ML Bootcamp

**Anant Upadhyay**

**IIT (ISM) Dhanbad**

**Algorithms used:-**

* Linear regression:-Train a linear model based on the training dataset using gradient descent.Find the line of best fit from the training dataset.
* Polynomial regression:-Train a polynomial model containing square,cube or higher powers of the features.
* Logistic regression:-Train the model using the sigmoid function.This is used for classification problems as then we can classify using a threshold.Eg:- a>=0.5 then outcome is positive otherwise negative.
* Neural networks:- Construct a n-layer neural network to make predictions on the given dataset.Then adjust the model’s parameters by the help of back propagation.I will train the model by nudging weights and biases of each neuron.
* KNN:-This algorithm finds the “k” nearest neighbors of the datapoint that is to be predicted, and then bases its prediction on its nearest neighbors.
* Other parameters/procedures:- I have added an additional parameter to regularize our data(for dealing with bias and variance).Also for increasing speed and accuracy we can normalize the given data(If required)

**Technology stack :-**

* Basic knowledge of the python programming language
* Knowledge of the numpy library to easily work with arrays,vectors and matrices
* Knowledge of pandas library to load,edit and manipulate data
* Knowledge of the matplotlib library to show visualizations of our model using graphs,scatter plots etc.
* Knowing the usage of Jupyter/google Collab notebooks(to document our project and present it in the form of a report).Eg:-Usage of markup and code cells,running cells etc.
* Knowledge of classes and vectorization in python(To make implementation easier,faster and later convert the work in a python library)
* Knowledge of the working and implementation of different algorithms and also how to train them.(Knowing how to create models using:- linear regression,polynomial regression,logistic regression,KNN,neural networks).And also knowing how to train them (using gradient descent,backpropagation etc).
* Some basic knowledge of statistics(for feature scaling) and differential calculus(for the gradient descent and backpropagation algorithms)

**Implementation Details Of the Project-**

We would implement different algorithms on the training dataset such as linear

regression,polynomial regression,logistic regression,KNN,neural networks and present it in the

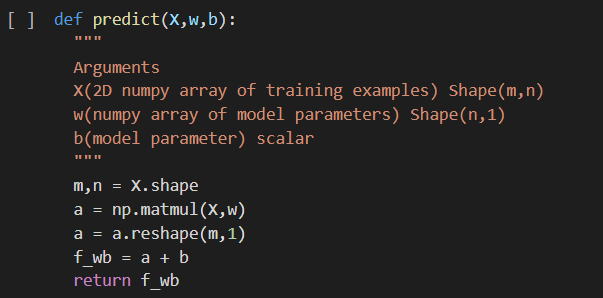
form of a report(Also convert it into a library if time permits).

**Linear regression:-**

**Model:-**

w(weight) and b(bias) are parameters

The predict function of my linear regression return the value fwb for all the training examples:-



**(I have used matrix multiplication to multiply all features with their respective weights**

**And added bias with the help of broadcasting).**

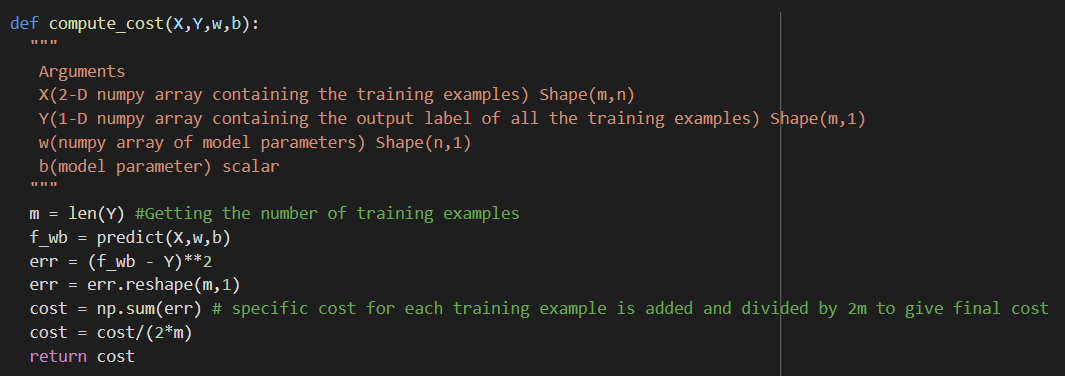
**Cost function:-**

We calculate the loss of each data point to measure how far our model is from the training

example.Then we sum all the losses to create the cost function.

