A colorful logo on a black background

Description automatically generated with low confidence

**Name : Soham Chakraborty, Kanek Adalaa and Akhil Palipally**

**Class : 12A Roll No………………………………………**

**Project Specifics: Exploring the Mathematical concepts and Application of derivatives in the Machine Learning Algorithm known as “Linear Regression”.**



|  |  |  |
| --- | --- | --- |
| **S.No.** | **Topic INDEX** | **Page No.** |
| **1.** | **Introduction** | **3** |
| **2.** | **Linear Regression** | **5** |
| **3.** | **Cost Function** | **5** |
| **4.** | **Gradient Descent** | **6** |
| **5.** | **Implementation** | **7** |
| **6.** | **Cost Function Implementation** | **8** |
| **7.** | **Gradient Descent Implementation** | **10** |
| **8.** | **Output** | **18** |
| **9.** | **Conclusion** | **19** |

Introduction

Machine Learning is a branch of Computer Science which deals with machines being able to learn and comprehend on their own, mimicking the behaviors of a human, in order to be able to complete complex tasks like act as a chat-assistant or other such tasks. In this project, I will be going over a particular algorithm of machine learning known as Linear Regression.

# Regression

To delve into the topic of linear regression, we must first delve into the topic of regression. However, to delve into the topic of Regression, let us first understand what supervised learning is.

In supervised learning, we are already given data that we then use to “train” our model. In layman’s terms, this simply means that the expected outputs or “answers” are presented to us, and we just need to make the machine learn from those answers by deducing a correlation between the data points. Each data point consists of an expected input and an expected output. Then, problems of supervised learning are further classified into regression problems and classification problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function.

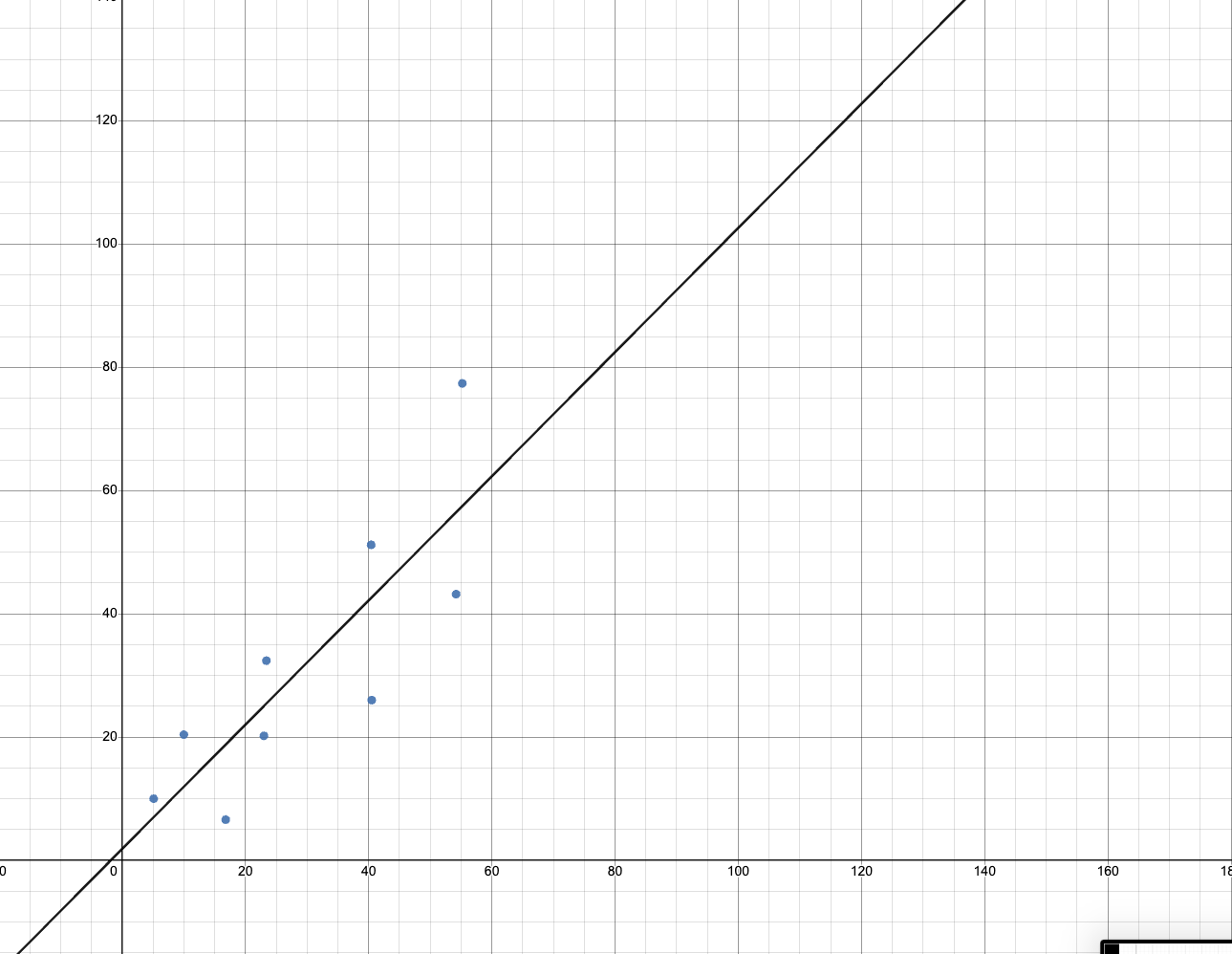
An example of this is the following data points:

A graph with blue dots

Description automatically generated

Let us take the example that these are housing prices of a small town in Rajasthan, India. For this, we will wish to create a model based on this data. However, what is a model? To put simply, a machine learning model is simply a complex equation which is used to predict outputs from inputs.

To give a simple intuition of linear regression, we fit a line through this data, like so:



Linear Regression

In the introduction, I gave a simple intuition to linear regression, but now we will discuss in detail how to find this equation of line, or using the correct term, “the line of best fit”. Let us first assume a polynomial, known as the hypothesis function:

In this function, we have two parameters θ₀ and θ₁. These parameters are computed based on the data given and will be of paramount interest to us.

Before this, we must first calculate how well our parameters work in generating the line of best fit.

# Cost Function

To measure the accuracy of our equation, we use the cost function, this cost function is written as:

Where m is the size of the dataset, is the cost function, is the predicted output from the hypothesis function we defined before of the ith input and is the actual data output value of the ith element.

To give an example, lets say we assume a value of , to test how close we are to the real output, we take the equation (in this case, ) and then put it in the cost function, and compute the value of it. The answer we get gives us the error that the parameters we introduced give and suggest us what improvements we must do. But to estimate the best parameter values, we use a different method

## Gradient Descent

To understand gradient descent, let us first imagine a graph of our parameters with the cost function, which will look something like this:

A rainbow colored waves on a grid

Description automatically generated

This is a plot of our parameters with our cost function . Let us first define our objective, which is to find the values of parameters such that our cost function is the minimum it can be. Now, let us take a look at this graph, we can see some blue spots and red spots, and those show the largest value of our cost function in correspondence with our parameters .

It is also possible to use the concepts of maxima and minima learned in Application of Derivatives to figure out the lowest possible values of . However, it has been found that such a method does not scale well with multiple parameter values, such as in the case of **multivariate linear regression**. Though this is not the topic of this project’s content, to stay consistent with industry and academic standards I have decided to implement a solution to this problem using this given method.

So now we can define a mathematical relation for gradient descent, which is the following:

(For and )

Later, we will also be implementing this algorithm in code, and to do so we must keep in mind that we need to simultaneously update so as to make sure that the changed values are not used for each.

Along with this, we have , which is defined as the learning rate of gradient descent. Depending on the power of a machine we modify this value in order to get an optimum result. This determines the rate at which we proceed towards the value of for which the cost function is minimized, i.e., we will get a more accurate result the lesser we put , and may not get a result if we keep a large value of , due to overshooting the minima.

Before we move on to coding, I would like to give an intuition of this formula. To do so, I’d like to again take the same graph, but with a bit of annotations.

