HOW DO MACHINE LEARNING ALGORITHMS WORK?

Artun SARIOĞLU artunsarioglu@sabanciuniv.edu
Computer Science, 2

Evrim ÖZEL evrozel@gmail.com
Computer Science, 4

Onur Yigit ARPALI arpali.onur@metu.edu.tr
Computer Science, 4

Yusuf Umut ÇİFTÇİ uumutciftci@gmail.com
Electrical and Electronics Engineering, 2

Ziya ERKOÇ tellobjkziya@gmail.com
Computer Science, 2

Selim BALCISOY
Computer Science

Abstract

We first became familiar with the concept and the implementation of decision trees, clustering and artificial neural network algorithms of machine learning. Then the visualization process of all three algorithms started. The Bokeh library is used for the visualization process of the algorithms. Interactive educative websites in Turkish have been developed by the group via d3.js. Later on, an experiment was conducted on non-professionals in order to control our progress and calculate our ability to transfer our knowledge. We have also contributed to open source with our visualization in Bokeh library.
Keywords: visualization, machine learning, decision tree, k-means, neural networks, bokeh.

1 Introduction

Our research problem was to discover intuitive visualization techniques to explain a small set of popular machine learning algorithms to ordinary people, particularly high school students and firms who want to use these algorithms but lack the knowledge of how to use them. This meant that we needed to prepare our documentation in Turkish so that our audience would be able to learn the material easier.

During our literature search, even though we found several visualization examples of advanced machine learning topics, the elementary and introductory concepts were very lacking in visualization. We only found websites that explained the material in by text, with few visuals. So we wanted to create a web app that focused on visuals first and explained the concepts later with minimal text usage. The usage of Turkish was very important too because the websites that we found were only in English, and hardly any Turkish machine learning introductions with visuals exist.

We wanted to cover different categories of ML concepts, namely decision trees, clustering algorithms, and artificial neural networks. We found that on the internet, decision trees especially had almost no examples of this kind of visualization, but later discovered that research on tree visualization techniques (i.e. algorithms for node/edge positioning) had been done in the past (Buchheim, Junger & Leipert, 2002). So we decided to use these techniques in the context of decision trees and web visualization. The Bokeh library was used to visualize the tree structure since it allowed us to use embedded Python in our web app, which meant that implementing a pseudo-code algorithm was much intuitive. We used the ID3 algorithm (Quinlan, 1985) to implement the decision tree.

For clustering, we found more examples compared to decision trees but they weren't oriented towards ordinary people like we intended.

Finally, for artificial neural networks, we found that material for this subject was a bit more extensive, with the most notable example being Tensorflow's playground example (https://playground.tensorflow.org). We couldn't find any Turkish resource for this topic, though, and we wanted to continue working with Bokeh library (which also didn't have any ANN related examples), so we decided to make a bokeh module for this as well. We used Tensorflow's Python library to implement the ANN algorithm.

We used the D3.js library for visualization when Bokeh wasn't enough for interaction with the user on browsers since D3.js is a javascript framework which is more flexible than Bokeh on the client side.

2 Methodology

2.1 Decision Tree

2.1.1 Visualization

After searching for visualization examples for decision trees, we couldn't find any example to visualize decision trees so far. We started to explore different libraries to suit our purpose, and we decided on Bokeh library. First, it's a Python library so we can code easily and avoid using javascript, which tends to be slower and not clear to debug. We used python on the server side, thus we avoided any performance issues since Python is more suitable for ML implementations. It has many libraries such as Numpy and Pandas to make matrix operations and data pre-processing easier.

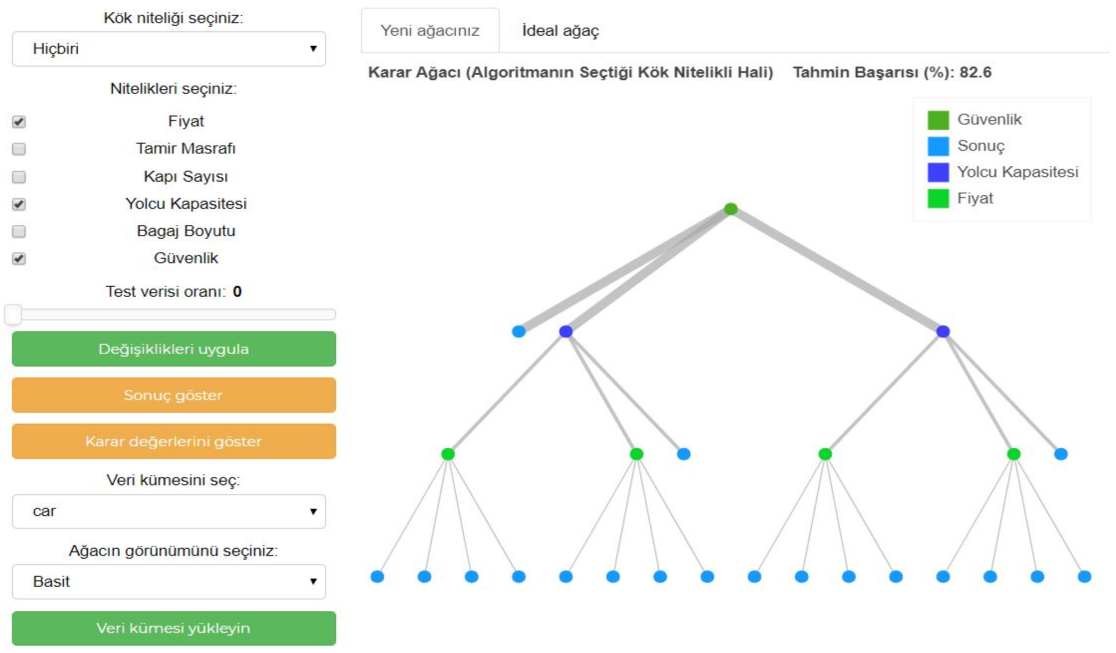
Bokeh's website contained many different examples which are complex but creates basic figures so that visualization is less constrained. A particular example we found was a periodic table example, which appealed to us because it had a nice grid layout. Normally, these library is used for regular plots like histograms, line plots, bar plots etc. but this example uses the layout freely without depending on a single plot, and it used rectangle-shaped glyphs and text which we structured our decision tree application based on this example.

