Logistic Regression model on labeling Pulsar Candidate

Mathematic Internal Assessment

Introduction:

Machine Learning is currently one of the fastest growing areas in technology, with its application in a vast number of fields, such as biology, economics, astronomy and so on. Examples of recent breakthroughs includes DALL-E-2 from OpenAI, “a new AI system that an create realistic images and art from a description in natural language” [1], or Alpha Fold, “an AI system by DeepMind that predicts a protein’s 3D structure from its amino acid sequence”[2]. Alpha Fold in particular, was able to become the top ranked protein prediction method during the CASP14 competition by a large margin, producing a model with high accuracy of correspondence to an actual protein sequence. Together, they greatly advance their respective industry.

As a programmer and aspiring astrophysicist who has been developing projects in many areas for 6 years, I have worked in areas such as app/web development, game development, gravitational simulation in which I used for my extended essay, and eventually Machine Learning. Taking courses in machine learning, I usually hide myself from the beautiful mathematics behind the development of those machine learning models. During this investigation, I aimed to explore the calculus and other mathematics behind machine learning by building a logistic regression model from scratch in Python, applying them to the area of astronomy where I will apply my model to classify neutron stars.

Background:

According to IBM, Machine learning is an area of computer science in which we use data and algorithms to imitate the way that humans learn, improving accuracy [3]. In order word, it is a set of instructions(algorithms) that is able to improve itself through the use of data. Machine

learning can be divided into many different types of method, two types are a linear model and a non linear model. A nonlinear model such as the neuronetwork is the newest model in the industry that helps to solve complicated problems such as translating languages, generating text and images or driving cars. A linear model is a model that tries to learn from a dataset that has linear correlation, creating a line of best fit to learn from the data. A popular example of this type of algorithm is the linear regression model. A few modifications to the linear regression model will then help it to classify instead of estimating data; this type of algorithm is called logistic regression[4]. Nowadays, These methods are often referred to as “old school” because of their rather simple approach. However, there seems to be potential in using those “old school” approaches to solve many problems that we are facing nowadays. One of such is to differentiate pulsar stars signals from space.

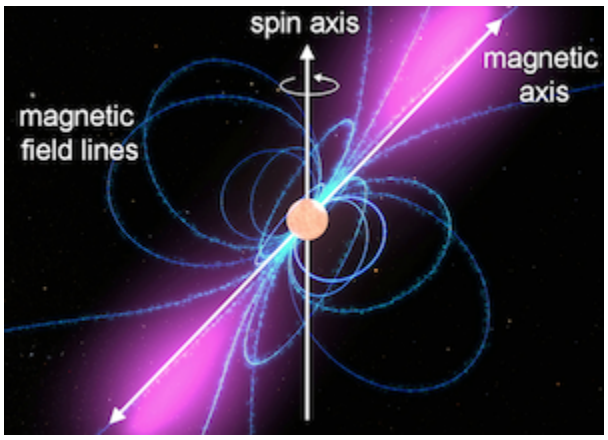


Diagram 1: Diagram of pulsar neutron star with strong magnetic field [6]

This investigation will be focusing on applying those ideas to classify neutron stars. Dust clouds are clouds of particles of elements floating in space, when those clouds are sufficiently massive enough, the gravitational force between them makes them collapse under their own weight and create stars. The pressure of the gravitational force within stars is strong enough to

make a reaction called nuclear fusion happen, where simple elements such as hydrogen fused together to create heavier helium, this process then convert the energy of their mass to energy. This energy then acts as the outward pressure acting against gravity to help the star from further collapsing, however, when the hydrogen runs out, the star further collapses. If the star is sufficiently massive enough, the pressure would then again fuse the helium into even heavier elements, and create what is called a massive star[5]. At the end of its lifetime, if the massive star's core is between 1 to 3 solar masses, the star will collapse and leave a neutron star. Most neutron stars are undetected due to most of them not having enough magnetic field. However, when the spin of a neutron star is fast enough, it emits detectable radiation that can be gathered on earth. These stars can be labeled as pulsar or magnetar. A pulsar is a neutron star with a magnetic field strong enough to funnel jets of particles along its poles, those high energy particles then emit the radiation that we would detect[6]. This type of star is of scientific interest as they are probes of space time, interstellar medium and states of matter. As a pulsar star rotates rapidly, their radio signal emits a certain pattern, this pattern can be varied amongst the pulsar stars. This pattern can slightly vary for those stars with respect to their rotation. A potential signal is called a "candidate", candidates of a star are averaged over many rotations of the star in order to classify them. "However, most detections are caused by radio frequency interference and noise, making legitimate signals hard to find." This creates an opportunity for a machine learning model to be developed to differentiate out pulsar star signals[7].

