created by the AI. Normally, a database would be enough to store this sort of information, but we want the ability to detect any attempts in the manipulation of the stored rating values and blockchain allows this capability. We will develop a web application that allow users to view these developer ratings but not allow any changes without the system knowing.

3 AI SYSTEM DESIGN

We tested a variety of different algorithms to solve the problem of using an AI system to create a rating based on GitHub statistics. The first issue that needed to be tackled was to obtain a dataset with which we could train and test our AI. Searching for a previously created dataset proved difficult as it seems that this type of endeavor has never been done before. With no publicly available dataset, we had to create our own. Figure 1 illustrates the process followed to create our dataset. After gathering the necessary GitHub data, a new problem arose. There were no base ratings we could use to train the AI. We would have to create our own but had to make sure it was done without bias. Unsupervised learning would be able to help create the base ratings as it can group similar data points together. Giving basic scores to each of the groups would allow the use of supervised learning since these types of algorithms need to have a base rating to train the AI. There are a few algorithms we will be testing.

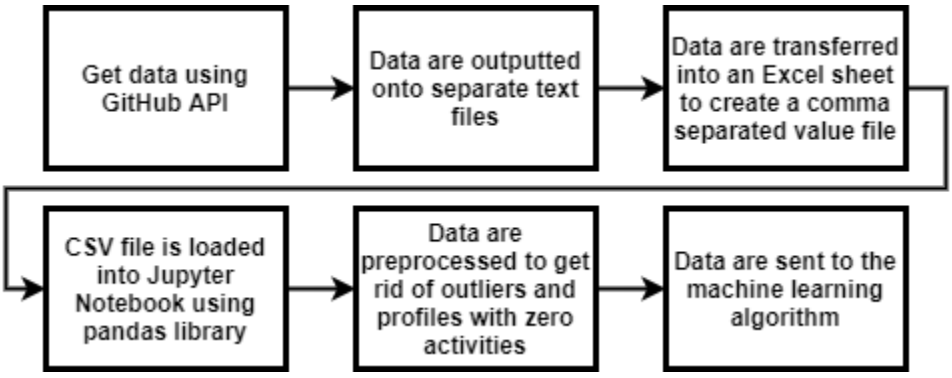

Figure 1: Workflow in creating the machine learning model.

3.1 Data Collection

The creation of the learning models begins with data collection. Since the idea of creating a rating for GitHub profiles using an AI system is new, there are no publicly

available datasets that we can use. A new dataset must be created by ourselves. As seen in Figure 1, we begin by gathering GitHub profile data through the GitHub Application Programming Interface (API) [13]. We decided to collect only 4 different statistics because they were the statistics readily available from the API. We searched for other statistics to gather but we were unable to reliably find them. For example, we wanted to determine if we could use the number of closed pull requests or merged pull requests as a statistic, but we were unable to gather these statistics accurately. After the data had been gathered, we created a comma separated value (CSV) file to hold the input data and their labels which will eventually be fed into the machine learning model. We used Jupyter Notebook to help implement and visualize the testing and training of the model.

To begin the data collection, a program was developed to automatically collect the data. Fortunately, GitHub has a free API that allows anyone access to any publicly available data. In addition, there was an open source library that allows developers to use the python language to access the API called PyGithub [14]. We used this library to access the API and keep the programming language consistent throughout the project.

The dataset consisted of the following attributes: username, total number of forks, stars, watchers, and the date and time of their latest activity. The usernames are obtained using a get function. This function returns a paginated list of usernames in the order they signed up on GitHub. A paginated list is a list that is broken up into multiple pages. After getting the username, a different get function is used to gather details on the username. Within those details another get function will be used to obtain the repositories associated with the user. After obtaining the list of repositories, the program will loop through the

repositories using the respective get functions to find the totals of each attribute. The program will loop through each username after obtaining the totals of the attributes needed for the learning model. Figure 2 illustrates the flow of the program. About 1400 usernames and their associated attributes were gathered during this process. Looking at the dataset and given the random nature of the dataset, we believe that it is a safe assumption that the dataset can be considered balanced. Figure 3 is a sample of what the data in the CSV file looks like.

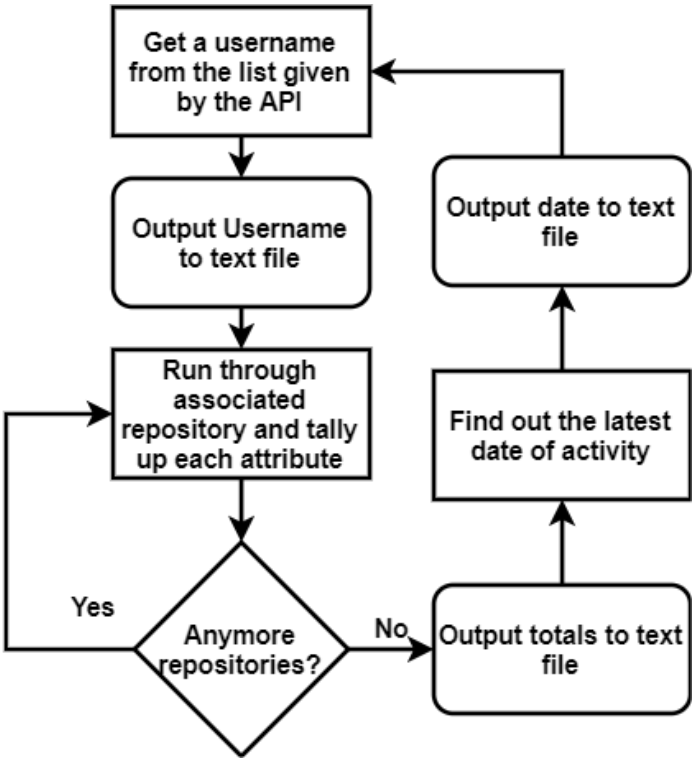

Figure 2: Data collection algorithm.

	User	Forks	Stars	Watchers	Last Commit Date	Cluster	Rating	Unused
0	sbecker	158	743	60	NaN	5	1.1	0
1	aharper	0	3	2	NaN	4	0.0	0
2	stocad	5	1	6	NaN	4	0.0	0
3	adambair	39	174	67	6/10/2019 21:54	3	0.5	0
4	ernesto-jimenez	70	511	41	NaN	5	1.1	0

Figure 3: Raw data from CSV file before preprocessing

3.2 Data Preprocessing

The preprocessing of the data happens after they are loaded into the Jupyter notebook. The pandas library [15] is used the load the data onto the notebook. We decided to only use the profile statistics pertaining to the features of forks, stars, and watchers as these statistics were the most consistent when dealing with information retrieved using the GitHub API. We initially recorded when a user had created their last commit, but we had found that the data gathered for this statistic were not as consistent as the other 3 statistics. For some reason, the data received from GitHub would list the date as not available if it were older than a certain limit. We could not determine what that limit was and so we decided to not feed this information to the unsupervised learning algorithm. Other pieces of information that were cut from our dataset were outliers such as profiles that far exceed the average statistics of most profiles as well as those who have accumulated totals of zero in all their attributes. This is done to prevent the skewing of data when creating the clusters needed from the unsupervised learning methods. We decided to limit the number of forks to 200, stars to 400, and watchers to 500. This combination seems to give best results. The results of the preprocessing can be seen in Figure 4.

	User	Forks	Stars	Watchers	Last Commit Date	Cluster	Rating	Unused
1	aharper	0	3	2	NaN	4	0.00	0
2	stocad	5	1	6	NaN	4	0.00	0
3	adambair	39	174	67	6/10/2019 21:54	3	0.50	0
5	aglasgall	12	30	30	NaN	0	0.25	0
6	marcinpohl	1	7	23	6/3/2019 16:24	4	0.00	0

Figure 4: Data after the preprocessing work has been completed

3.3 Unsupervised Learning

With no previously created rating systems based off GitHub attributes, there are no grounded truths that can be used to judge how good a profile is. We will have to create our own grounded truths. Moreover, it must be done such that there is minimal bias. To accomplish this feat, we utilize unsupervised learning as the results of these algorithms can show similar groupings or clusters. Through these clusters we can assign a rating to the profile. We assigned ratings between 0 and 1. The base rating creation is crucial because without these ratings, the AI system we want to program will have no way of figuring out how to assign ratings. It would be the same as asking a child what the sum of two numbers are without teaching the child how to add numbers together. The child has no idea on how to answer the question as would an AI in the same situation. Having this grounded truth allows the AI system to have a guide as to how it should conduct itself when given the GitHub statistics. Figure 5 shows the various steps that needed to be taken for the AI to be properly created.

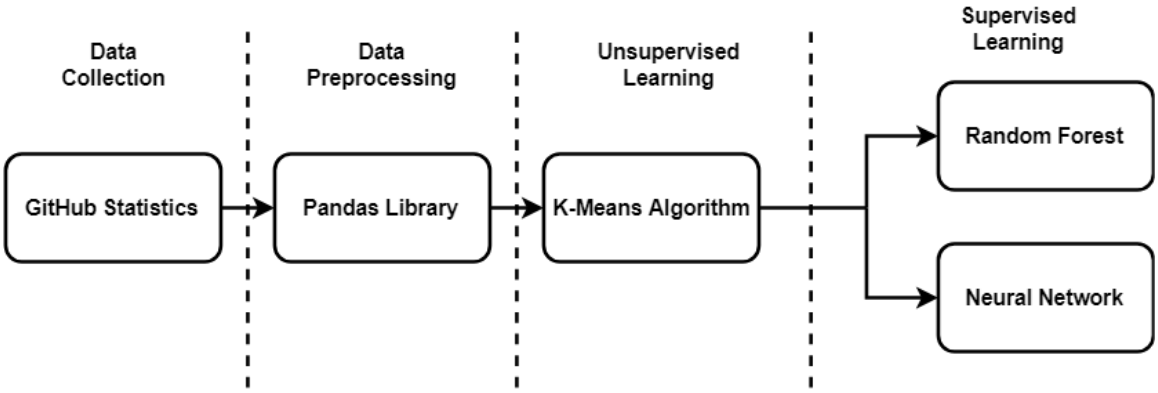

Figure 5: Workflow in selecting the proper machine learning model.