In actual implementation we use the vectorized version to implement mean square error cost.

represent the ith training example from our training dataset



**(I have used the mean-square-error metric(As shown above) to measure the cost of my**

**model.The code is vectorized,for faster implementation so that the cost of each training**

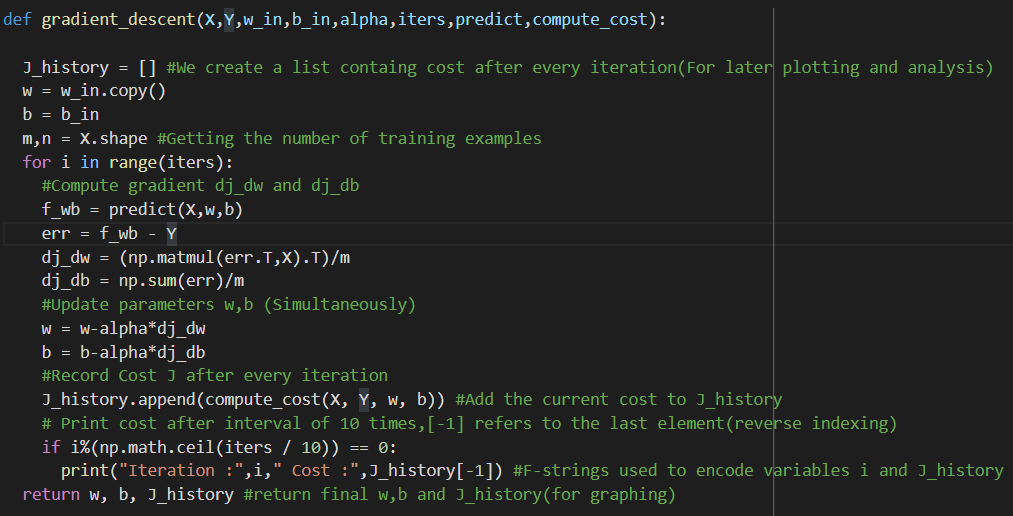
**example is calculated all at once.)**

**Training(By gradient descent):-**

is the learning rate.We simultaneously update w,b after each iteration. To update

Simultaneously we make a copy of w parameter,and after every iteration we store it in a list,

J\_train.



**Polynomial regression:-**

The implementation is almost same as shown for linear regression just the model is tweaked a

little bit to account for higher powers of x.

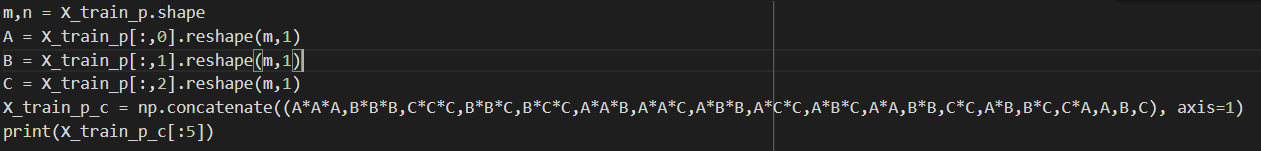
**Model:-**

More higher powers can also be used.Note the regularization parameter may be introduced if

required.

In the model,the predict function and gradient descent is same as linear regression but we

generate multiple terms for a 3 degree polynomial.

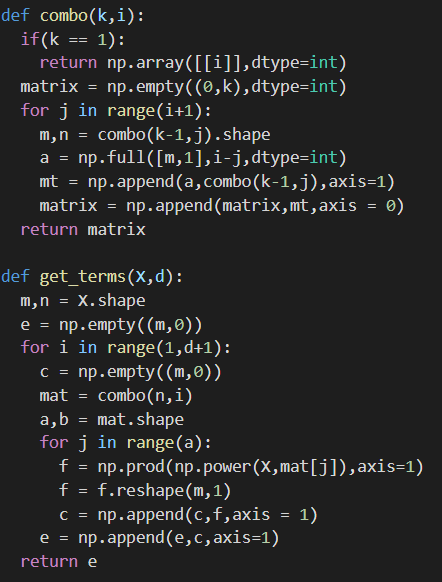


(All the different terms are concatenated along with A,B,C in one matrix of size (m,19) where m

Is the number of training examples).