Dataset

The dataset that the investigation will be using is the HTRU2 dataset. Initially discovered on Kaggle, the dataset describes a sample of pulsar candidates collected during the High Time Resolution Survey[7]. The dataset features 12528 examples, with 1,639 positive examples and

16,259 negative examples, however, only a portion of 12528 will be used during training the data, the rest was taken out by the people who organized the Kaggle competition. The attributes of the dataset, or as this investigation will be referred to as features contain 8 quantitative features and 1 binary qualitative feature. Those feature are as below:

- Mean of integrated profile
- Standard deviation of integrated profile
- Excess Kurtosis of integrated profile
- Skewness of integrated profile
- Mean of DM-SNR curve
- Standard deviation of DM-SNR curve
- Excess Kurtosis of the DM-SNR curve
- Skewness of the DM-SNT curve
- Class

An integrated profile is when a waveform of a pulsar emitted periodically is averaged synchronously over hundreds of pulses or more. Its means, standard deviation, excess kurtosis and skewness represent our first four features. The Dispersion Measure/ Signal to Noise Ratio curve(DM/SNR) is the curve that measures the dispersion of the emitted radio signal when it travels through space[9]. Its means, standard deviation, excess kurtosis and skewness represent our next four features. The last feature is a binary feature representing the class of the candidate. If the candidate's class is 1, then the candidate was determined to be a pulsar star, if the candidate is 0, then it is not.

Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
121.15625	48.37297113	0.375484665	-0.013165489	3.168896321	18.39936666	7.449874149	65.15929771	0
76.96875	36.17555664	0.712897866	3.388718563	2.399665552	17.57099693	9.414652256	102.7229747	0
130.5859375	53.22953353	0.133408289	-0.297241641	2.743311037	22.36255299	8.508363784	74.0313242	0
156.3984375	48.86594223	-0.215988596	-0.171293649	17.47157191		2.958065943	7.197841911	0
84.80468675	36.11765898	0.825012787	3.274125373	2.790133779	20.61800857	8.405008383	76.29112787	0

Table 1: First five candidate in the dataset

Data processing [Self note: Feature scaling, deriving the equation]

There are a few steps we need to do before we can use our model on the data. Let's first looks at the information that describe the dataset:

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
mean	111.0418	46.52144	0.412274	1.778431	12.67476	23.87352	8.333489	100.2613	0.092034
std	25.67283	6.801077	1.00196	6.20845	29.61323	20.18857	4.535783	107.1776	0.289085
min	5.8125	24.77204	-1.73802	-1.79189	0.213211	0	-3.13927	-1.97698	0

max	189.7344	91.80863	8.069522	68.10162	222.4214	110.6422	34.53984	1191.001	1

Table 2: Information about pulsar stars

The range of each data is different, this can lead to a slow down in gradient descent, and furthermore, the value that will be running through the model will be multiplied and summed up, this will lead to values that are too big for the computer to handle. Therefore we can do what is called feature scaling, which limits the range of the data between 0 and 1, this helps us to deal with.

$$x_{new} = \frac{x_{old} - \mu}{max - min}$$

	Mean of the integrated profile	Standard deviation of the integrated profile	Excess kurtosis of the integrated profile	Skewness of the integrated profile	Mean of the DM-SNR curve	Standard deviation of the DM-SNR curve	Excess kurtosis of the DM-SNR curve	Skewness of the DM-SNR curve	target_class
0	0.054993	0.02762	-0.00375	-0.02563	-0.04278	-0.04948	-0.02345	-0.02942	0
1	-0.18526	-0.15433	0.030652	0.023039	-0.04624	-0.05696	0.028694	0.002063	0
2	0.106263	0.100066	-0.02843	-0.0297	-0.04469	-0.01366	0.004641	-0.02199	0
3	0.246608	0.034974	-0.06406	-0.0279	0.021587	-0.21577	-0.14266	-0.07801	0
4	-0.14265	-0.1552	0.042084	0.0214	-0.04448	-0.02942	0.001898	-0.02009	0

Table 3: New scaled values of the dataset

Lastly, we will split the data into 1:19 ratio, with the smaller set of candidates being left out of the training process, which will help us to see how well the model can be generalized into other candidates that it has never seen before.