The K-means clustering algorithm is a great way to group similar observations together [16]. In our case, we will be using it to create the grounded truths or ratings missing from our dataset. As stated before, we are using this algorithm to create unbiased ratings. We do this by grouping the observations together and then assign values based off the groupings. Let u_k represent a cluster center that observations will gather around. The algorithm works by initializing a user defined amount of cluster centers, u_k. Let x_n be the observation points and r_{nk} represent the cluster membership, which will be used to determine which cluster each point belongs to. To determine r_{nk}, Eq.1 is used.

$$r_{nk} = \begin{cases} 1: argmin(x_n - u_k = k) \\ 0: otherwise \end{cases} \quad (1)$$

r_{nk} is only 1 when the value of k provides the closest cluster center. Otherwise it has a value of 0. Then let J represent the distortion or how close or far away a point is from the cluster center. We determine that distortion through the following equation.

$$J = \sum_n \sum_k r_{nk}(x_n - u_k)^2 \quad (2)$$

In simple terms, this equation is the sum of distances of every sample from its cluster center. The algorithm seeks to find the optimal cluster centers that provide the least

amount of distortion J. Using (1), we can find u_k We take the derivative of Equation 1 with respect to u_k and equating it to 0. The resultant derivative can be found below.

$$\sum_n r_{nk} \left(2(x_n - u_k)(-1)\right) \tag{3}$$

Rearranging and simplifying (3) by splitting the sigma over the terms, dividing out the 2, moving u_k out of the sigma as it is independent of n, and moving the rest of the terms to the hand side gives us (4).

$$u_k = \frac{1}{\sum r_{nk}} \sum_n r_{nk}\, x_n \tag{4}$$

This equation allows for new cluster centers to be found that best fit the sample data. The algorithm will loop back to find the cluster membership of the sample data with the new cluster points until it finds that the new cluster center points do not change, or it finds that distortion J makes minimal changes between iterations. We will be using the scikit-learn library [17] to access the k-means algorithm and produce the clusters we need for our base ratings.

3.4 Supervised Learning

After using the unsupervised learning to obtain the basic ratings or grounded truths, the supervised learning algorithms can then be used properly. The random forest regression algorithm will be the first algorithm the data will be trained and tested on. Regression algorithms have been chosen due to the nature of the data. Classification is usually done when a machine needs to decide between a defined set of categories/labels whereas regression is much more appropriate for open-ended problems.

3.4.1 Random Forest Regression

Random forest regression is an ensemble algorithm in which it uses the collection of results to provide a prediction as seen in [18]. Specific to the random forest, it uses numerous decision trees to make its predictions. Having a multitude of trees benefits the algorithm by reducing the amount of over-fitting of the data that decision trees are prone to and helps increase the accuracy of model. When making a prediction, each tree runs a separate algorithm and votes for the value it comes up with and then the model takes the average of the votes to make its prediction. This is illustrated in Figure 6. Before training can begin, the data needs to be split up into four separate groups: training data, training labels, testing data, and testing labels. This is accomplished using functions from the scikit-learn library. The library will also be used to access the functions to run the random forest algorithm. To train the algorithm, we need to decide how many trees are going to be used, and the max depth each tree can be. These two factors greatly affect the amount of time needed to train the model and the amount of error accrued by the model. After figuring out the optimal number of trees and the max depth of them, the training of the model against the training dataset can begin. The model is fitted onto the training data and labels to train the model. To test the model, the prediction function is used on the testing data set. We can compare the results of the prediction with the testing labels created earlier and determine the mean absolute error of the model.

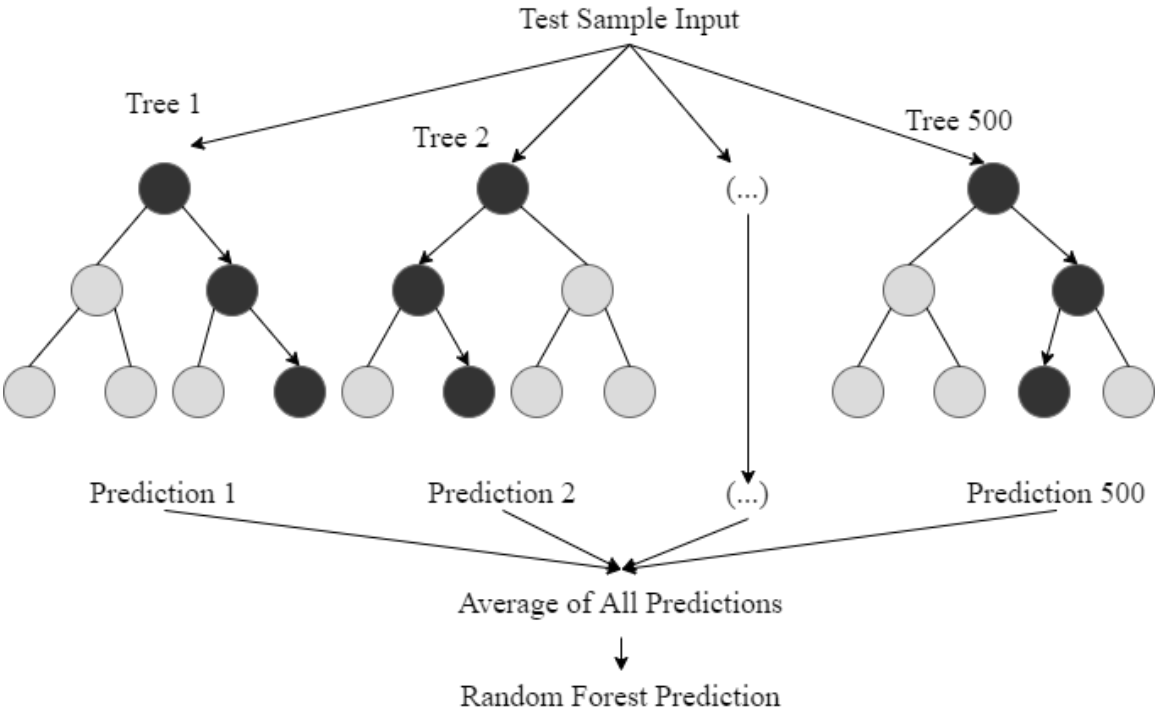

Figure 6: How the random forest algorithm makes a prediction.

3.4.2 Neural Networks

To develop the neural network, we decided to use the TensorFlow framework. The framework provides access to a multitude of functions and tools, which will ease the process of creating a neural network. Through TensorFlow, we access the Keras library [19], which is a very beginner friendly machine learning framework. The framework allowed us to explore and use various functions with ease and allowed the training and testing to be accomplished smoothly. The library contains a couple of activation functions and numerous loss and optimizer functions. For activation functions, it has the sigmoid function and the ReLU function. The equation and graph of the sigmoid and ReLU functions can be found in Figures 7 and 8, respectively.

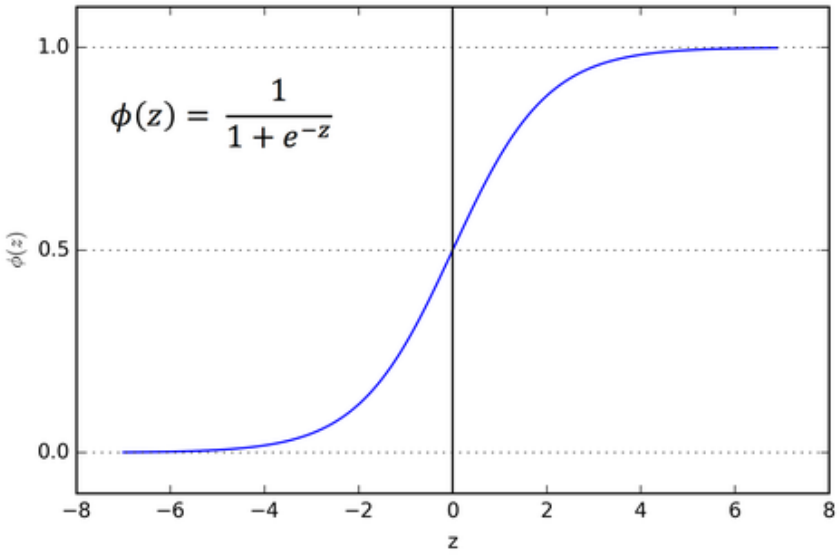

Figure 7: Sigmoid graph and its equation.

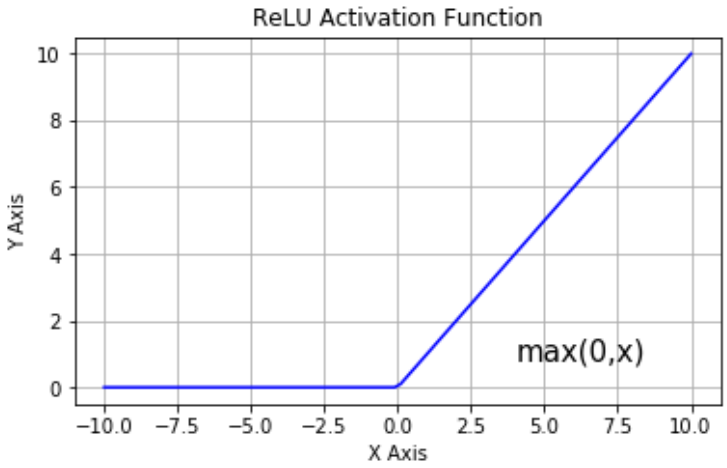

Figure 8: Graph of the rectifier linear unit activation function.