After figuring out this example, we started to search for Bokeh data structures. Bokeh has its own data management objects which organize the data efficiently, eases the use of different utilities like tooltips for accessing data on the plot, it also updates the glyph data automatically so the programmer doesn't have to re-draw the glyphs with each data modification and provides increased performance compared to regular data objects.

We also focused on the decision tree algorithm itself. We decided to use ID3 decision trees which classifies mostly nominal data sets since these data sets tend to be more understandable to non-professionals. To give an example, we initially used a dataset that classifies cars according to seats, luggage area, safety index, price etc. (Dua & Taniskidou, 2017). Depending on these attributes, this dataset classifies the cars as acceptable or unacceptable.

We first implemented the ID3 decision tree with 3 statistical methods which are information gain, gain ratio, Gini index to generate the tree. Basically, these methods decide which attribute should be used to divide the data set according to its attribute values. Information gain and gain ratio rely on the entropy concept. Entropy is a measurement of the complexity of the dataset's distribution at certain points. But entropy is not enough on its own to split entire datasets. We need to compute the weighted average over all sets resulting from the split. At this point, we get "information" which is a different type of measurement. Subtracting the "information" from the entropy of the data set, we get information gain that we can use to generate all the tree. The gain ratio is calculated by dividing the information gain by intrinsic information, which is the entropy of the distribution of instances into branches. This gives us how much information we need to determine the branch of an instance. As we understand, gain ratio uses more information than the information gain so it has better performance to split the dataset into branches. In addition to these two statistical methods, we implemented a third statistical method which is Gini index. Gini index, basically, divides the instances according to its purities. The purity is an index to split the instances very similar to entropy. In the end, all these methods look so complicated to non-professionals. So we decided to just use Gini index in the decision tree.

The ID3 decision tree which we describe above returns a data structure which describes the relation between nodes, Gini index values of every node and the leaf labels. As we mentioned in the first paragraph of this section, we use Bokeh layouts with basic glyphs. So we need to position these nodes on the layout to visualize the tree nicely. Positioning issues cropped up at this point. First, we tried to create our own positioning algorithm, but decision trees can differ a lot, so we needed to create an adaptable algorithm to cover all trees. Our algorithm worked fine actually: there was no collision between branches or nodes, but it didn't look very good. Because we couldn’t calculate the proportions between nodes, we made a literature search about this issue. We found different positioning algorithms. The first algorithm we found was the Reingold-Tilford drawing algorithm (Reingold & Tilford, 1981). This algorithm is considered the basis of the drawing algorithm that we found and used. After the creation of this algorithm, John Walker improved this algorithm and created his own positioning algorithm for general trees (Walker, 1990). After searching for more literature, we finally found a brand-new positioning algorithm (Buchheim et al. 2002). It draws the rooted tree in linear time so it has a much better performance and it can work better with our server, which we will mention in the next section.


Kök niteliği seçiniz: Hiçbiri

Nitelikleri seçiniz:

- [x] Fiyat
- [ ] Tamir Masrafı
- [ ] Kapı Sayısı
- [x] Yolcu Kapasitesi
- [ ] Bagaj Boyutu
- [x] Güvenlik

Test verisi oranı: **0**

Veri kümesini seç: car

Ağacın görünümünü seçiniz: Basit

Karar Ağacı (Algoritmanın Seçtiği Kök Nitelikli Hali) **Tahmin Başarısı (%): 82.6**

Figure 1: Decision tree visualization with Bokeh

Continuing with the tree visualization, we added several tools to the tree application to make the visualization more intuitive. The user can add or remove different attributes from the tree to make the tree more detailed or simple, or they can change the root node to create a different tree and see the difference in accuracy or Gini values of the nodes depending on the changes. The user can compare the newly created tree and the ideal tree that the decision tree algorithm computes and see how changing properties of the tree changes accuracy. The ability to add new datasets to the web app was added, so users can use the web app to visualize different datasets. The layout of the widgets and plot was designed for adaptability so it works the same with different computers with different screen sizes. The user can choose to show or hide edge and leaf labels depending on if they want an uncluttered view. After finishing this example, we polished the visuals and decided to send this part of the project as a contribution to Bokeh’s examples.

2.1.2 Interactive and Educative Web-Apps

Apart from visualizing the decision tree algorithm, we created interactive web-pages helping people to learn core concepts of decision tree including entropy and classification. We used interesting examples to explain how decision tree actually works.