$$testSet = \frac{12528}{20} => 627$$

$$trainSet = 626.4 \cdot 19 => 11.901$$

Model development:

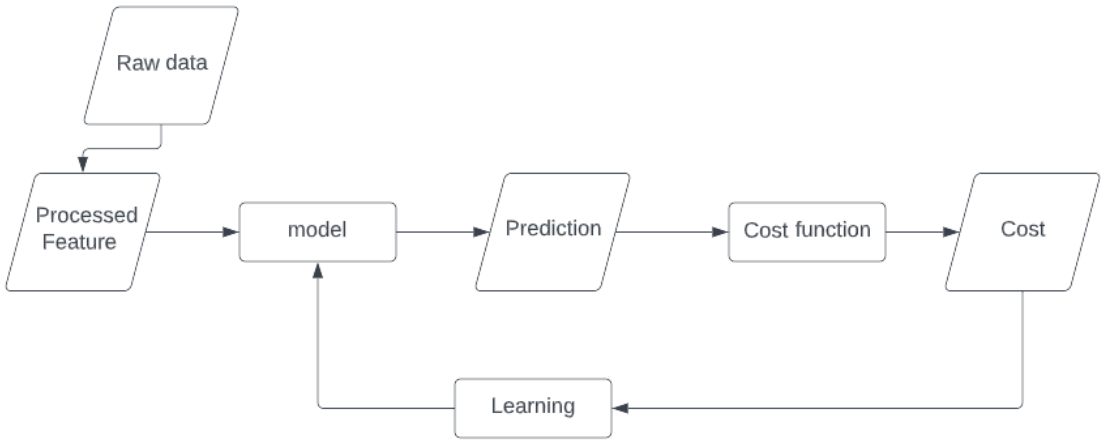


Diagram 2: Routine of training a machine learning model

The model is developed based on a simple routine, shown in diagram 2. We can start by developing a simple model of linear regression, then move on developing a more advanced neural network.

We can first determine a mathematical representation of the linear regression model, since a linear regression model is in itself just a line of best fit problems, its equation is just a line:

$$h(x) = \theta x + b$$

Theta itself can be split into theta 1 and theta 0:

$$h(x) = \theta_1 x + \theta_0$$

If we have more than 1 input such as x_1, $x_2 \ldots x_j$, we can simply this this expression put all the x_1, $x_2 \ldots x_j$ into one vector of x, then we can change theta into a vector of its own of theta, we can then write an equivalent equation for a linear regression model as

$$h(x) = \theta \cdot x$$

The cost function is the function that would measure the amount of error our prediction would make relative to the real answer. The simplest method is the mean squared error (MSE), the cost of the prediction for any candidate at index i would be as followed:

$$C_i = (y_i - h(x_i))^2$$

That costs will then be averaged over n number of candidates to derive the MSE for each iteration :

$$C = \frac{1}{n} \sum_{i=0}^{n} (y_i - h(x_i))^2$$

For linear regression, for the model to start to fit to the data, or to learn the data, the parameter of theta will need to be changed in order to minimize the cost. As the traditional way of setting the gradient of the cost function to 0 is rather expensive computationally, models, especially more complicated models such neural networks use the concept of gradient descent.

Gradient descent in essent, is to let theta go down the gradient of the cost function at some interval of a.

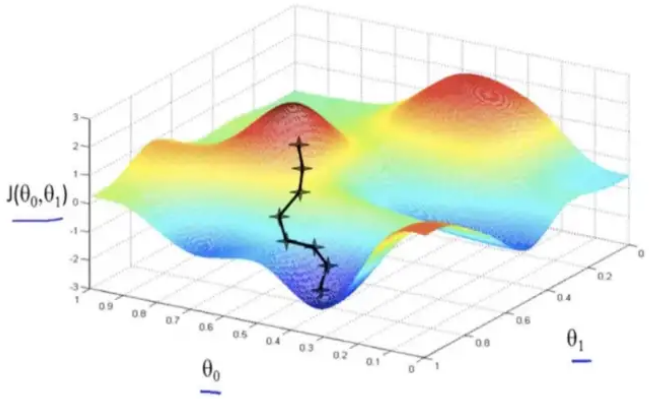


Diagram 3: illustrating gradient descent[10]