I have also Implemented an algorithm to generalize this process for n degree polynomial, known

as get\_terms().



The function get\_terms() makes use of another function known as combo() which gives the

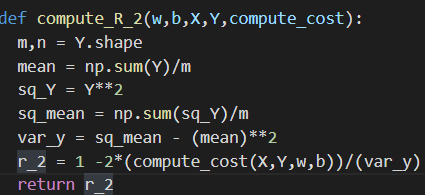
Pairs that satisfy the equation:-

Where a is a whole number.The pairs satisfying this equation are then used to generate the

Polynomial terms.

Minimum cost is obtained at n=5 degree polynomial

**Computing R2 score:-**

****

**Formula of r2 score is:-**

**Which is then simplified to**

**R\_2 = 1-2\*(mse\_cost/variance\_of\_y)**

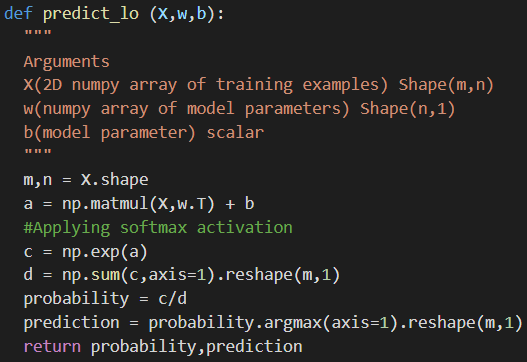
**Logistic regression:-**

Here we use the sigmoid function to adress classification problems.In such problems the input is

Some real number and the output can be represented as a 0 or 1.

**I have to deal with multiple classes so I have used softmax activation.**

**Model:-**

****

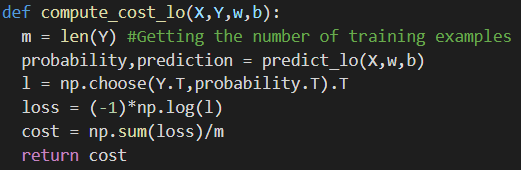
Here we use a different kind of loss function to penalize the model(Known as cross-entropy loss).