Firstly, we used the melon and watermelon analogy to explain the logic of classification. In the watermelon-melon example, we listed the watermelon and melons based on their radii and put the ones with the same size in the same column. The objective here is to find a radius that best separates watermelon and melon. In the figure below, the split point is set to 11 cm. Model creation process in Machine Learning is called training which resembles the learning process of a human.

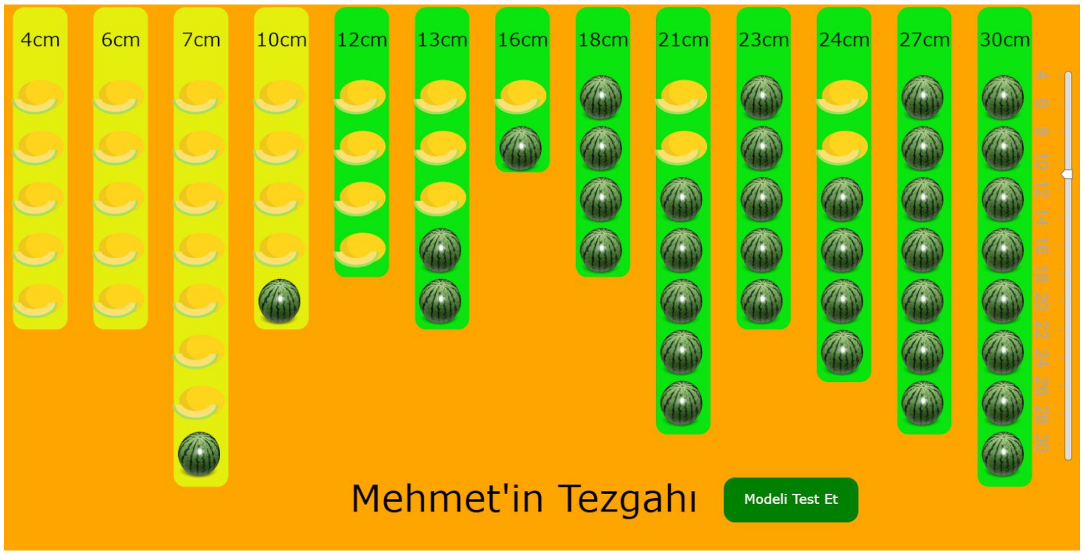


Figure 2: Training our model

In that setting, our model will classify fruits having a radius greater than 11 cm as watermelon, otherwise; melon. As watermelon and melon can be similar in size, there might be some error or wrong-classification in the model. In Machine Learning, one of the measurements to explain the model performance is called accuracy. Accuracy is basically the ratio of correct guesses divided by the total number of guesses. Having completed the training phase, test phase begins. In the test phase, model encounters with different data and tries to apply its assumption here. In the below figure, the model just created which determines split point as 11 cm, is tested. The accuracy of this case happens to be 94%. Eventually, with this example, we aimed to explain how classification works which is the root of the decision tree algorithm.

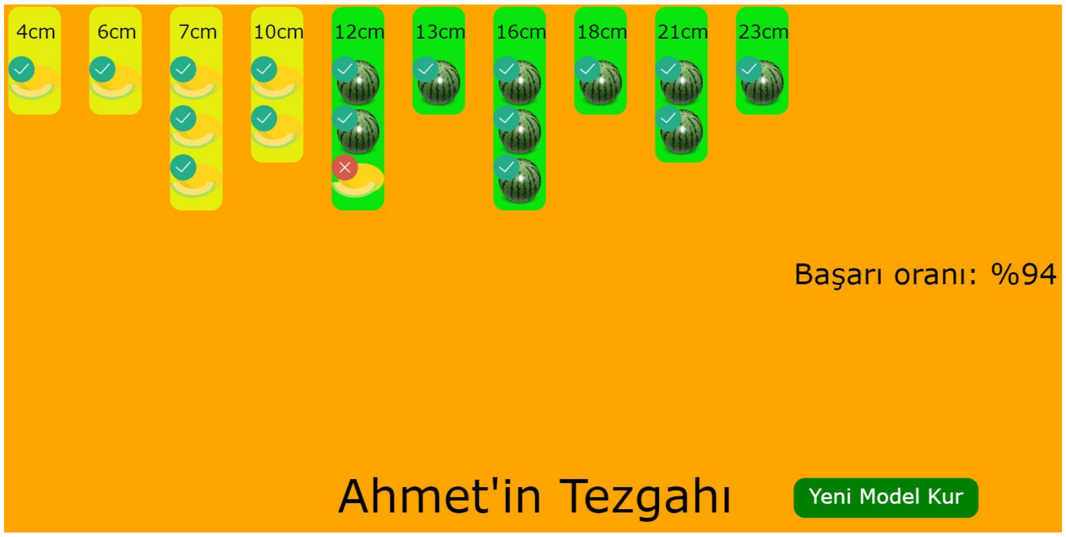
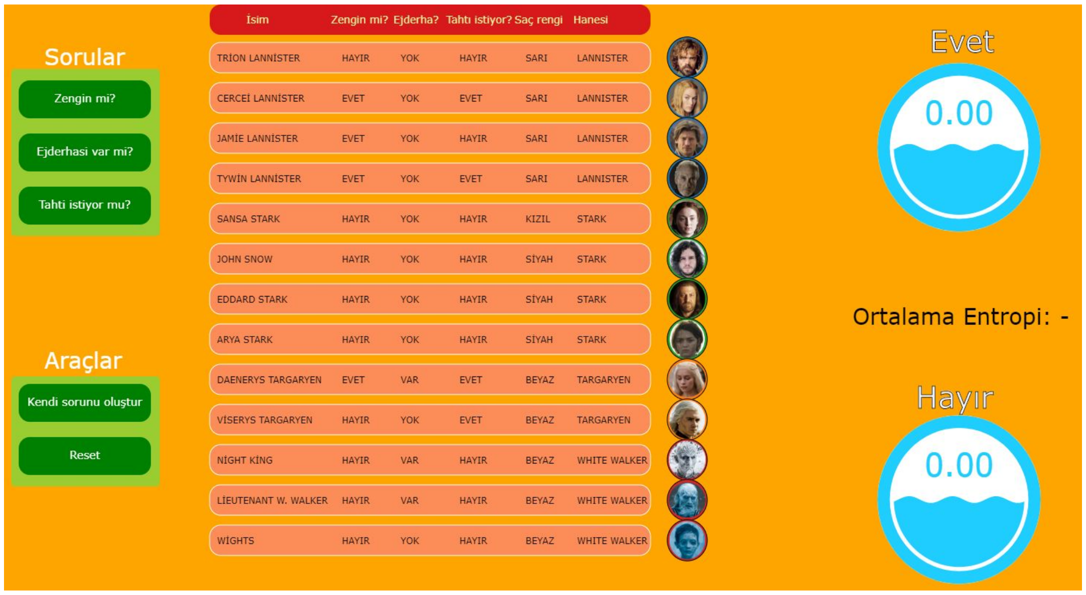