In equation form, it would looks like this:

$$\theta_j \;=\; \theta_j \;-\; \frac{dC\left(x,\theta_j\right)}{d\theta_j}$$

As we have two different parameters for the equation we can change, the theta that we wished to change at each iteration would be labeled with the index of j. The partial derivative of the cost function in respect to theta of index j can solved by first take the derivative of the cost for each candidate:

$$\frac{dC_i\left(x,\theta_j\right)}{d\theta_j} \;=\; \frac{dC_i\left(x,\theta_j\right)}{dh\left(x_i,\theta_j\right)} \cdot \frac{dh\left(x_i,\theta_j\right)}{d\theta_j}$$

$$\frac{dC_i\left(x,\theta_j\right)}{d\theta_j} \;=\; \frac{dC_i\left(x,\theta_j\right)}{dh\left(x_i,\theta_j\right)} \cdot \frac{dh\left(x_i,\theta_j\right)}{d\theta_j}$$

$$\frac{dC_i\left(x,\theta_j\right)}{d\theta_j} \;=\; \frac{d\left(y-h\left(x_i,\theta_j\right)\right)^2}{dh\left(x_i,\theta_j\right)} \cdot \frac{dh\left(x_i,\theta_j\right)}{d\theta_j}$$

$$\frac{dC_i\left(x,\theta_j\right)}{d\theta_j} \;=\; \frac{d\left(y-h\left(x_i,\theta_j\right)\right)^2}{dh\left(x_i,\theta_j\right)} \cdot \frac{d\left(\theta_1 x \;+\; \theta_0\right)}{d\theta_j}$$

$$\frac{dC_i\left(x,\theta_j\right)}{d\theta_j} \;=\; 2\left(y-h\left(x_i,\theta_j\right)\right) \cdot \frac{d\left(\theta_1 x \;+\; \theta_0\right)}{d\theta_j}$$

This then lead to two different equation we need to execute gradient descent:

$$\frac{dC_i\left(x,\theta_0\right)}{d\theta_0} \;=\; 2\left(y-h\left(x_i,\theta_0\right)\right)$$

$$\frac{dC_i\left(x,\theta_1\right)}{d\theta_1} \;=\; 2\left(y-h\left(x_i,\theta_1\right)\right)x_1$$

The gradients for all candidates will be summed and averaged to be solve for the gradient at each training step:

$$\frac{dC\left(x,\theta_0\right)}{d\theta_0} \;=\; \frac{2a}{n}\sum_{i=0}^{n}\left(y-h\left(x_i,\theta_0\right)\right)$$

$$\frac{dC\left(x,\theta_1\right)}{d\theta_1} \;=\; \frac{2a}{n}\sum_{i=0}^{n}\left(y-h\left(x_i,\theta_1\right)\right)x_1$$

For simplicity, we will another x in our dataset, an x at index 0 would always have the value of 1, hence combining the two equation into a final form of:

$$\frac{dC\left(x,\theta_j\right)}{d\theta_j} \;=\; \frac{2a}{n}\sum_{i=0}^{n}\left(y-h\left(x_i,\theta_j\right)\right)x_j$$

The same concepts would apply to developing a logistic regression model. In order to

$$g\left(x\right) \;=\; \frac{1}{1+\;e^{-x}}$$

classify data using the linear regression model, we will need to add another layer to the equation. The sigmoid function is that has a range of $y \in [0, 1]$.

$$h\left(x,\;\theta\right) \;=\; g\left(\theta\cdot x\right)$$

Labeling a sigmoid function as g(x), we can apply it over our linear regression model to limit the range of our prediction in between 1 and 0.

This then changes the expression for our cost function. The cost function we can then use is the what commonly known as the binary cross entropy cost function

$$C\left(h\left(x\right),\;y\right) \;=\; \frac{1}{m}\sum_{i=0}^{n} y\log\left(h\left(x\right)\right)\;+\left(1-y\right)\log\left(1-h\left(x\right)\right)$$

To understand how the function work, we can look at it's graph

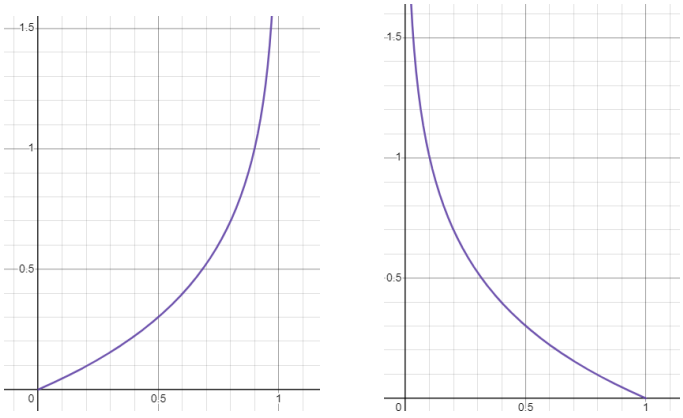


Diagram 4: on the left is when y = 0, and on the right is when y = 1

When y is equal to 0, the function made it so that values closer to 0 would make the cost function output a lower value, when y is equal to 1, it would make it so the output would be the cost value would be lower if our model predicted a value closer to 1.