A rainbow colored graph with lines and dots

Description automatically generated

So to give an intuition, we first start with some value of and then work our way through the slope of the graph, essentially, we want to go as downhill as possible, and thus in the immediate surrounding of the point, we would like to find another point as downhill as possible, we then repeat the same with the next point, and repeat this process until we reach the value of such that the is the lowest value it can be. This is basically the process we are carrying out when we use the formula I gave above.

Implementation

It is now time for us to implement the mathematical concepts defined above into a working piece of software which we can then use to train our own models. To do so, I shall be using *python*.

# Introduction

Let us start off by introducing ourselves to the data. In my file structure, I have a dataset in the form of a text file which contains the population of the city (in 10,000s) in the first column and the second column is the profit of the food truck in that city. Now I would like to plot this data, to give us a good visualization of it.

A graph with red dots

Description automatically generated

The code to plot the data is the following:

A screen shot of a computer program

Description automatically generated

# Cost Function

Now we shall code the function to compute the cost function of set parameters . We shall use the formula:

For that purpose, the code is given below:

**A screen shot of a computer program

Description automatically generated**

## Intuition

Let us look at the formula again to familiarize ourselves with this code:

Programming is all about breaking a problem into steps, and so I’d like to segregate this formula into multiple parts, which I shall highlight:

Now that it is easier to understand let us look at the first part of this code:

A screen shot of a computer

Description automatically generated

This is the blue highlighted part of our formula, and I have highlighted this as we need to define this function first if we are to proceed with the next part of the problem.

Now that we have defined the blue part, let us continue with the yellow part of the problem. For that, the code is below:

A black rectangle with yellow and blue text

Description automatically generated

In this, we use a *for loop* to iterate through the dataset that we have, and then add to the *cost* variable. As you can see, we use some special syntax to achieve this, however `(h(X[i]) – y[i])\*\*2` is the same as . Once we have added this, we have essentially added the square of the output subtracted from the result of the cost function to our result and now the last thing we must do is divide it with .

That, results in the final code for implementing the cost function.

To test the cost function, the following code has been given: Picture 1

And thus, we have successfully been able to calculate the cost function for a given pair of parameters.