Figure 3: Testing the model

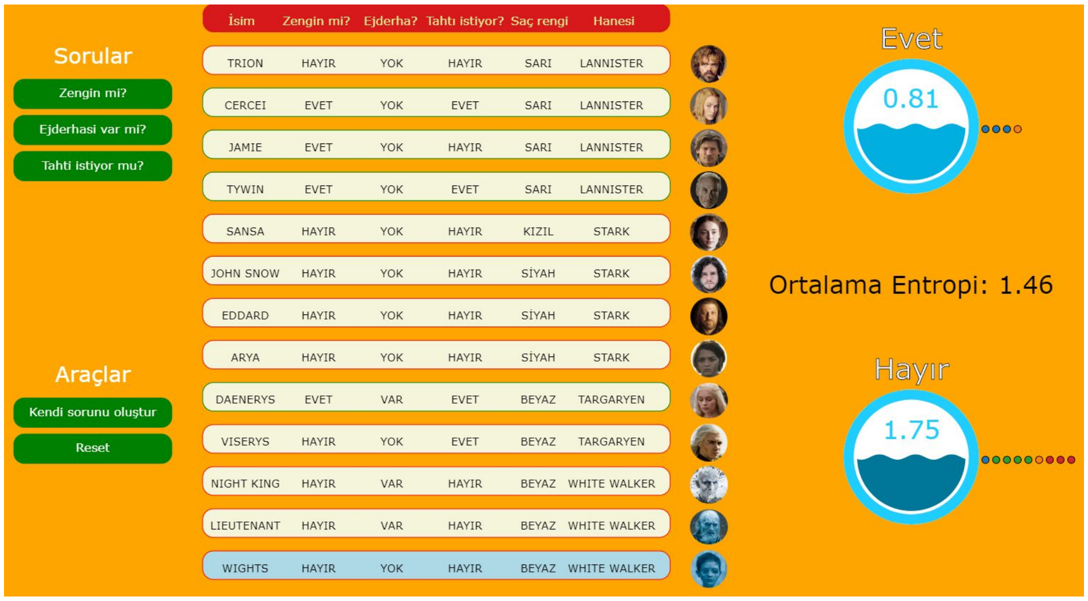
In our second example, we explain the concept of entropy. Entropy is basically the measurement of impurity. In other words, it measures how far our data is from being homogeneous. To draw attention, we used Game of Thrones characters. Each row represents a character in the series and each column represents one of the features of the characters. These features include ‘wealth’, ‘dragon ownership’, ‘demand for the throne’, ‘hair color’. The first column is the name of the character and as it is unique it is not a feature. The last column represents the house where a character is from. Each house has its own color assigned by us and can be seen as a border around the image of characters. The main problem emphasized in this example is to predict the house of the character by using his/her features. This problem is not solved in this example but how the decision tree algorithm approaches this problem is explained. The main task in this example is to separate characters into yes and no pools with respect to a feature by asking a question. The characters then will go to the respective pool based on his/her wealth. Measurement of the impurity as mentioned before is called entropy. In our example, we describe the entropy as the darkness of the water. As each color represents a house, if, for example, one pool contains characters from each house this pool would be very impure, therefore; entropy would be quite high. However, if all characters from the same house were to gather in a pool, this would mean low impurity, thus; high entropy. Therefore, Question selection is important, as different questions may affect the distribution of the characters. One possible question would be “Is the character rich?”. In this example, there are two ways to do the separation. One can select previously one of the determined questions which are located on the left of the page. After the selection, separation process then will visually and automatically continue. One can also create his/her own question. Having determined a general question in his/her mind, for each character one can click yes and no buttons above the pools to separate characters into two pools.