Now we can take the derivative of this cost function to derive an algorithm we can use. We can first set up our derivative chain as follow

$$\frac{dC\left(x,\theta\right)}{d\theta} = \frac{dC\left(x,\theta\right)}{dh\left(g\left(x,\theta\right)\right)} \cdot \frac{dh\left(g\left(x,\theta\right)\right)}{dg\left(x,\theta\right)} \cdot \frac{dg\left(x,\theta\right)}{d\theta}$$

We can then solve the derivative of the first part of the derivative chain.

$$\frac{dC\left(x,\theta\right)}{dh\left(g\left(x,\theta\right)\right)} = \frac{d}{d\left(h\left(g\left(x,\theta\right)\right)\right)}\left(y\log\left(h\left(g\left(x,\theta\right)\right)\right) + \left(1-y\right)\log\left(1-h\left(g\left(x,\theta\right)\right)\right)\right)$$

$$\frac{dC\left(x,\theta\right)}{dh\left(g\left(x,\theta\right)\right)} = \frac{d}{d\left(h\left(g\left(x,\theta\right)\right)\right)} y\log\left(h\left(g\left(x,\theta\right)\right)\right) + \frac{d}{d\left(h\left(g\left(x,\theta\right)\right)\right)}\left(1-y\right)\log\left(1-h\left(g\left(x,\theta\right)\right)\right)$$

$$\frac{dC\left(x,\theta\right)}{dh\left(g\left(x,\theta\right)\right)} = \frac{y}{h\left(g\left(x,\theta\right)\right)} + \left(1-y\right)\cdot\frac{1}{\left(1-h\left(g\left(x,\theta\right)\right)\right)}$$

$$\frac{dC\left(x,\theta\right)}{dh\left(g\left(x,\theta\right)\right)} = \frac{y}{h\left(g\left(x,\theta\right)\right)} + \frac{\left(1-y\right)}{\left(1-h\left(g\left(x,\theta\right)\right)\right)}$$

The derivative of the second part, which is a sigmoid derivative

$$\frac{dg\left(h\left(x,\theta\right)\right)}{dh\left(x,\theta\right)} = \frac{d}{dh\left(x,\theta\right)}\cdot\left(\frac{1}{1+e^{-h(x,\theta)}}\right)$$

$$\frac{dg\left(h\left(x,\theta\right)\right)}{dh\left(x,\theta\right)} = \frac{d}{dh\left(x,\theta\right)}\cdot\left(1+e^{-h(x,\theta)}\right)^{-1}$$

$$\frac{dg\left(h\left(x,\theta\right)\right)}{dh\left(x,\theta\right)} = -1\left(1+e^{-h(x,\theta)}\right)^{-2}\cdot -e^{-h(x,\theta)}$$

$$\frac{dg\left(h\left(x,\theta\right)\right)}{dh\left(x,\theta\right)} = \frac{e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)^{2}}$$

We can keep this form, but to solve the general derivative chain, I found that we need to convert it into a different form[11], which is shown as follow

$$\frac{e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)^2} = \frac{1\cdot\ e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)\cdot\left(1+e^{-h(x,\theta)}\right)}$$

$$\frac{e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)^2} = \frac{1}{\left(1+e^{-h(x,\theta)}\right)}\cdot\frac{e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)}$$

$$\frac{e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)^2} = \frac{1}{\left(1+e^{-h(x,\theta)}\right)}\cdot\frac{1-1+\ e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)}$$

$$\frac{e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)^2} = \frac{1}{\left(1+e^{-h(x,\theta)}\right)}\cdot\left(\frac{1}{\left(1+e^{-h(x,\theta)}\right)}-\frac{1+\ e^{-h(x,\theta)}}{\left(1+e^{-h(x,\theta)}\right)}\right)$$

$$\frac{dg\left(h\left(x,\theta\right)\right)}{dh\left(x,\theta\right)} = g\left(h\left(x,\theta\right)\right)\cdot\left(1-g\left(h\left(x,\theta\right)\right)\right)$$