**Loss function:-**

**where j=Y[i]**

**Cost function:-**

**Compute cost for logistic:-**

****

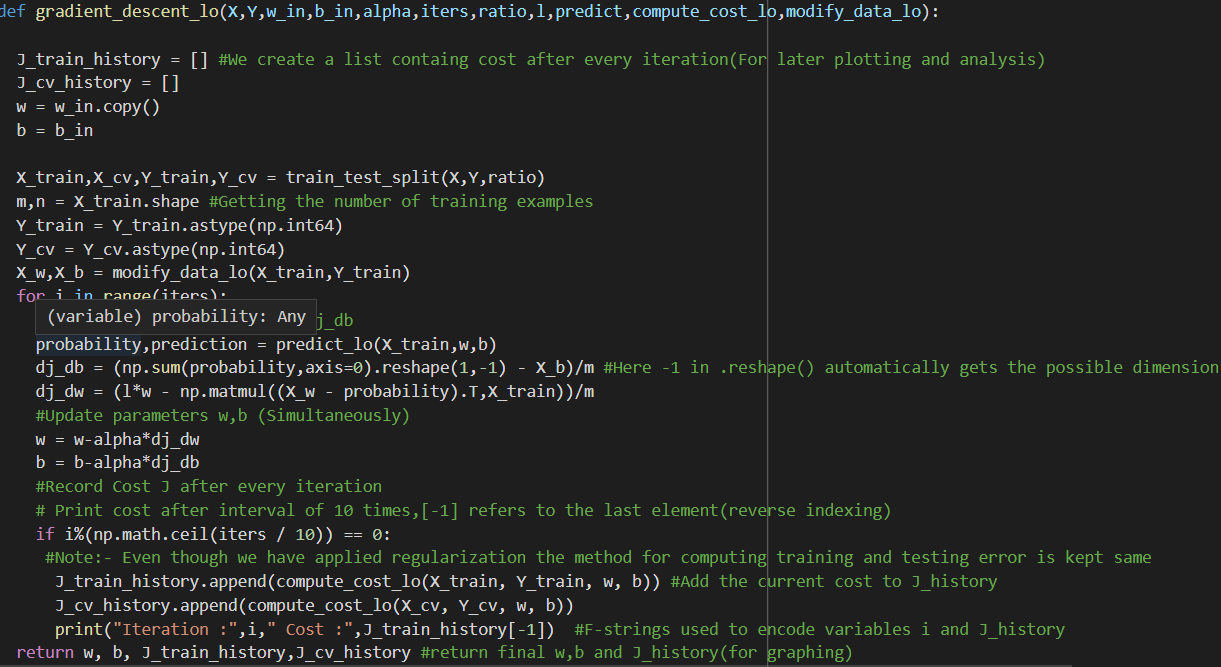
**Training(By gradient descent):-**

Here Yi is the one-hot encoded version of the output classes

Eg:- if initially Yi=np.array([[1],[0],[2]]) then after one-hot encoding

Yi = np.array(([0,1,0],[1,0,0],[0,0,1]))

**Gradient descent for logistic:-**

****

**The gradient-descent used here is vectorized inorder to achieve faster results.**

**Neural Network:-**

We have multiple layers of neuron in a neural network.The input is 1st propagated through the

Input layer.Each neuron takes the activation of all neurons in it’s previous layer as input and

Uses it to compute its own activation using its activation function.The activation function of all

neurons(except the output neuron) are Relu function.The activation function of output layer is

Chosen according to the requirement of the problem.

The neural network computes the output by forward propogation.

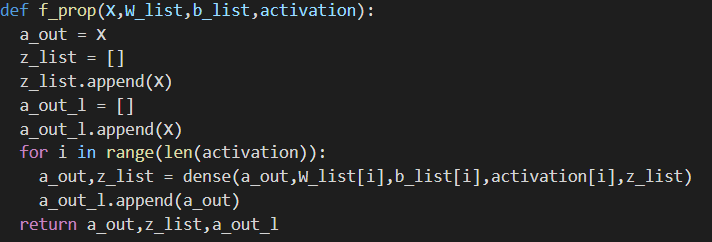
**Forward propagation:-**

Above equation represents the activation of the jth neuron in the Lth layer of the neural network.

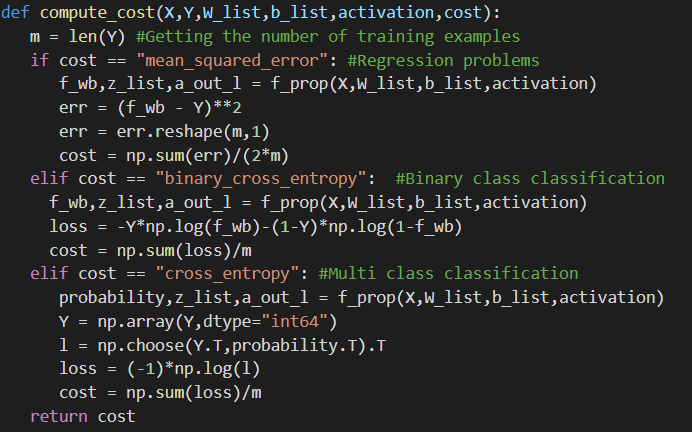
Here g(x) represents the activation function of the neuron.

We have to recursively pass the output through each layer of the neural network.Which is

Accomplished by forward propagation.