İsim	Zengin mi?	Ejderha?	Tahtı istiyor?	Saç rengi	Hanesi
TRION LANNISTER	HAYIR	YOK	HAYIR	SARI	LANNISTER
CERCEİ LANNISTER	EVET	YOK	EVET	SARI	LANNISTER
JAMİE LANNISTER	EVET	YOK	HAYIR	SARI	LANNISTER
TYWIN LANNISTER	EVET	YOK	EVET	SARI	LANNISTER
SANSA STARK	HAYIR	YOK	HAYIR	KIZIL	STARK
JOHN SNOW	HAYIR	YOK	HAYIR	SİYAH	STARK
EDDARD STARK	HAYIR	YOK	HAYIR	SİYAH	STARK
ARYA STARK	HAYIR	YOK	HAYIR	SİYAH	STARK
DAENERYS TARGARYEN	EVET	VAR	EVET	BEYAZ	TARGARYEN
VİSERYS TARGARYEN	HAYIR	YOK	EVET	BEYAZ	TARGARYEN
NIGHT KİNG	HAYIR	VAR	HAYIR	BEYAZ	WHITE WALKER
LIEUTENANT W. WALKER	HAYIR	VAR	HAYIR	BEYAZ	WHITE WALKER
WİGHTS	HAYIR	YOK	HAYIR	BEYAZ	WHITE WALKER

Figure 4: Game of Thrones example overview

An example separation could be seen in the below figure in which characters are separated after asking the question of “Is the character rich?”. This created 0.81 and 1.75 entropy respectively for yes and no pools. The weighted average entropy is 1.46. It is calculated by $\frac{0.81 * 4 + 1.75 * 10}{14}$ where 4 and 10 are the number of characters in the yes and no pools respectively.



İsim	Zengin mi?	Ejderha?	Tahtı istiyor?	Saç rengi	Hanesi
TRION	HAYIR	YOK	HAYIR	SARI	LANNISTER
CERCEI	EVET	YOK	EVET	SARI	LANNISTER
JAMIE	EVET	YOK	HAYIR	SARI	LANNISTER
TYWIN	EVET	YOK	EVET	SARI	LANNISTER
SANSA	HAYIR	YOK	HAYIR	KIZIL	STARK
JOHN SNOW	HAYIR	YOK	HAYIR	SİYAH	STARK
EDDARD	HAYIR	YOK	HAYIR	SİYAH	STARK
ARYA	HAYIR	YOK	HAYIR	SİYAH	STARK
DAENERYS	EVET	VAR	EVET	BEYAZ	TARGARYEN
VISERYS	HAYIR	YOK	EVET	BEYAZ	TARGARYEN
NIGHT KING	HAYIR	VAR	HAYIR	BEYAZ	WHITE WALKER
LIEUTENANT	HAYIR	VAR	HAYIR	BEYAZ	WHITE WALKER
WIGHTS	HAYIR	YOK	HAYIR	BEYAZ	WHITE WALKER

Figure 5: Characters separated

As a result, in this example, we aimed to explain the concept of entropy with an example from popular culture.

Our last example in the decision tree is about graphs. We aimed to introduce people to data analysis and actually let them see how the decision tree algorithm perceives the data. Lens dataset is used in this example in which there exist features including age, spectacle prescription, astigmatic, tear production rate. Each entry in the dataset represents a patient and along with feature, it includes a decision column determining if that patient could use soft lenses, hard lenses or no lenses at all. Below figure shows our example. Columns are features while rows are the decisions that are just mentioned. We wanted people to actually find a relationship between features and the decision thereby doing some generalizations such as people with low tear production rate (rightmost column) cannot wear lenses which is actually the way roughly how decision tree works.

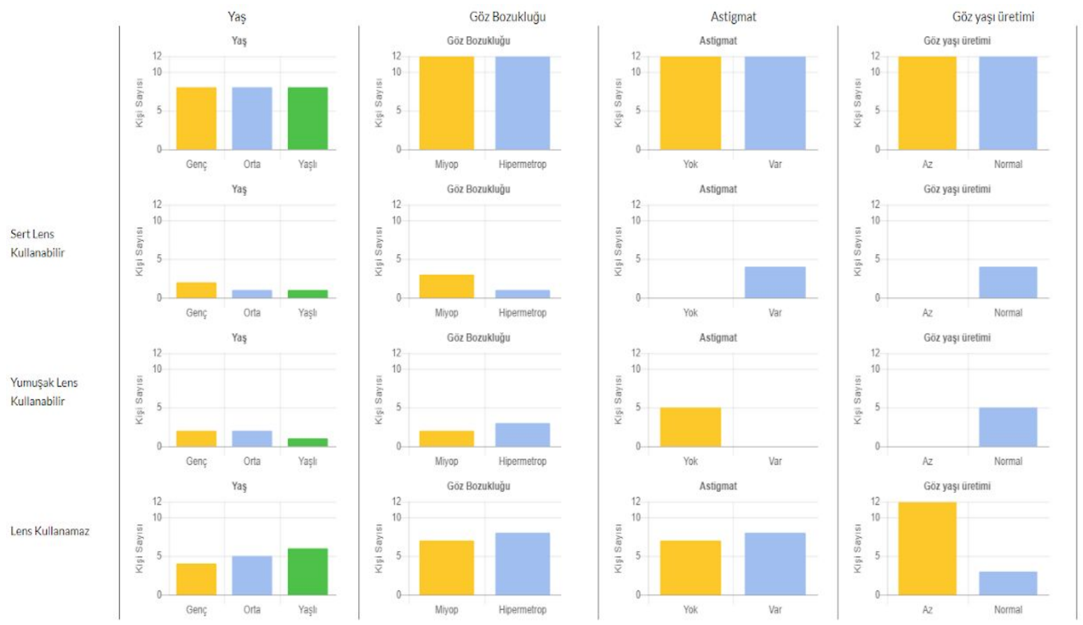


Figure 6: Charts example overview

Thus, we tried to make people think in a way that the algorithm thinks.