And then we can derive the last part with respect to theta at j, which represent any theta we aiming to update at each learning interval

$$\frac{dh\left(x,\theta\right)}{d\theta_j} = \frac{d}{d\theta_j}\theta\cdot x$$

$$\frac{dh\left(x,\theta\right)}{d\theta_j} = \frac{d}{d\theta_j}\left(\theta_0 x_0\ +\ \theta_1 x_1\ ....\ \theta_j x_j\right)$$

$$\frac{dh\left(x,\theta\right)}{d\theta_j} = x_j$$

Putting all the part together, we can then solve for a way to solve for gradient at each time step

$$\frac{dC\left(x,\theta\right)}{d\theta_j} = \left(\frac{y}{g\left(h\left(x,\theta\right)\right)}\ -\ \frac{(1-y)}{1-g\left(h\left(x,\theta\right)\right)}\right)\cdot g\left(h\left(x,\theta\right)\right)\left(1-g\left(h\left(x,\theta\right)\right)\right)\cdot x_j$$

$$\frac{dC(x,\theta)}{d\theta_j} = \left(\frac{y}{g(h(x,\theta))} - \frac{(1-y)}{1-g(h(x,\theta))}\right) \cdot x_j$$

$$\frac{dC(x,\theta)}{d\theta_j} = \left(\frac{g(h(x,\theta))(1-g(h(x,\theta)))y}{g(h(x,\theta))} - \frac{g(h(x,\theta))(1-g(h(x,\theta)))(1-y)}{(1-g(h(x,\theta)))}\right) \cdot x_j$$

$$\frac{dC(x,\theta)}{d\theta_j} = (y - yg(h(x,\theta)) - g(h(x,\theta)) + yg(h(x,\theta))) \cdot x_j$$

$$\frac{dC(x,\theta)}{d\theta_j} = (y - g(h(x,\theta))) \cdot x_j$$

For each theta at a j, we can then take the mean over it's derivative over all x, which would come out as follow

$$\frac{dC(x,\theta)}{d\theta_j} = -\frac{1}{n}\sum_{i=0}^{n}(y - g(h(x,\theta))) \cdot x_j$$

Applying this equation over the equation of form of a gradient descent, we would have the equation we would use to update the parameter of our model over time with learning rate of a:

$$\theta_j = \theta_j + a \cdot \frac{1}{n}\sum_{i=0}^{n}(y - g(h(x,\theta))) \cdot$$

Evaluating model:

1. Validating model:

To validate the model, we can first test by plugging one of the first five candidates and do a test gradient descent. Let's start by generating our theta vector with all elements equal 1, and x vector from all the features of the first candidate.

X	0.054993	0.02762	-0.00375	-0.02563	-0.04278	-0.04948	-0.02345	-0.02942
Theta	1	1	1	1	1	1	1	1

Table 4: Showing the values of testing x and theta

From table 1, we know that this candidate is a false candidate and is not a pulsar star, which means that the expected output of the model would be that of 0. Let's see what our model would predict the candidate to be:

$$h\left(x,\ \theta\right)\ =\ \theta\cdot x\ =\ -0.0919018$$

$$g\left(-0.0919018\right)\ =\ \frac{1}{1+e^{0.0919018}}\ =\ 0.477041$$

The model predicts that 0.47% the candidate to be a pulsar, which is already not bad. We can then see what the cost value for such prediction would be:

$$C\left(0.477041,\ 0\right)\ =\ -\left(\left(0\right)\cdot\log\left(0.477041\right)\ +\ \left(1-0\right)\log\left(1-0.477041\right)\right)\ =\ 0.281532$$

A score of 0.28 is not a bad score, but we can do better. We can apply to the model a mini gradient descent over this one candidate, which can help us to validate the algorithm developed in the last section. Let's first calculate the gradient of relative to all theta.

$$\frac{dC\left(x,\theta\right)}{d\theta_j}\ =\ \left(0\ -\ \ 0.477041\right)\ \cdot\ x$$

The gradient vector would turn out to look like this:

dC	-0.0262339	-0.0131758	0.0017889	0.0122265	0.0204078	0.0236039	0.01118661	0.0140345

	1571	7242	0375	6083	1398	8868	145	4622

With the example value of a to be 0.1, we can subtract this the gradient to get a new set of theta:

Theta	0.9973766 084	0.9986824 128	1.0001788 9	1.0012226 56	1.0020407 81	1.0023603 99	1.00111866 1	1.0014034 55

We can have our model to predict again with this new set of theta:

$$h\left(x,\ \theta\right)\ =\ \theta\cdot x\ =\ -0.0923861$$

$$g\left(-0.0923861\right)=\ \frac{1}{1+e^{0.0923861}}\ =0.476920$$

$$C\left(0.476920,\ 0\right)\ =\ -\left(\left(0\right)\cdot\log\left(0.476920\right)\ +\ \left(1-0\right)\log\left(1-0.476920\right)\right)\ =\ 0.281432$$

Voila!, our model seems to be successful at learning from the error that it produces, as the cost went down by 0.001. Even though this is a very tiny value, the value will be run in the program over a 10000 iteration with our model learning from over 10000 candidates.

2. Computed model:

   To sum up, the model ran over 10000 iterations, with a learning rate of 0.05, trained on a set of 11901 candidates and the final performance was validated on a set of 627 candidates. The result are as follows

Initial Theta	0.0488135	0.2151 8937	0.1027 6338	0.0448 8318	-0.076 3452	0.1458 9411	-0.062 41279	0.3917 73	0.4636 6276
Final Theta	-2.537 999	-1.865 629	-0.710 207	1.6248 67	1.2534 62	1.1238 28	1.5081 51	-0.589 576	-0.018 547

Table 5: initial and final theta

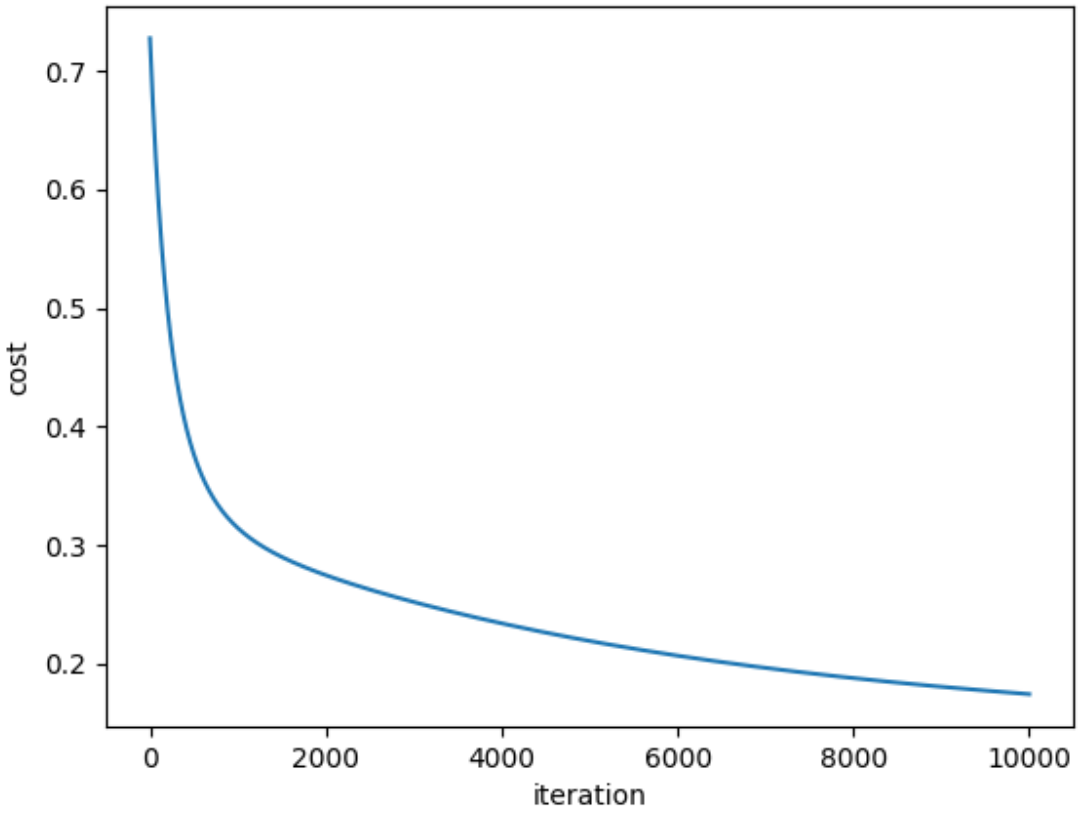
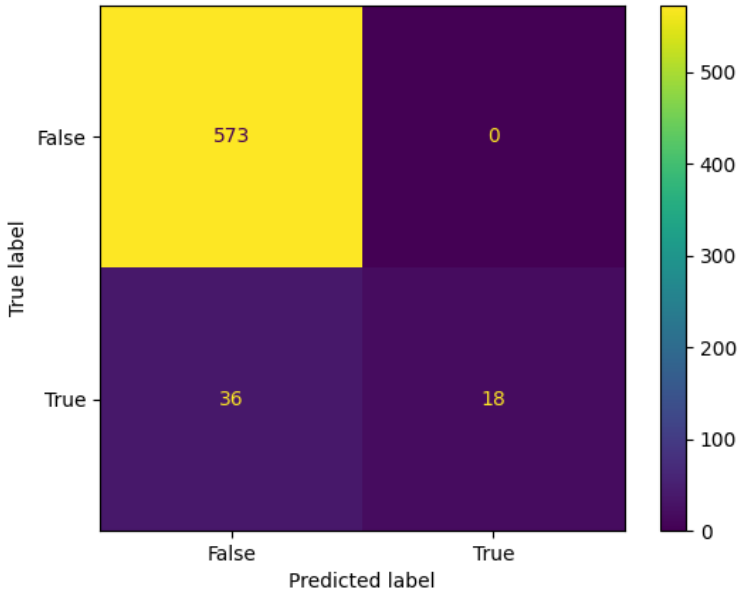