****

**Computing cost of neural network:-**

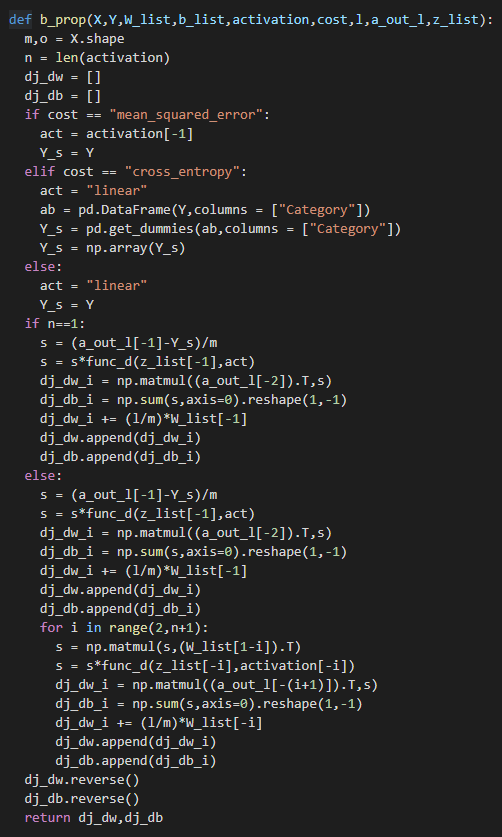
****

The cost function depends on the kind of model being trained.Eg:-For regression problems the

Cost function is generally mean\_squared\_error.

**Training(By gradient descent):-**

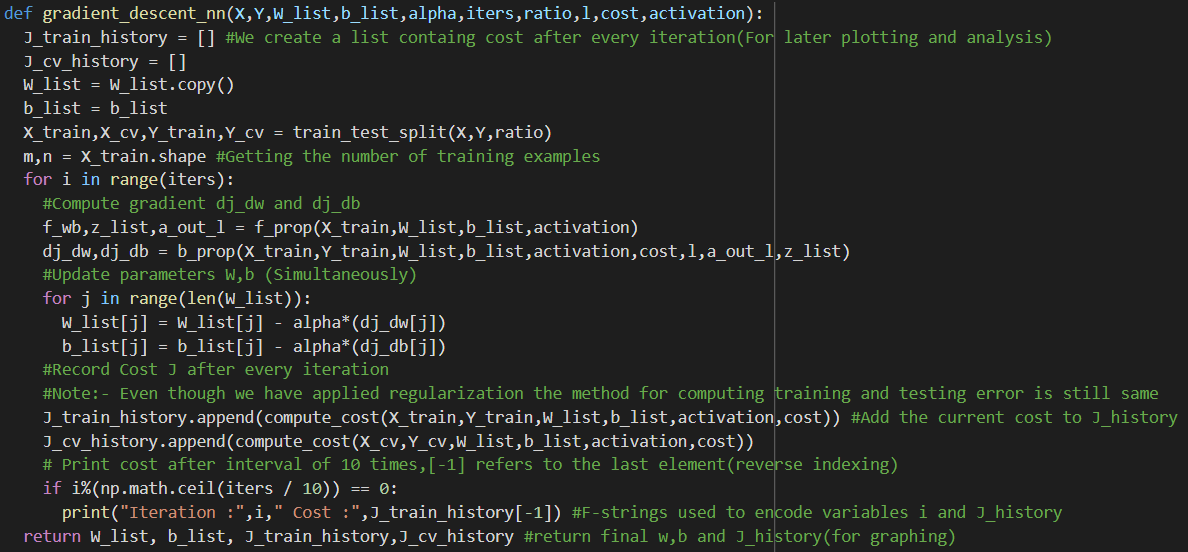
**The updation of weights and biases is done using the back-propagation algorithm:-**

****

I have created a list dj\_dw,dj\_db which stores all the updates for the weights and biases.The list

Is then reversed so that during update each dj\_dw(j) matches with its corresponding w(j).

**Gradient descent algorithm for neural network:-**

****

**KNN:-**

First we find the distance of the k nearest neighbors from the point,using the formula:-

Here x,y are the data points and and represent the ith feature of x and y.