For the applicable loss functions, they are as follows: mean squared error, mean absolute error, mean absolute percentage error, mean squared logarithmic error, squared hinge, hinge, categorical hinge, and logcosh. These listed functions inform the network know how close or far off the prediction created by the network was. Choosing the right function is imperative as we want to reduce the amount of loss obtained here as that would lead to a more accurate prediction. For the available optimizer functions, there are

the Stochastic gradient descent (SGD), root mean square propagation or RMSprop, Adaptive Subgradient or Adagrad, Adadelta, a variant of Adagrad, Adam, Adamax, a variant of Adam, and Nesterov Adam or Nadam. These functions update the weights in each of the node of the neural network with the intent to reduce the value received from the loss functions listed prior. Picking the right optimizer function is crucial during the training phase as the loss and optimizer functions work in tandem to reduce the amount of error produced by the model. With the variety of available loss and optimizer functions, all will need to be tested in combination with the activation functions to discover the best combination. After selecting the activation, loss, and optimizer functions, the model will be trained and tested just like the random forest model. The network will be trained and tested to gather results.

3.5 Model Results and Analysis

An analysis of the results will show whether an AI was developed with satisfactory accuracy. We have to check that the k-means clustering was successful in creating the base ratings needed for the supervised learning and to decide which algorithm will be used for the AI. When analyzing the k-means clusters, we must make sure that the amount of data points within each cluster are around the same. When making the decision as to which supervised learning algorithm will be chosen, the amount of error each model created is observed to determine the better performing model.

3.5.1 K-means Clustering

Since there are 3 key features being looked at by our model, the k-means algorithm must take account of this fact as special functions will be needed to display the clusters.

As stated previously, we limited the range of the data points taken into the algorithm to prevent the skewing of cluster data. The algorithm was told to produce 5 clusters and each cluster was given a rating of 0, 0.25, 0.5, 0.75, or 1 based on the points within each cluster. These rating scores are crucial as they will serve as the grounded truth for the machine learning models so they can be trained properly. The output of the model using the testing data can then be compared with the rating data that was given by their cluster to test its accuracy. Figure 9 shows the result of the clustering.

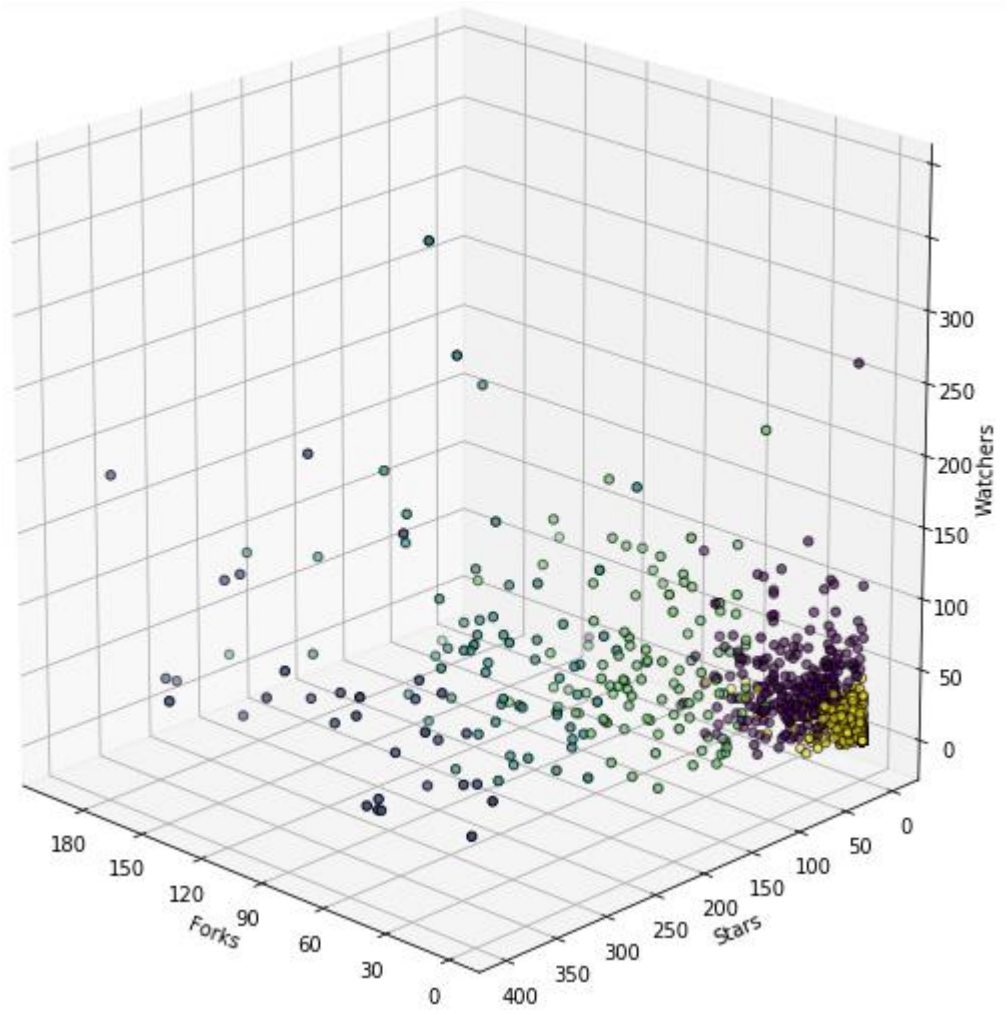

Figure 9: 3D Scatter plot of the observations generated by the Matplotlib library [20].

3.5.2 Random Forest Regression

With random forest regression there are 2 main parameters we need to test to find the best performing combination. To do this, we used the GridSearch functions found in the scikit-learn library. These functions allow us to use the cross-validation functions which will help us find the optimal number of trees and tree depth. We limited the number of trees to a max of 100 and a max tree depth of 30. We used these hard limits because having too many trees or having too tall of a tree ruins the performance of the model during training and in turn bring down the performance of the rating system. The negative mean absolute error scoring results were compared against each other to determine which the optimal combination of trees and tree depth. As seen in Figure 10, it was determined that a max depth of 16 and the number of trees the algorithm uses is 18. From this run it also determined that the mean absolute error during validation was around 0.0093966. We decided to use the mean absolute error because this is the more common metric used to determine how good an algorithm is when dealing with a balanced dataset. This can be seen in Figure 11. The reason it is shown as negative is specific to the cross-validation score function. With the way the algorithm works, GridSearch always tries to maximize the score parameter or in this case the mean absolute error. The only problem is that a low mean absolute error is desired, so the score gets converted to a negative value to prevent the GridSearch from returning a wrong result.

```
{'max_depth': 16, 'n_estimators': 18}
{'bootstrap': True, 'criterion': 'mse', '
se': 0.0, 'min_impurity_split': None, 'mi
tors': 18, 'n_jobs': None, 'oob_score': F
```

Figure 10: Results of the GridSearch validation.

```
print(score.mean())
```

```
-0.009396638193231086
```

Figure 11: The testing and the calculation of the model's average error.

In Figure 12, the results of a single training and testing run with the parameters found by the GridSearch validation. With an error of 0.01252, it shows that the model makes highly accurate predictions and the predictions that are wrong are not off by much.

```
predictions = rf.predict(xtest)
errors = abs(predictions - ytest)

print('Mean Absolute Error:', round(np.mean(errors), 5))
print('R^2 Score:', round(np.mean(rf.score(xtest, ytest))))
```

```
Mean Absolute Error: 0.01252
R^2 Score: 1.0
```

Figure 12: Mean absolute error obtained in a single run.

3.5.3 Neural Networks

With neural networks, k-fold cross-validation is needed to be used to find the optimal combination of the various functions for the activation, loss, and optimizer parameters. K-fold validation is the process of training and testing the model using the entire dataset to determine the best model parameters. The reason we want to use this process of validation is because the best way to train a model is by giving it as much data as you can as that leads to have the best learning results for the model and vice versa for testing to get the best validation results. Consequently, that means there is no data to be used during

the testing phase or training phase in either situation. K-fold works around this dilemma by rotating which piece of data is used as the training and testing sets. This allows the entire dataset to be a part of the training and testing sets for a model and from these sets, we can take the averages of the accrued mean error to determine which model has the best fitting combination. Below in Tables 1 and 2 are the results of the cross-validation taking place on the wide variety of neural network combinations.

Table 1: Average Error for Neural Network with Sigmoid Activation

Loss Functions	Optimization Functions						
	RMSProp	SGD	Adagrad	Adadelta	Adam	Adamax	Nadam
Mean Absolute Error	0.0243	0.1083	0.05849	0.1741	0.05378	0.05047	0.0571
Mean Squared Error	0.03356	0.05532	0.07153	0.1935	0.05041	0.05045	0.06171
Mean Absolute Percentage Error	0.1143	1.39E8	0.1948	0.1984	0.2009	0.1946	0.2024
Mean Squared Logarithmic Error	0.9416	0.6024	0.6141	0.3425	0.2461	0.6443	0.6254
Squared Hinge	3.6236	1.9051	1.8906	0.4059	3.1343	2.1096	3.0498
Hinge	3.7258	3.6978	3.8118	0.7251	3.6113	3.8591	3.8198
Categorical Hinge	5.2359	3.5496	2.1766	1.4823	4.6206	5.1351	4.5687
Logcosh	0.03095	0.05693	0.09797	0.1858	0.0523	0.05112	0.05914

Table 2: Average Error for Neural Network with ReLU Activation

Loss Functions	Optimization Functions						
	RMSProp	SGD	Adagrad	Adadelta	Adam	Adamax	Nadam
Mean Absolute Error	0.02058	0.03452	0.03748	0.103	0.01523	0.02306	0.01662
Mean Squared Error	0.01714	0.03882	0.04449	0.1176	0.01949	0.02592	0.01788
Mean Absolute Percentage Error	0.03408	NaN	0.0722	0.1858	0.05296	0.05541	0.5403
Mean Squared Logarithmic Error	2.6281	0.09264	0.09049	0.1192	1.2611	0.4785	1.2356
Squared Hinge	13.5934	4.8994	1.821	0.1891	10.6708	5.1658	11.1646
Hinge	17.2885	5.4805	2.954	0.1691	19.1304	7.0248	19.9702
Categorical Hinge	5.923	3.5546	2.8113	0.6879	4.7644	4.7797	4.6336
Logcosh	0.01737	0.0414	0.04587	0.122	0.01795	0.02621	0.02053

According to Table 2, the combination of the ReLU activation function, the mean absolute error loss function, and Adam optimizer function with default parameters created the least amount of error out of all the function combinations at a value of 0.01523. The recommended parameters of the Adam function are the default settings according to the Keras library. In the Figures 13 and 14, you can see the full results of a single test below.