Diagram 5: Cost measures over learning iterations

After 10000 iterations, the model was able to achieve the cost of 0.174560 on the last iteration of the training set. The result when the model was aimed to predict on the training set came out to have the cost of 0.175027. The low difference between the two costs between the

training set and the testing test means that the model has successfully been generalized to predict pulsars candidates that it has not seen before.



ram 6: confusion matrix on the prediction of the model

Rounding any prediction that is over 0.5 to be equal to 1, and any other result to be 0, we can produce a confusion matrix and an accuracy score for the model. The model impressively has the accuracy of 94.3%, with the diagram 6 showing that it was able to recognize non-pulsar out of the data with a 100% accuracy. However, table one also shows that the model is rather not good at differentiating out which candidate is a pulsar star. We can see this bias of the model due to the relatively low number of positive candidates.

Conclusion and reflection:

During this investigation, I was able to create a model of logistic regression from scratch, written it in Python and was able to achieve an accuracy of 94.3%. However, there are still a few ways that our model can be improved upon. First, the model can be designed better for it to achieve a better performance on predicting positive candidates. A few ways I can think of are to exclude a few negative candidates from the dataset so as not to be biased to the trend in the dataset itself. A more complicated model such as the neuronetwork, which I initially was doing my paper on, due to its complexity, may be able to achieve better performance. I eventually didn't follow through with the algorithm due to the scope of it can't be fit in a single 20 page internal assessment.

In general, I was glad that I can explore the math behind one of the simpler mathematics models, and although not followed through, partially explored the mathematics behind gradient descent for even a neuronetwork, otherwise called backpropagation. Although I had to switch to a simpler model, I was satisfied and baffled by the performance that it was able to achieve. The neuronetwork itself could be acting as an interesting extension to my investigation, or to improve even further on a model developed for my research. Another type of math that I found can be further explored is to fix over-fitting, however, the small difference between the cost did make it seems like the model was able to generalize to predicting other candidates.

Citation:

[1]https://openai.com/dall-e-2/

[2]https://alphafold.ebi.ac.uk/

[3]https://www.ibm.com/cloud/learn/machine-learning

[4]https://blog.dataiku.com/top-machine-learning-algorithms-how-they-work-in-plain-english-1

[5]https://science.nasa.gov/astrophysics/focus-areas/how-do-stars-form-and-evolve

[6]https://imagine.gsfc.nasa.gov/science/objects/neutron_stars1.html

[7]https://archive.ics.uci.edu/ml/datasets/HTRU2#

[8]"mean profile." McGraw-Hill Dictionary of Scientific & Technical Terms, 6E. 2003. The McGraw-Hill Companies, Inc. 27 Nov. 2022 https://encyclopedia2.thefreedictionary.com/mean+profile

[9]https://essay.utwente.nl/71435/1/GROOTJANS_MA_EWI.pdf

[10]https://medium.com/@DBCerigo/on-why-gradient-descent-is-even-needed-25160197a635

[11]https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e