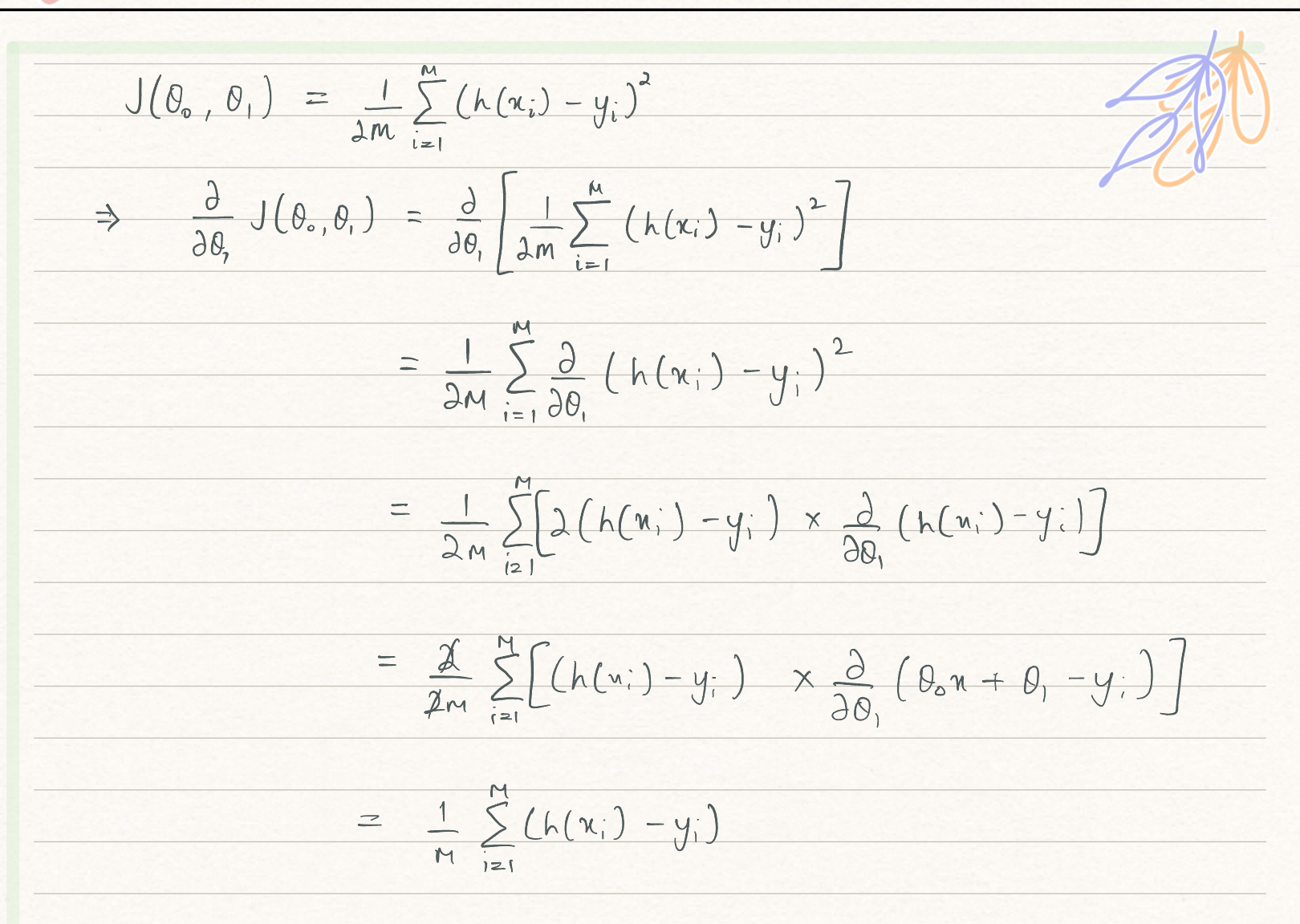
# Gradient Descent

Now that we have defined our cost function, we are going to use it in order to find the gradient descent.

Firstly, we must compute the partial derivative given in the equation:

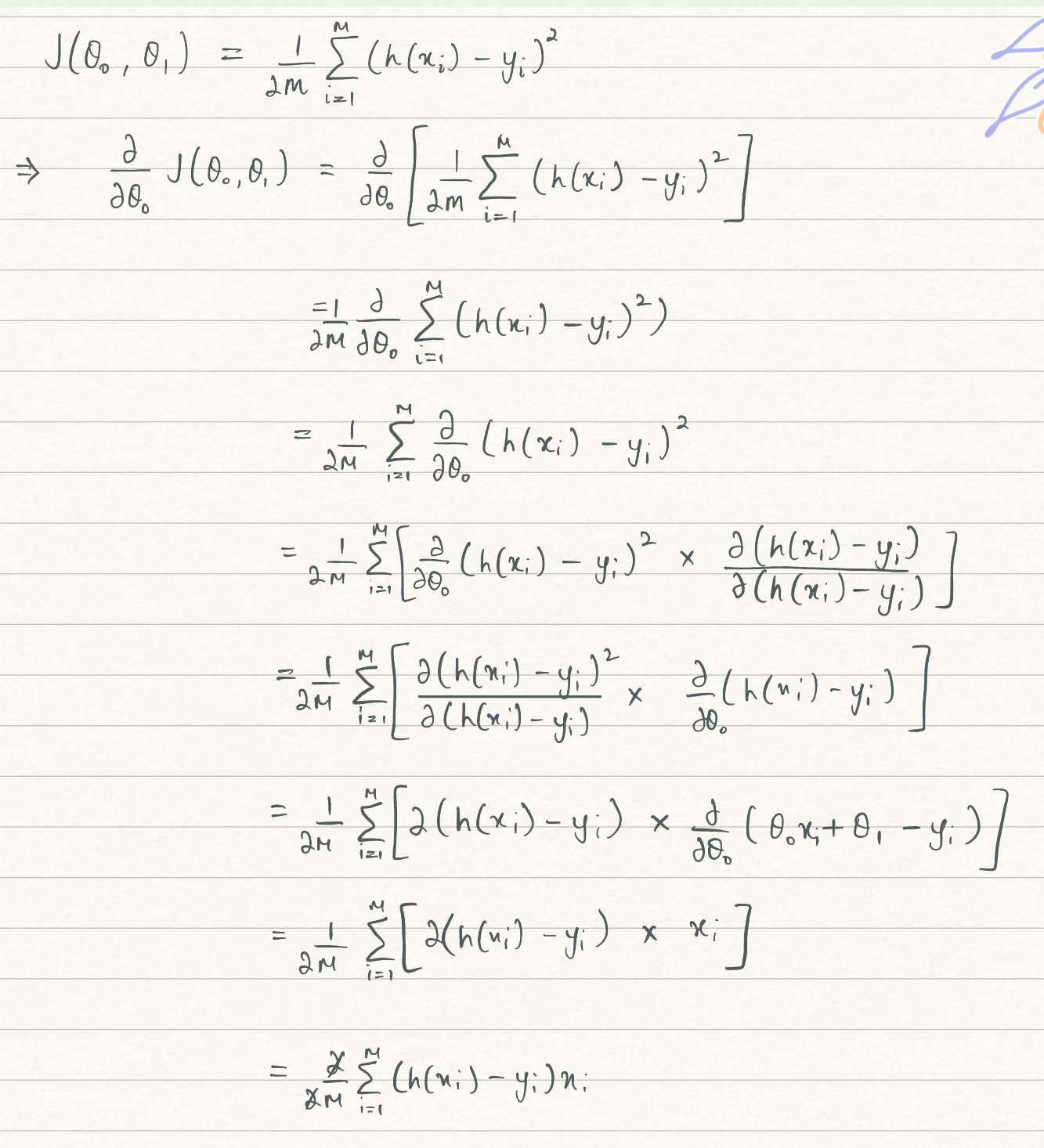
And once we compute this, we get:

The derivation for which is:



And

The derivation for which is the following:



So now our gradient descent equations are:

and

And the code for this is given below: A screen shot of a computer program

Description automatically generated

## Intuition

Now I would like to explain the code that I have given above, I’d like to again, break the problem at hand into small parts. I’d like to reiterate the equations at hand:

and

Now, I will highlight the first equation into parts:

Let us first look at the yellow highlighted part:

The code for this is the following:

A black rectangular object with colorful lights and text

Description automatically generated

Now let us add this to the blue part:

The code for this is the following:

A computer screen with text and numbers

Description automatically generated

Now let us assign this to a temporary variable, to later assign this value to .

A screen shot of a computer program

Description automatically generated

We shall later assign this temporary value simultaneously with .

Now let us go on with the second equation:

The code for the yellow part of the equation is the following:

A blue rectangle with white text

Description automatically generated

Now we add this to the code of the blue part of the equation:

A screen shot of a computer program

Description automatically generated

Now let us assign this to a temporary variable like so: A screen shot of a computer program

Description automatically generated

Now we shall combine these two codes that we have created: A screen shot of a computer program

Description automatically generated

However, this is not enough. In our function, we have a parameter named which allows us to specify the amount of times we want to repeat the process given above. Now let us code for the same:

A screen shot of a computer program

Description automatically generated

Now we just need to assign the temporary values of and and we have the final code:

A screen shot of a computer program

Description automatically generated

The output for the code is the following:



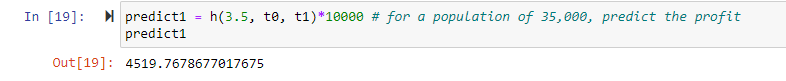
and thus we now have the values of and required for a good fit of our linear regression model.

## Output

Let us now plot the model: A line graph with red dots

Description automatically generated

A sample prediction of the model can be seen below:



# Conclusion

In conclusion, we have used the definition of hypothesis function, cost function and gradient descent in order to implement linear regression in code in order to act on data which we can then use to further business insights, predict a certain value given for a certain input, etc. In this project I not only showed the mathematical equations behind the hypothesis function, cost function and gradient descent but also showed how to implement such mathematical equations to code in order to realize our mathematical and theoretical concepts to practical and real working machine learning models.

Bibliography

Andrew Ng’s course on machine learning on [coursera.org](https://coursera.org/share/1bd100f620e35029edc3601672059494)