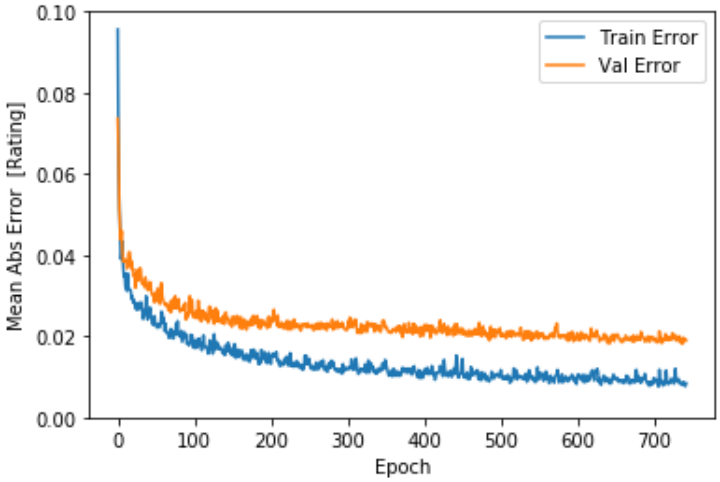

Figure 13: Graph of mean absolute error metrics during the training phase.

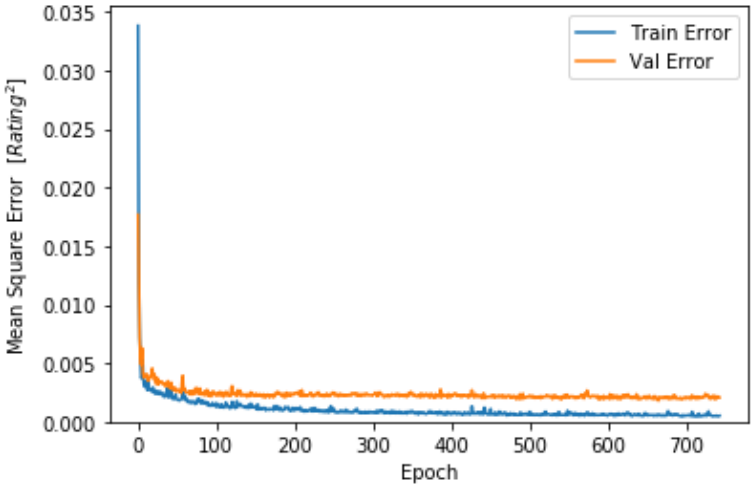

Figure 14: Graph of mean squared error metrics during training phase.

Based on Figures 13 and 14, the training and validation errors accrued by the model during each run of the training phase. The validation error is the error obtained from running a set of data through a trained network. The training error is the error obtained on the training set of data. As time passes, you can see the amount of error between each run steadily decrease. At the end of a random testing phase, the mean absolute error was found to be about 0.0179 as shown in Figure 15. While this value is incredibly low, it

seems the random forest algorithm still obtains a lower amount of error than the neural network as seen in Figure 16.

```
loss, mae, mse = model5.evaluate(normxtest, ytest, verbose=0)

print("Testing set 5 Mean Abs Error: {:5.4f} Rating".format(mae))
```

```
Testing set 5 Mean Abs Error: 0.0179 Rating
```

Figure 15: The mean absolute error calculating during one run.

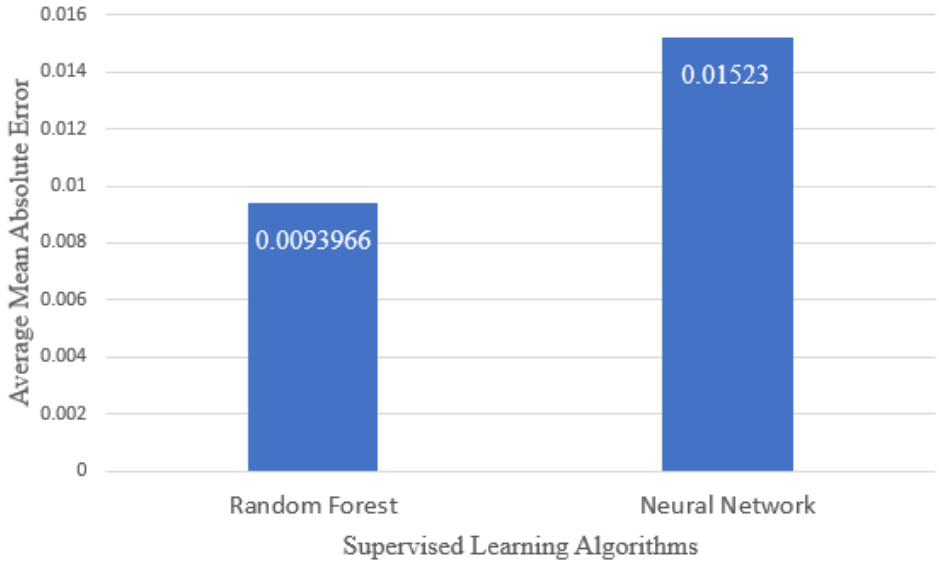

Figure 16: Algorithm comparison graph.

Looking at Figure 17, the graph displays a comparison between the predictions and the actual values. The line striking through the graph is the visual representation of where each dot should be. Based on the results of the graph, the network performs well when calculating the 0 ratings. When calculating the 0.25, the network had a little trouble with the predictions and tends to undershoot. For the 0.5, 0.75, and 1.0 ratings, the network seems to not have too much trouble in the predictions and only needs minor adjustments to get on the line. Figure 18 gives us another view of the prediction error. It appears the amount of error obtained is centered around 0. This means that most of the points that are

not correct are off by a little. This histogram reinforces the data shown in Figure 17 in which we see many predictions for all ratings fall slightly below the correct prediction line.

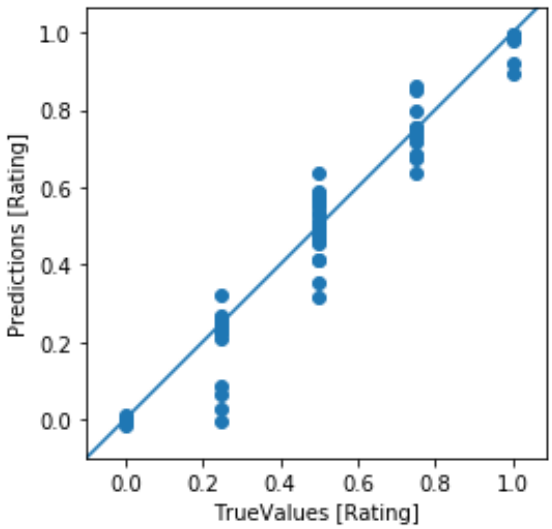

Figure 17: Comparison graph between the predicted and real ratings.

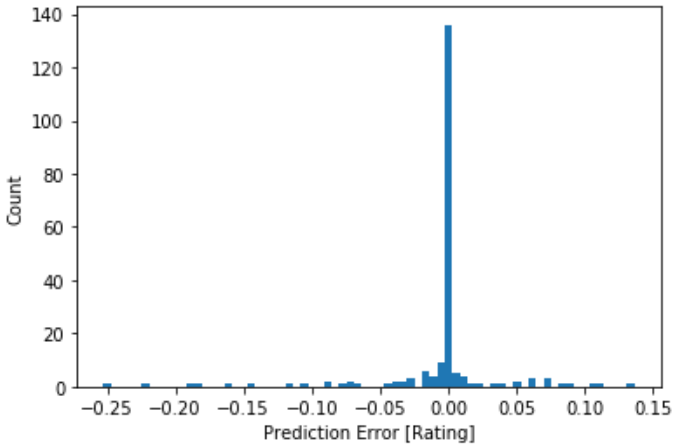

Figure 18: Histogram of prediction error during testing.

4 BLOCKCHAIN

To securely store the data from the machine learning model, we decided to use blockchain to ensure the security of the data. Blockchain technology is based on the idea of creating a public ledger or a public list of records. This section contains a review of blockchain and its properties, the system design of our blockchain application, and the results of our design.

4.1 Technology Review

This section discusses the properties of blockchain and introduces the Ethereum platform we have chosen to use for our application. The blockchain's properties are important to understand as they are the reason that we chose to use it over a normal database implementation. The introduction to Ethereum will explain why it was chosen and how it works compared to Bitcoin's implementation.

4.1.1 Blockchain Properties

The blockchain has several key characteristics such as immutability of data on the chain, decentralization of the data, and traceability and transparency of transactions [11]. These distinct properties of blockchain are the reason that blockchain was chosen over a database implementation. Since our application contains some sensitive information such as the rating associated with each profile, the properties of security, decentralization, and logging capabilities are desired to ensure the success of the application.

4.1.2 Decentralization

Decentralization was a desired trait for the storage due to its benefits over the centralized option. A centralized implementation is one in which all data are stored and

taken from a single central location. Efficiency is paramount in a centralized system as nodes only access relevant data. However, if the central node were ever hacked and its data manipulated, all systems connected to the node would also have their data manipulated. In a decentralized system, this sort of event would almost never happen. All nodes in the system are required to contain the same data. While this is not a very efficient way of using space in the system due to having multiple redundancies, the system will have a very few points of failure since the multiple redundancies throughout the system will be able to rectify any changes to a compromised node. The built-in redundancies act as a deterrent against attacks on a single node. Any attack can be rectified and reversed, but at the cost of reducing the system's storage capabilities. Our application plans to make use of decentralization to ensure that ratings stored on the blockchain cannot be irreversibly changed without the system knowing.