2.2 K-Means Algorithm

2.2.1 Visualization

2.2.2 Interactive and Educative Web-Apps

K-Means algorithm is developed to solve clustering problems. Simply put, clustering is gathering the objects with similar features together. One natural clustering problem may occur when a lot of people orders a food, say, pizza. The restaurant manager should share the orders among couriers almost evenly so that customers do not get angry. Below is a screenshot from our example. We used a Barcelona neighborhood as a place due to its interesting urban planning. Each person represents an order and people located in the upper-right of the screen are the couriers.

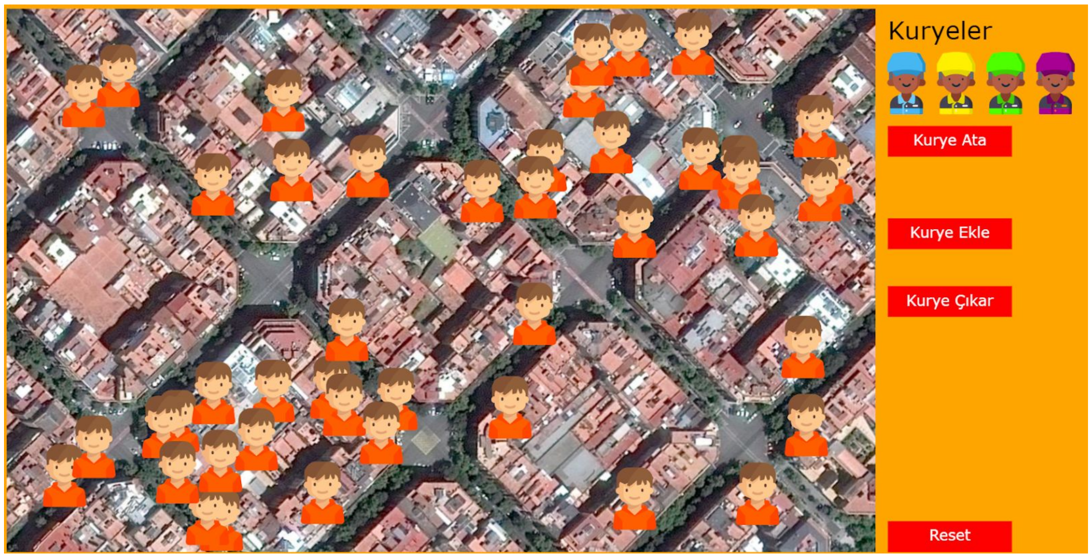


Figure 7: Pizza order example overview

We implemented a basic version of K-Means algorithms. Firstly, one moves each courier into the neighborhood. Then, each customer is assigned to the courier close to himself/herself. After that, courier goes into the middle of the customers assigned to him/her. Finally, as the position of the courier is changed, new customers may be assigned to him/her again. This process continues until almost no more customer switches his/her courier. This is basically how K-Means works. We also gave the user the ability to add and remove couriers to test the algorithm interactively.