4.1.3 Immutability

The immutability characteristic of blockchain stems from the decision to use a decentralized system. Since all nodes in the system contain the same data, it is very easy to detect if any change to the data is made. If a hacker wanted to make changes to any transaction in the past, he or she would need to change the all data in the block until the current block across all nodes in the system. In addition, this must be done while the system creates new blocks of data. Depending on how large the network is, the computing power needed for this endeavor would be enormous and essentially theoretical. The necessary computing power discourages those that would try to hack an

active system. This immutability will assure users that all stored ratings in the blockchain were created only by the AI system and have not been manipulated in any way.

4.1.4 Traceability and Transparency

Transparency aspect of blockchain stems from the fact that it claims itself as a public ledger. This means that any user of the network can see any transaction in the past; even the very first one that may have been made years ago. This is something that is unheard of when dealing with financial holdings as most companies hide these details. The only thing that is not transparent are the identities of the users. Users are only identified through their public address and do not have to reveal their identity if they do not wish. The traceability works with the transparency aspect as in each transaction, you can see between which addresses was the transaction made and what assets were transferred. Being able to see the addresses is perfectly fine for our use case as it allows us to check where transactions are being made from. If there are any fishy transactions being made, we will be able to detect them.

4.1.5 Ethereum

We have chosen to use the Ethereum network platform as the blockchain we want to demonstrate our work with. The reasons for the choice are due to the accessibility to the vast number of tools available for new blockchain developers and plenty of examples and support we can call upon to learn how to work with the blockchain and fix any problem we come upon later. In addition, Ethereum's smart contract implementation complements the needs of our system in that we can call the contract whenever a new rating has been created.

4.1.6 Platform

Ethereum's blockchain design heavily borrows from Bitcoin's system but adds additional functionality to be able to support applications other than just money [21]-[24]. Just like Bitcoin, Ethereum uses the proof of work system as a way of validating any transaction. However, the way it works differs. In Bitcoin's proof of work version, miners are racing each other to add the next block of transaction. One block holds 32MB of data and can hold anywhere between 1 to several thousand transactions. To solve for the target hash, a complicated equation needs to be solved. Miners race against each other to complete these complex equations so there is no guarantee the solution to the equation is the correct hash. The miner's equipment needs to be able to output thousands of hashes quickly if he or she wants to earn the reward for adding the new block on the chain.

In Ethereum's case, the competition is not as stiff. Miners still run the calculations to prove validation of the transaction, but the creator of the transaction must pay a fee of sending such a transaction. This is found by checking what is known as the gas price of the transaction. Every unit of gas is measured in “gwei”. Refer to Table 3 to see how the conversion from gwei to Ether works. Each transaction will have a gas limit which represents the maximum amount the sender is paying to pay. This gas is used to pay the miners used in the proof of work system. The amount gas used varies from transaction to transaction as the senders manually set the value as a way of enticing miners to work on their transaction in addition to how much data is contained in the transaction. Furthermore, miners can choose what kind of transactions he or she want their system to

work on. They can filter out all transactions below a certain gas price so senders need to be smart when deciding what the gas price should be.

Table 3: Conversion Table to Calculate the Lowest Denomination of Ether, Wei [21].

Ethereum Conversion Table	
Monetary Unit	Converts to
gwei	1,000,000,000 Wei
Ether	1,000,000,000,000,000,000 Wei

In addition to the proof of work difference, how transactions happen also differ. In Bitcoin's version, the system tracks the transaction history associated with each account to determine who has how much bitcoin. Whenever a transaction is made, nodes scan the entire system and need to verify a couple of things: that the sender has a sufficient amount of money in their account and that he or she have not already sent it to someone else recently. After the verification, the transaction gets placed into a block to be mined. Figure 19 illustrates a top-level view of how a transaction happens.

In Ethereum's case, the system tracks the accounts themselves. Each transaction tracks who the sender and receiver are in addition to the other information being transferred between the accounts. The reason why Ethereum can track accounts the way it does is due to its state machine design to keep track of all the information. Each transaction block occurring between accounts move the global state of Ethereum from one state to the next. Figure 20 illustrates how a transaction moves the Ethereum global state. Each transaction is a cryptographically signed set of instructions that was generated

by an externally owned account. An externally owned account means it is an account controlled by private keys as opposed to contract accounts, which are controlled by contract code. Transactions are created by externally owned accounts and hold multiple data components. These components are the nonce, which represents the number of transactions sent by the sender, the gas price, gas limit, the address of the recipient, the amount of Wei to be transferred, a signature that identifies the sender, an initialize function, which is a piece of code that is run once to initialize a new contract account and then discarded, and a data field that is primarily used for message calls. Of all these components, our application makes use of the address of the recipient, the signature that identifies the sender, the initialization function, and the data field used for message calls.

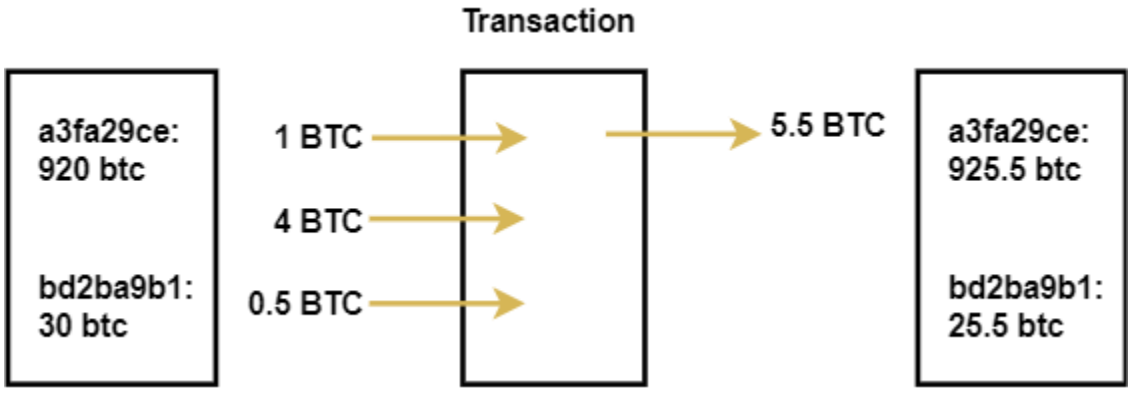

Figure 19: Illustration of Bitcoin transaction.

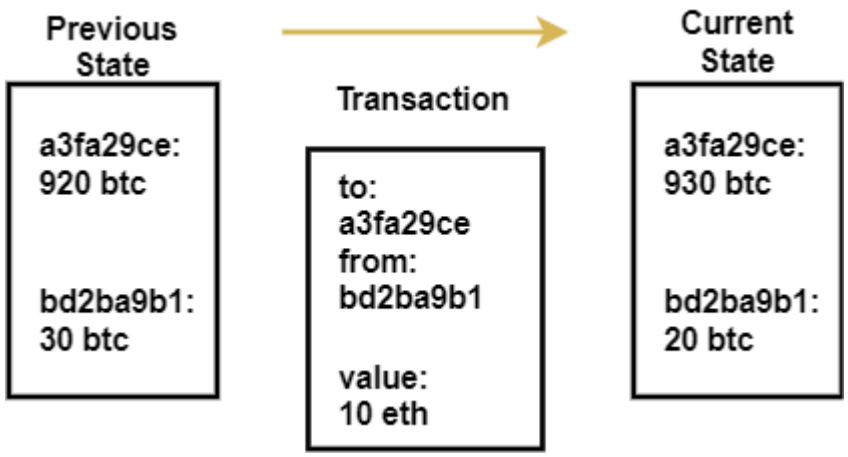

Figure 20: Illustration of the Ethereum state machine

4.1.7 Ganache

Ganache is a blockchain application developed by the Truffle Blockchain Group [25]. It provides an in-memory blockchain and creates a testing environment to allow the experimentation of transactions and smart contracts. Experimenting with smart contracts and transactions are necessary to reduce unwanted smart contract transaction interactions that could possibly lead to the loss of data with transactions not happening in the intended way. In addition, experimentation can occur without needing to use real money. The application provides a few dummy accounts with 100 Ethereum in each one. Any interaction with each account is recorded and can be shown on the interface. Figure 21 is a sample picture of what the interface looks like and how the addresses of externally owned accounts look like and how the balance is shown for each account.

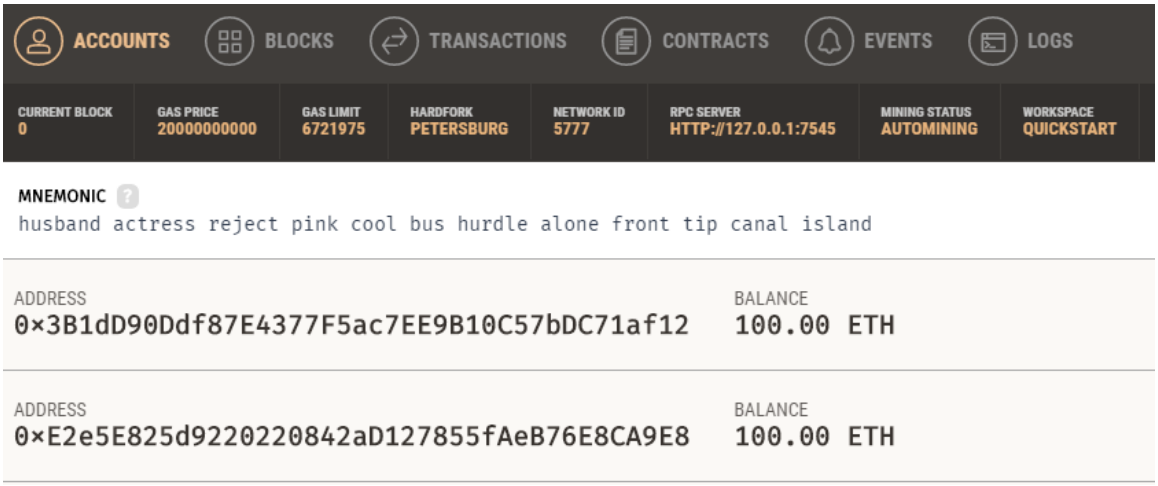

Figure 21: Ganache application interface.

4.1.8 Remix

Remix is a browser-based IDE that allows for smart contracts to be created and deployed. Since the language Solidity has a very similar structure as C, it was not hard to work with and write a smart contract from scratch. The interface is simple and allows the user to easily find any details about the contract needed to interact with it such as its address and application binary interface or ABI. Remix also allows us to connect to our Ganache blockchain such that the smart contract can be deployed onto the Ganache blockchain and allow for the interaction of the smart contract. The smart contract we developed is simple. The contract contains 4 variables: sender address, username, rating, and version. We created a function capable of setting the username, rating, and version number of the rating that the web application can use when it wants to store information onto the blockchain. The contract also contains 3 separate functions to get the block's respective username, rating, and rating version number. These functions are used only if a

username match is found in the transaction. Figures 22 and 23 display the interface of the Remix IDE.

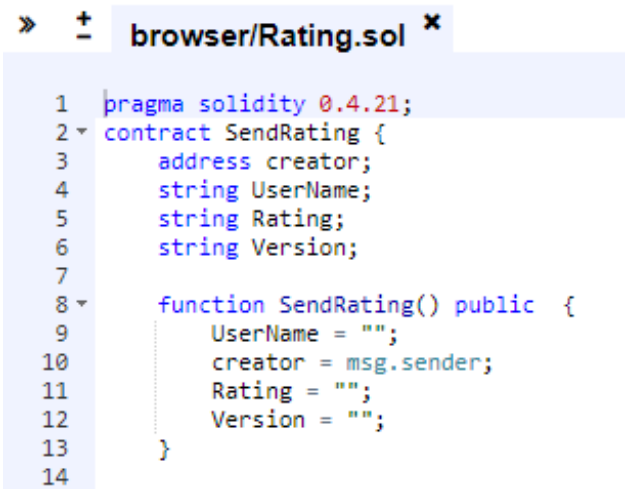

Figure 22: Remix interface where the smart contract code goes.

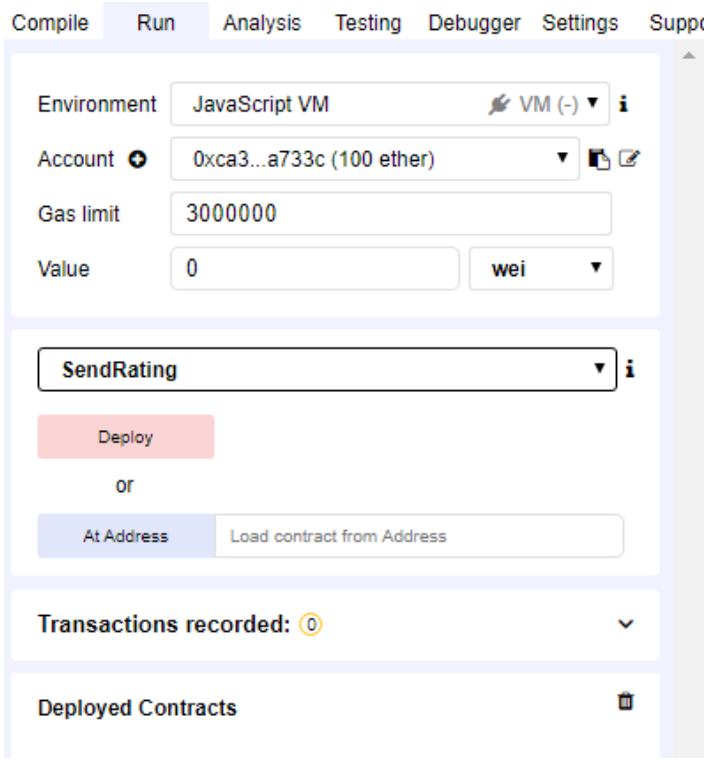

Figure 23: Remix interface where a user deploys the contract.

4.1.9 Web3

The web3 library [26] will connect the Ethereum blockchain with a flask web application. The library has a multitude of functions that allow the user to interact with the blockchain such as using the smart contract to look at information stored in the blocks. With this library, the AI system can store the GitHub profile names and their associated ratings and version number and the web application can pull information from the blockchain and display for the user.

4.2 Blockchain System Design

Designing the system was simple. The web application prompts the user for a GitHub profile username. Blocks from the blockchain are then pulled from the blockchain through the smart contract. This is done so that the system can check what username is stored in each of the transactions stored in the block. If a match is not found, then the AI system needs to calculate a rating for the username. The results of the AI system are then displayed onto the web page and stored in the blockchain. If a match is found, all relevant data are pulled and saved into the web application. The version number of the rating is then checked against the current version. If they match, then the username, rating, and version number are displayed on the web page. If the versions do not match, then the rating for the username is recalculated by the AI system again. Figure 24 shows how the data should flow through the application.

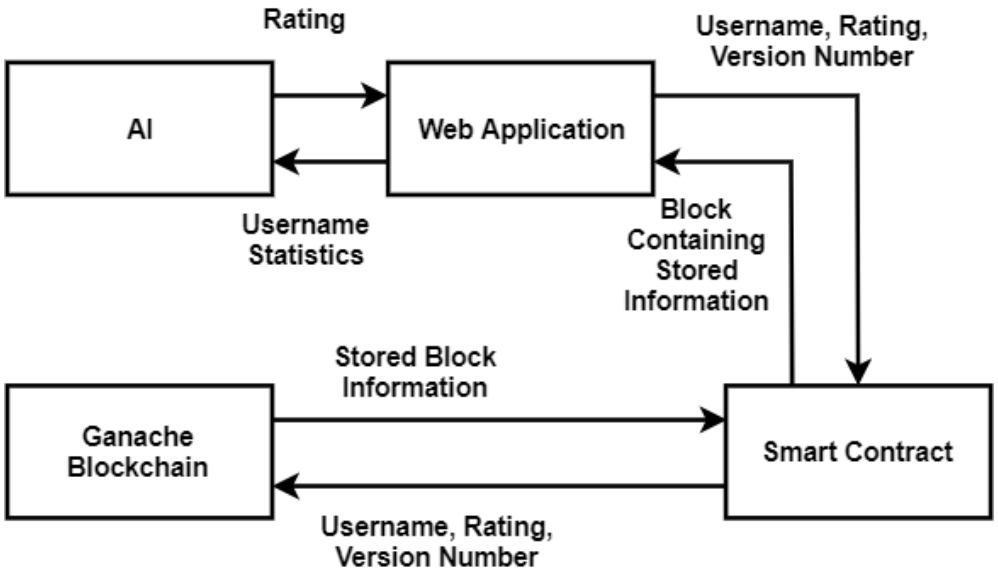

Figure 24: Data flow in blockchain application.

4.2.1 Rating Lookup

This functionality is called when a username is given to the web application. The application needs to know if the username has had a rating created before. The application loops through the blockchain block by block and searches each transaction stored on each block. The information stored in each transaction needs to be decoded since we stored extra information in each transaction. In each transaction, the application pulls the username using the smart contract function to match it with the given username to prove whether a rating has been created for the profile before. If a match has been found, the rating and version number is pulled from the transaction using the other 2 separate functions from the smart contract. The version number is then compared to determine if the rating is out of date and needs to be updated. If the rating does not need to be updated, then the username, rating, and version number are displayed onto the web page. If rating does need to be updated, then the application will use the GitHub API to

search for the statistics of the profile so the AI system can create a rating for the profile again.

4.2.2 Rating Creation

Ratings are created by the AI. The rating creation process usually begins due to the GitHub username not being found on the blockchain. The job of the AI system is to pull data from GitHub using the GitHub API, calculate the profile's score, store the score on the blockchain, and display it for the user. The web application starts off by prompting the user for a GitHub username. The web application searches for the username on the blockchain and tries to match the given username. If there are no matches, the application uses the GitHub API to search for the username's statistics. If the username was not found or the profile was found to be private, the web application would throw an error message stating the problem.

After the AI system calculates the rating, it needs to be sent to the blockchain for storage and displayed on the web page for the user. The username, rating, and version number are all sent to be stored on the chain. The smart contract function is invoked to set the username, rating, and version number. A transaction is then sent onto the blockchain.

The other reason for creating a rating is because the rating associated with the username is out of date. This is determined by checking the version number stored with the rating. It is matched up against the current version number tracked by the web application. When the version number does not match, it means that the AI system was dynamically trained, and all previous ratings are out of date. A new rating needs to be

created by the AI system again as it has been tuned to the new data the system has received.

4.3 Results

An analysis of our results will show whether the system was successfully developed. The AI and the blockchain portions need to be integrated successfully to build our rating system. We must check that the communication between the AI and blockchain are successful and that the rating system can be interacted with through the web application. To accomplish this, the information obtained by the AI and the information stored in the blockchain must match each other. Figure 25 is the detailed view of how information is passed in the system. We must also look to how to improve the system as time goes on as well. A developer's skill is always fluctuating. Our rating system must reflect that. Allowing the AI to have some sort of dynamic training to create new versions of past ratings will allow the system to accurately judge the developer based on the average of the ratings it has created.

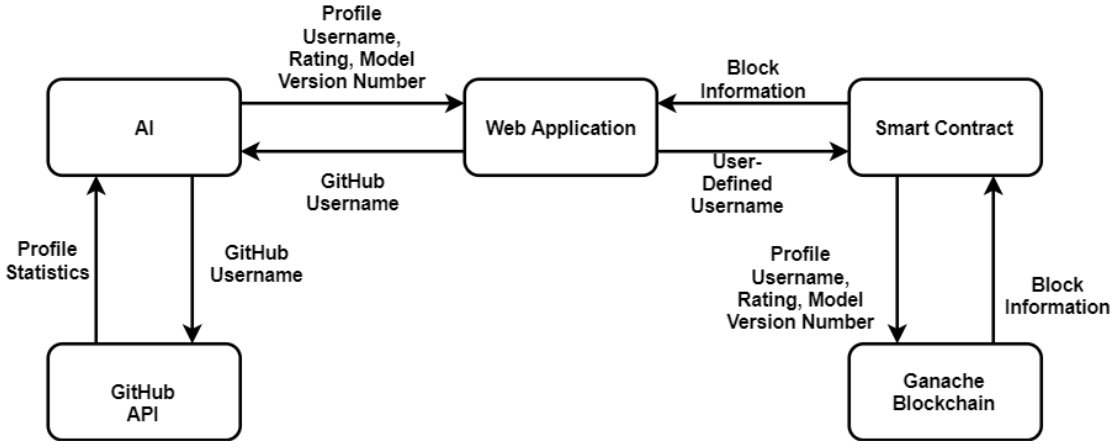

Figure 25: System design illustration.