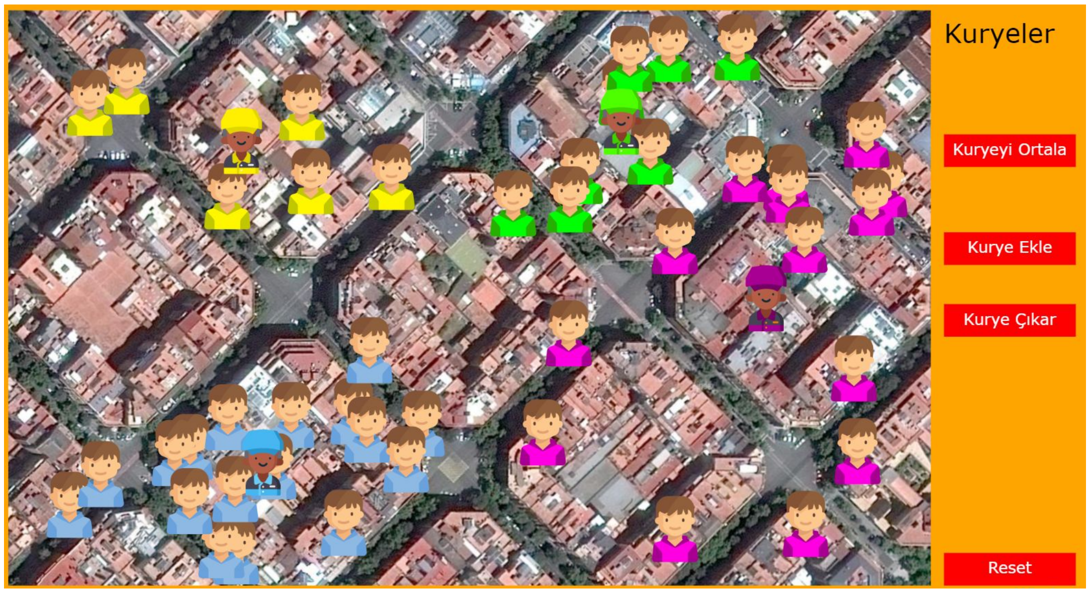


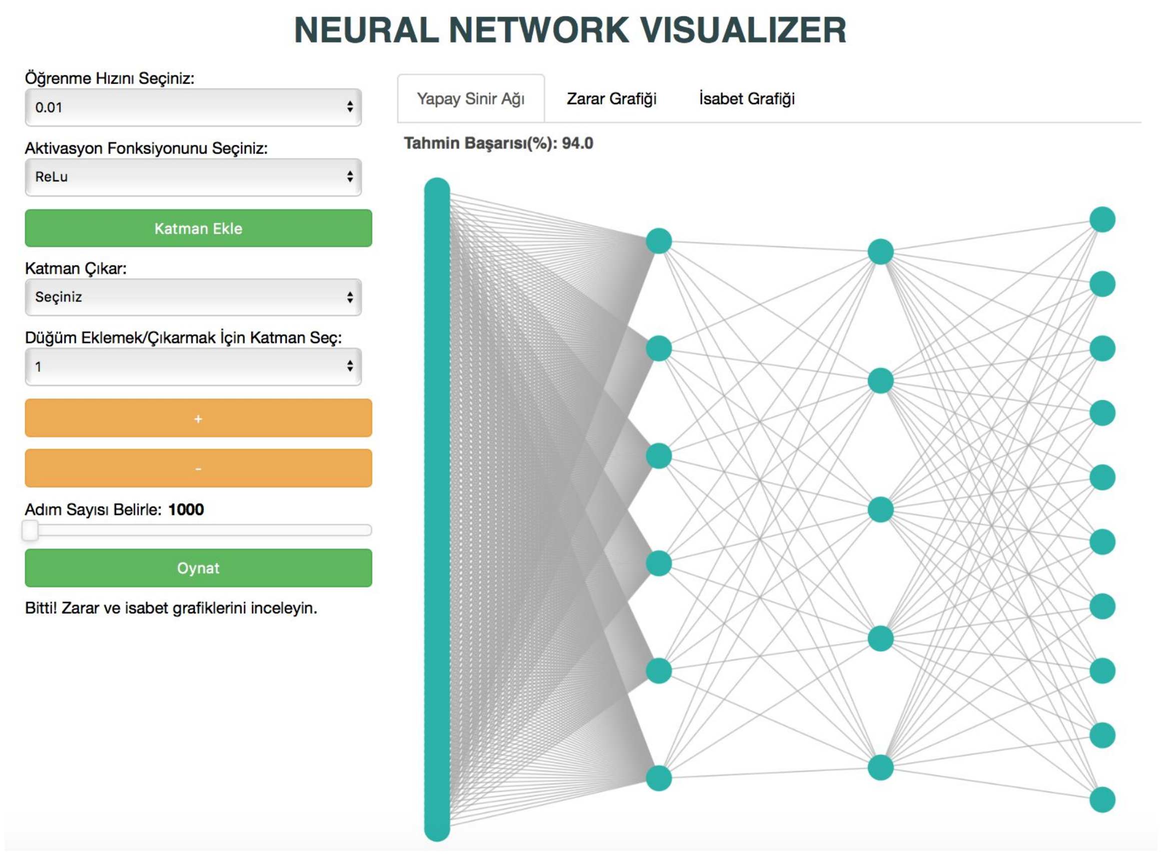
Figure 8: Customers clustered by couriers

As a result, in this example, we aimed to explain the classification problem and how it can be solved by the K-Means algorithm.

2.3 Artificial Neural Networks

2.3.1 Visualization

Lastly, we focused on visualizing Artificial Neural Networks (ANNs). As we did in the previous algorithms, we searched different libraries and examples for the most suitable methods to use. We had two options: scikit-learn, which is a generalized ML algorithms library. It doesn't contain just contain ANN algorithms, it comes with many different classification and clustering algorithms. But in this example, we decided to go with the second option, which is Tensorflow, a specialized library designed by Google for neural networks. We also thought that this was a good chance to learn a new framework in this project. Tensorflow already has built-in methods for constructing the ANN and it also gives us a lot of flexibility to optimize our ANN structure such as epoch numbers, activation functions, number of layers/nodes, learning rate, learning decay, optimizers (stochastic gradient descent and Adam optimizers) and different loss functions. Also, we get different results easily like accuracy, precision, recall, specificity.



Öğrenme Hızını Seçiniz: 0.01

Aktivasyon Fonksiyonunu Seçiniz: ReLu

Katman Çıkar: Seçiniz

Düğüm Eklemek/Çıkarmak İçin Katman Seç: 1

Adım Sayısı Belirle: **1000**

Bitti! Zarar ve isabet grafiklerini inceleyin.