4.3.1 System Integration Results

Creating our blockchain web application meant integrating the AI system and the blockchain and ensuring that the communication between the two entities was successful. The rating from the AI, the username given by the user, and the AI's version number tracked by the web application needs to be passed from the application to the smart contract and then to the blockchain to be securely stored. The web application must also be able to pull information from the blockchain and be able to parse through the block of information and find the matching username if it was stored there. The web application must also be able to trigger the dynamic training of the AI system model with the new influx of data it receives when searching and creating new ratings for usernames. Figure 26 displays the web page after a rating has been created. As you can see, the user "ftrias" was entered into the search bar above the messages and the web application displayed the user's rating and their associated statistics from which the rating was calculated from. Figure 27 shows the various stages the application goes through during the rating creation. Figure 28 then shows how the rating information linked to the user was stored.

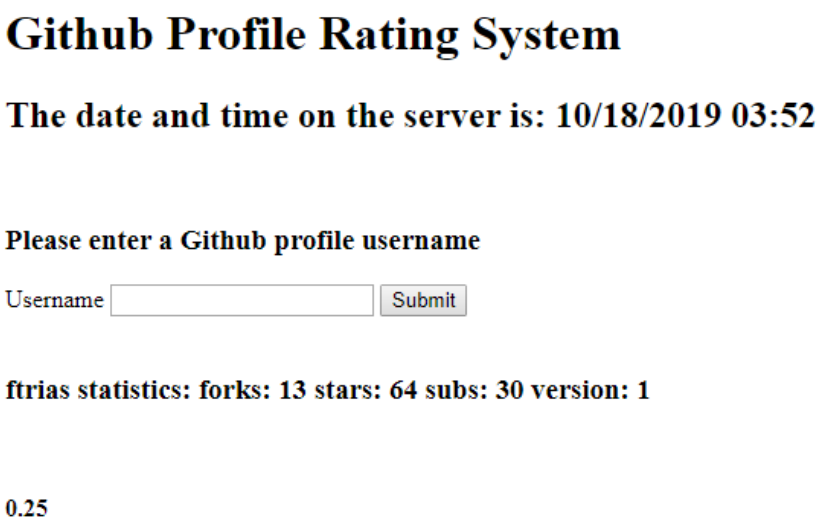

Figure 26: Resulting web page after creation of rating for GitHub profile.

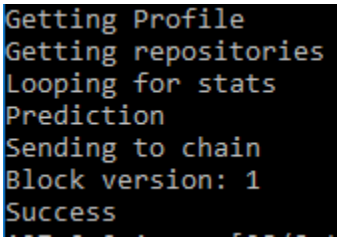

Figure 27: Messages showing which stage the program is at.

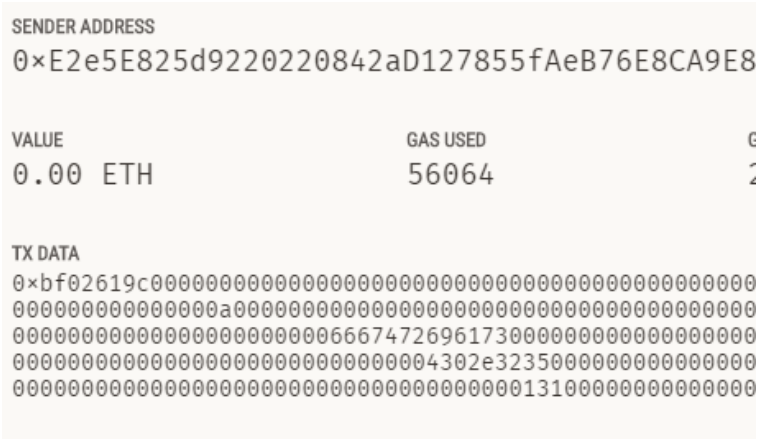

Figure 28: Data stored on the blockchain.

Looking at Figure 28, if we convert the hex value 0x667472696173 to the American Standard Code for Information Interchange (ASCII), it produces the string "ftrias". This username was seen being input previously in Figure 26. Converting the hex value 0x302e3235 to ASCII will produce the string 0.25. This is the same rating that was shown in Figure 26 as well. To show that the system can pull data from the blockchain, another rating is created and displayed onto the web application as shown in Figure 29. Finally, Figure 30 shows the web application displaying the information from the previously created rating from Figure 26.

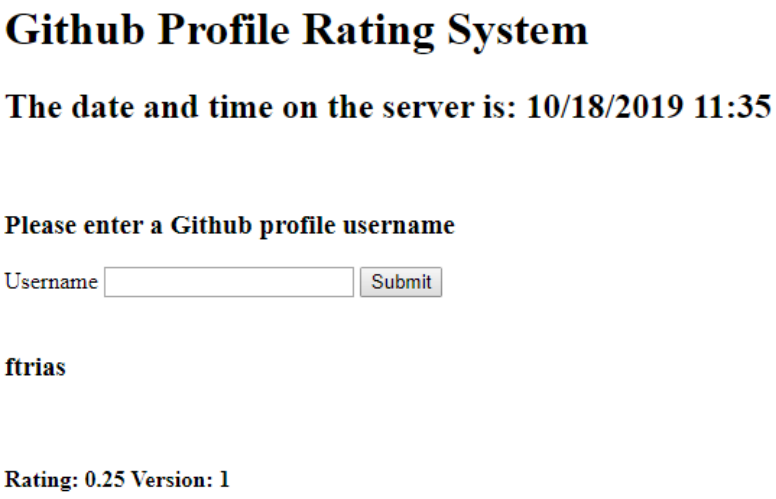

Figure 29: Creating another rating for a profile.

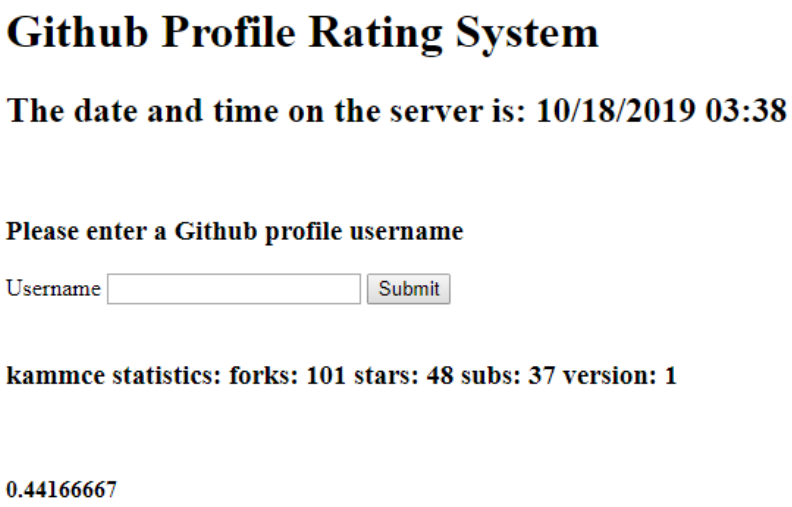

Figure 30: Web page displaying a profile with a previously created rating.

4.3.2 Dynamic Training

The final part of the application is the dynamic training of the model. Whenever the application is given a username that is not found on the blockchain, the application searches GitHub for the user and its statistics. As it loads the statistics into the AI system to create a rating, the application saves the statistics into an array. This array is going to be used as a new dataset. This is done so the AI system can use this new information to tune itself to new information that is up to date. When the array becomes full of new statistics, it is time to re-tune the AI system to the up-to-date information. To make sure that all ratings are kept up to date, the application keeps track of the version number of the AI. This version number helps the application keep track of when each rating was created. This is accomplished by adding in a version number parameter into the smart contract code as discussed previously. The application keeps track of the version number and when there are enough new usernames that have been searched for, the model tunes itself to adapt to the new information.

The data that was saved in the array is fed into the k-means clustering algorithm to re-cluster all the new data. Based on the new clusters, each data point is given a new base rating. The web application will save all the new usernames and ratings into a CSV file. This is done so that the server keeps a tab on all data that passes through the application and helps to identify when the system may be failing. The file is then fed to the model and fitted to the new data for tuning. The application will also output a message stating that the model was updated. The process the system goes through for dynamic training can be seen in Figure 31. The results of the training can be seen in Figure 32. After the dynamic training has been completed, any profiles previously searched will need to have their ratings recalculated since the model was updated. In Figure 33, you can see how the rating for user "ftrias" changed from Figure 26. The reason we see the rating go down is most likely due to its stats not being as powerful as before. Powerful profiles being searched for in the previous version can have this effect on older data as these powerful profiles will increase the average of each statistic and lower the power a single point has. The updated rating and its version number are shown in a new block in Figure 34. The hex value of 32 translates to 2 in ASCII.

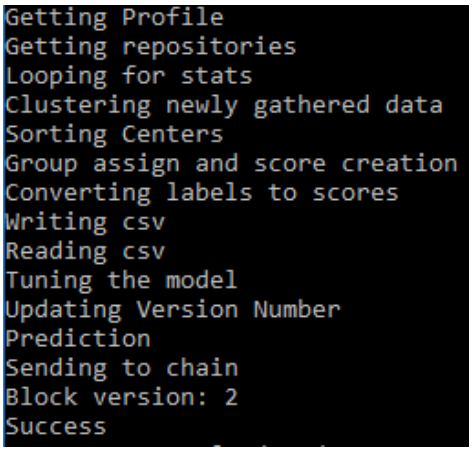

Figure 31: Console messages depicting the dynamic training stages.