Tahmin Başarısı(%): 94.0

Figure 9: Artificial neural network visualization with Bokeh

We decided to use Bokeh again for the layout of the plot and to get user input. The user can select different combinations of settings and see how the loss and accuracy graphs are affected by choosing a particular tab. The web app gets the user input values and sends them to the ANN library. Since the computation takes a while, a progress bar was added with textual info so the user knows that they have to wait.

To visualize our ANN, we chose the famous MNIST dataset (LeCun, Bottou, Bengio, Haffner, 1998) which contains images of handwritten digits, because it was suitable for classification. In this visualization, we allow users to set epoch numbers, activation functions, number of layers and nodes in each layer and learning rate. Users can understand the effects of different settings. For example, when the user sets the learning rate too low and makes the ANN structure too simple, it can be observed that these settings are not enough to classify images successfully. The user can see that with a low learning rate the system can get stuck on the local minima by observing our loss/accuracy graphs that we give to the user. Also with these graphs, the user can observe different results like when the model is converging.

2.3.2 Interactive and Educative Web-Apps

Artificial neural network (ANN) is a classification problem just as decision tree that simply tries to do prediction, for example, predicting the name of the animal, based on what it learned. In ANN, there is a complex network between input (features) and the outputs (decision). This network consists of bonds each of which has its own weight (or power) to alter the output (decision). In our example, we oversimplified theses bonds and the network as a knob. Change in the value of the knob gives the sense of change in the weights of the bonds. We used animal classification problem particularly classifying dogs and cats. User, in this example, will try to match the animal label with the image. At first, the user will only be see a single knob with 4 switched which naturally will only create 4 possibilities. After that, he/she will realize that with 4 possibility it is very hard to find a setting to match all animals with its label. Then, he/she will be shown the second knob which will increase the number of combinations to 16 and in that way he/she will be able to find a setting so that all animals are successfully paired with its label.

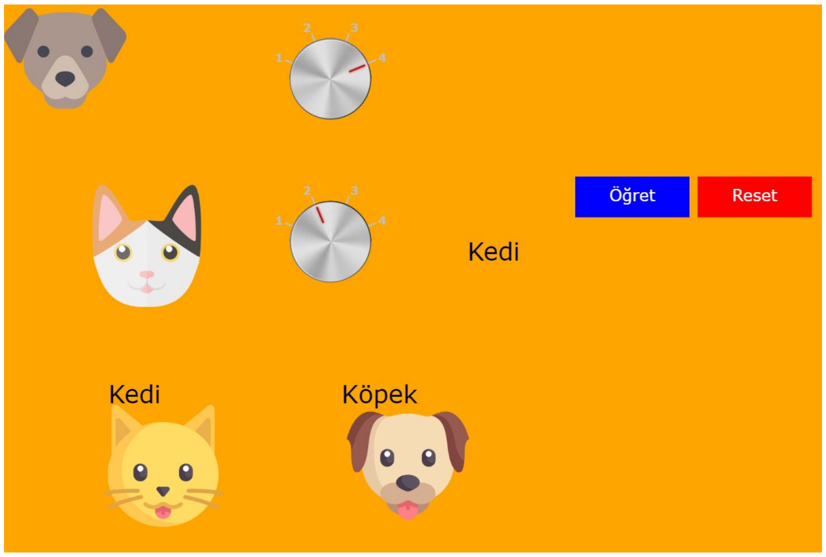


Figure 10: Artifical Neural Network example overview

As a result, we aimed to explain the how the neural network algorithm works.

3 Experiment

After designing and implementing the decision tree, we conducted a simple experiment on a small group of high school students to see if it's an effective method to learn these ML concepts. We prepared a set of questions for each module of the decision tree part to measure the understandability. As we describe these decision tree modules above, we first showed the students diagrams of our data according to its different attributes so we tried to give the students a sense of the data that we are working with. For example, we used the lens dataset (Dua et al. 2017) which classifies patients according to lens type usage depending on attributes such as age, tear production etc. because we thought that it would be an interesting topic to discuss and easier for them to learn about the modules.

According to our feedbacks of the students, we improved and re-designed some properties of the modules and we implemented future parts of the project considering these feedbacks.

4 Conclusion

Feedbacks from non-professionals formed the last version of our interactive and educative website. These were accurate because the experiment was interactive. Therefore, our website also became interactive. In feedbacks, non-professionals stated they could digest the concepts of the ML algorithms well. We are in the last stage of contribution to the Bokeh library. Hopefully, our decision tree visualizer will be published on Bokeh's website. As a result, we produced a beneficial website in Turkish for those who are not familiar with Computer Science or ML to understand the concepts of ML algorithms.

References

Buchheim C., Jünger M., Leipert S. (2002) Improving Walker’s Algorithm to Run in Linear Time. In: Goodrich M.T., Kobourov S.G. (eds) Graph Drawing. GD 2002. Lecture Notes in Computer Science, vol 2528. Springer, Berlin, Heidelberg

Dua, D. and Karra Taniskidou, E. (2017). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

Quinlan, J.R. (1985). *Induction of Decision Trees.* Boston, MA: Kluwer Academic Publishers

Reingold, E.M., & Tilford, J.S. (1981). *Tidier drawings of trees.* IEEE Transactions on Software Engineering, Vol. SE-7, No. 2

Walker, J.Q. (1990). *A Node-Positioning Algorithm for General Trees.* Softw: Pract. Exper., 20: 685-705. doi:10.1002/spe.4380200705