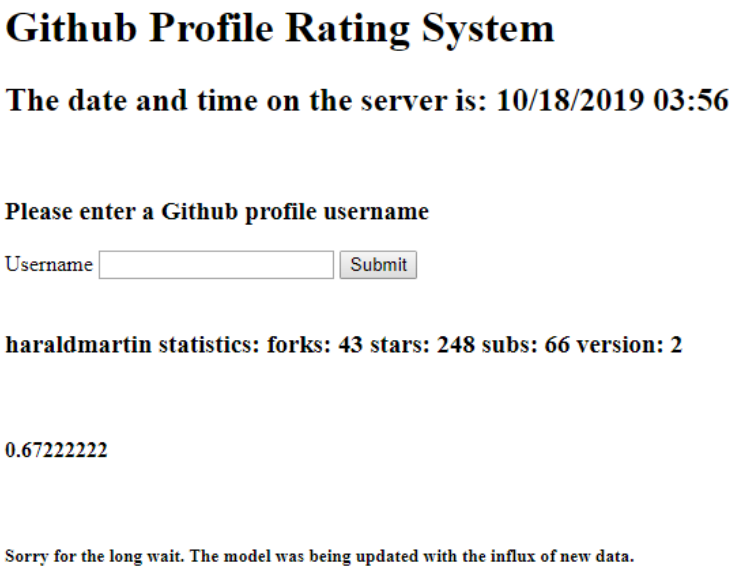

Figure 32: Application informing user of the dynamic training.

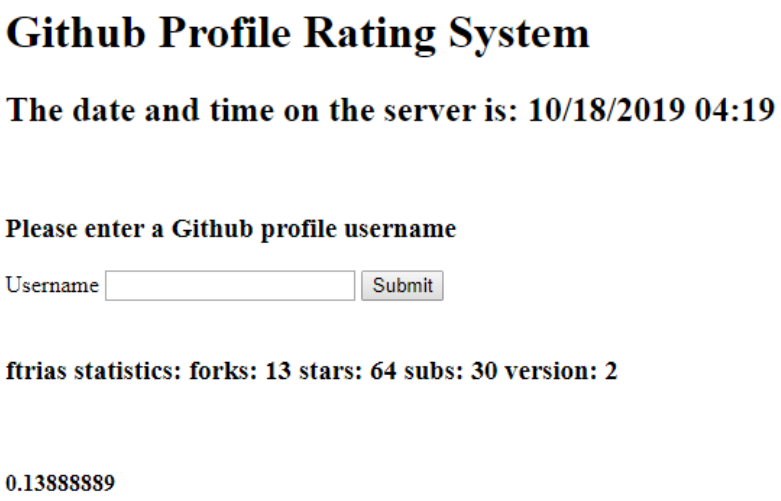

Figure 33: User ftrias with a new rating due to dynamic training.

Figure 34: User ftrias with a new rating stored onto the blockchain.

5 FUTURE WORK

Additional work that can be done to improve the work reported here. One improvement would be how the AI system interacts with a user's information. If the system were to include an analysis of the code uploaded by the GitHub profile and possibly the code uploaded on other websites, it would give a better picture of who the user is and what he or she can do. Profile statistics only tell part of the story and being able to look at a repository's code and analyze it will help tell the complete story of the user. When the AI system is able to do more than judge statistics, the ratings of users will become much more accurate and will lead to an increase in users trusting the ratings given by application and leads to the possibility of more frequent use of the application.

Another area of improvement of the web application is to offer the chance to calculate the rating of a specific repository. In many situations, users will be looking at individual repositories as opposed to the individual contributors. Performing more research into the GitHub API will determine if this is feasible. This feature would make the application more attractive to users as it allows the application to be used in more situations.

Another area of improvement is the use of a different blockchain platform that allows for better customization. The customization of the blockchain platform would allow for better optimization of the application. Since some of the features in our implementation are not being used, time will be wasted to work around them. For example, the system does not make any use of the Ether currency, but we are forced to use it because it is part of the blockchain platform. In addition, we want to be able to track who has access to the system as we do not want to allow anyone but the AI system to write to the blockchain.

When users can trust that the information being written onto the blockchain is done only by the AI system and do not have to worry about the currency aspect associated with many blockchain platforms, users are more likely to use the application. Ethereum is a good platform to demonstrate how data could be stored onto the blockchain. Unfortunately, it costs money to store information on the chain which would make the application unfeasible in the future due to the costs.

Finally, improving the AI will be done. From the beginning of this project, knowledge on machine learning was primitive. With the prospect of feeding additional information to the AI such as the analysis of development code, the algorithm needs improvements. Furthermore, the dynamic training algorithm will be improved as it needs to keep pace with the results given by the AI.

6 CONCLUSION

In this thesis, we introduced the idea of combining the technologies of an AI system and blockchain to create a rating system to tackle the problem of being able to trust open source code bases. We used GitHub to help provide the basic information on their profiles. Due to the nature of creating our own dataset, we also had to create our own base ratings so that the machine learning algorithms could learn how we wanted profiles to be rated. Using the k-means clustering algorithm, we grouped similar profiles together and allowed us to label groups a certain rating with minimal bias. After solving the problem of not having any base ratings for profiles, we could now work on the AI system to automatically create these ratings. We were able to narrow our algorithms to random forest and neural networks. Both gave similar outputs, however, in its current state we recommend using the random forest implementation as it reduces the amount of overfitting we found in the usage of the neural network due to how simple the dataset is. Even so, when the introduction of judging actual code comes into play, we will consider using neural network if the dataset starts to get complicated. Finally, the rating information was successfully stored onto the Ethereum blockchain and can be accessed using the web application and displayed for the user.

LITERATURE CITED

[1] Pittenger, M. (2019). Why the Equifax breach should never have happened | TechBeacon. [online] TechBeacon. Available at: https://techbeacon.com/security/why-equifax-breach-should-never-have-happened [Accessed 19 Oct. 2019].

[2] J. Schnebly and S. Sengupta, "Random Forest Twitter Bot Classifier," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2019, pp. 0506-0512.

[3] H. Valecha, A. Varma, I. Khare, A. Sachdeva and M. Goyal, "Prediction of Consumer Behaviour using Random Forest Algorithm," 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gorakhpur, 2018, pp. 1-6.

[4] J. Hu, "Personal credit rating using Artificial Intelligence technology for the National Student Loans," 2009 4th International Conference on Computer Science & Education, Nanning, 2009, pp. 103-106.

[5] L. Chen, J. Zhou, L. He, Q. Chen, J. Zhang, and Y. Yang, "Modeling User-Item Profiles with Neural Networks for Rating Prediction," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, 2017, pp. 301-308.

[6] H. Wang, D. Yang, N. Duan, Y. Guo and L. Zhang, "Medusa: Blockchain Powered Log Storage System," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 518-521.

[7] M. Kumar, A. K. Singh, and T. V. Suresh Kumar, "Secure Log Storage Using Blockchain and Cloud Infrastructure," 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bangalore, 2018, pp. 1-4.

[8] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han and F. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 11, pp. 2266-2277, Nov. 2019.

[9] B. Gao, Q. Zhou, S. Li and X. Liu, "A Real Time Stare in Market Strategy for Supply Chain Financing Pledge Risk Management," 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, 2018, pp. 1116-1119.

[10] Y. Xu, M. Wu, Y. Lv and S. Zhai, "Research on application of block chain in distributed energy transaction," 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, 2017, pp. 957-960.

[11] K. Salah, M. H. U. Rehman, N. Nizamuddin and A. Al-Fuqaha, "Blockchain for AI: Review and Open Research Challenges," in IEEE Access, vol. 7, pp. 10127-10149, 2019.

[12] Y. Wehbe, M. A. Zaabi and D. Svetinovic, "Blockchain AI Framework for Healthcare Records Management: Constrained Goal Model," 2018 26th Telecommunications Forum (TELFOR), Belgrade, 2018, pp. 420-425.

[13] GitHub. (2019). REST API V3. [online]. Available at: https://developer.github.com/v3. [Accessed 21 June 2019]

[14] V. Jacques. (2019). PyGithub. [online] ReadtheDocs. Available at: https://pygithub.readthedocs.io/en/latest/index.html [Accessed 25 June 2019].

[15] W. McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56, (2010).

[16] Piech, C. (2019). K-Means. [online] Stanford.edu. Available at: https://stanford.edu/~cpiech/cs221/handouts/kmeans.html [Accessed 26 Jun. 2019].

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830, (2011)

[18] W. Koehrsen. (2018). An Implementation and Explanation of the Random Forest in Python. [online] Available at https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76 [Accessed 1 July 2019]

[19] Keras.io. (2019). Home - Keras Documentation. [online] Available at: https://keras.io/ [Accessed 21 Jul. 2019]

[20] J. Hunter. Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95, (2007)

[21] A. Rosic. (2019). What is Ethereum? [The Most Comprehensive Step-by-Step-Guide!]. [online] Blockgeeks. Available at: https://blockgeeks.com/guides/ethereum/ [Accessed 19 Oct. 2019].

[22] A. Hertig. (2019). How Ethereum Works - CoinDesk. [online] CoinDesk. Available at: https://www.coindesk.com/information/how-ethereum-works [Accessed 19 Oct. 2019].

[23] P. Kasireddy. (2019). How does Ethereum work anyway? [online] Medium. Available at: https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369 [Accessed 19 Oct. 2019].

[24] The Economic Times. (2017). What is the difference between bitcoin and ethereum?. [online] Available at: https://economictimes.indiatimes.com/markets/stocks/news/what-is-the-difference-between-bitcoin-and-ethereum/articleshow/62171361.cms [Accessed 19 Oct. 2019].

[25] Truffle Suite. (2017). Ganache Truffle Suite. [online] Available at: https://www.trufflesuite.com/ganache [Accessed 30 Aug. 2019].

[26] P.Merriam, J. Carver. Web3.py(2018). Web3.py 5.2.2 documentation. [Online]. Available: https://web3py.readthedocs.io/en/stable/index.html.[Accessed 30 Aug. 2019].