Note:- In actual implementation square root is ignored,to speed up computation

**For classification:-**

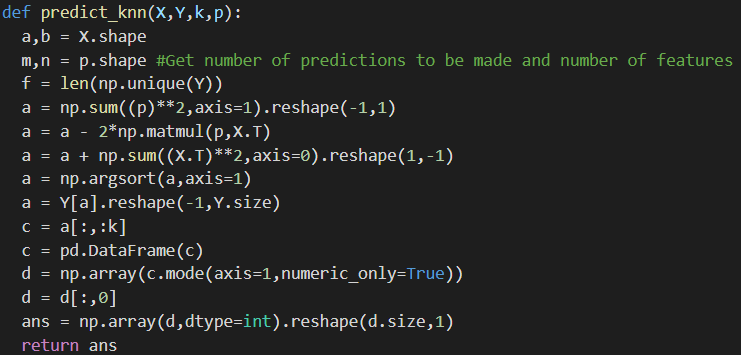
We classify the label by seeing the class which is most common to its k nearest neighbors.

**For regression:-**

We return the output as an average of its k nearest neighbors

In the project we use knn to make a prediction using the

classification dataset.



In the predict function we take the (a,b) matrix of X.(a is number of training examples,and b

Is number of features)and the (m,n) matrix p(where m is number of predictions and n is

Number of features).Also b=n,Using these 2 matrices we get the matrix a of shape (a,m),which

Stores the distances of each prediction from each training example.Then we sort the matrix and

See the first the first k columns to base out prediction on(This is achieved using the pandas

Function for mode).

**Development process:-**

Just creating a model and training it on the training data(Using gradient descent) is not enough

to ensure a successful model.For example we may need to increase/decrease the learning rate

to an optimum level so that training of model is not too slow and also ensure our model is not

overshooting.So to effectively develop our model we split our training data into

two parts and .After training our data on we test it on .

**By using and we draw the following conclusions:-**

* If and both are high the model has **high bias**(model is underfitting)
* If is low but is high model has **high variance**(model is overfitting)

**Flowchart for dealing with bias and variance:-**

****

Note here is the regularization parameter.This parameter is introduced in the cost function to

avoid the problem of overfitting(**High variance**).Regularization keeps the values of all the

Weights low in order to avoid overfitting.

**Implementation:-**

As you can see an additional regularization term is introduced in the cost function .During

gradient descent we consider the regularized cost function.

**Feature engineering(If required):-**

We could add/remove some features to adress underfitting and overfitting.If the range of data

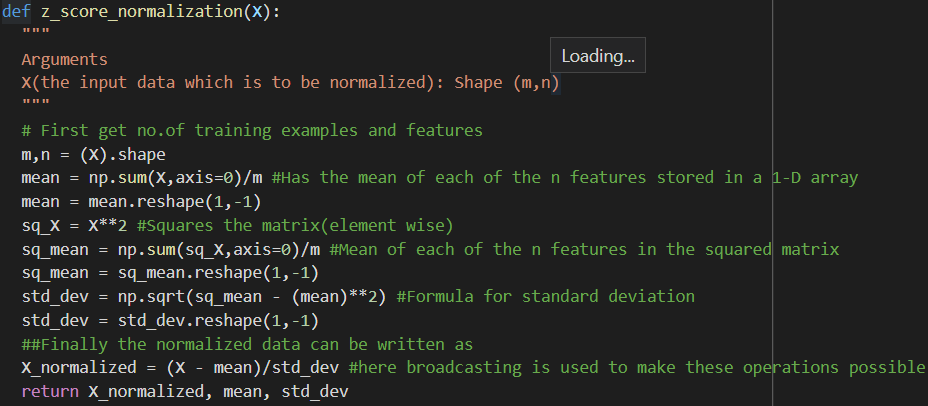
varies a lot among given features we could normalize our data for faster and more accurate

results.

**Z-score normalization:-**

Here z is the normalized data,x is the original data, is the mean of the data and is the

Standard deviation of the data.



z\_score\_normalization is applied in all algorithms(except of knn).This is done so that all

Features belong in the same range so that it can lead to faster gradient descent.

**About Me:**

* **Personal Details**

Name:- Anant Upadhyay

Branch:- 1st year,B-tech in Computer Science and Engineering

Admission No:- 22JE0110

D.O.B:- 24/11/2004

* 1 star coder in codechef :- <https://www.codechef.com/users/anant_up11>
* Made a project on the game tic tac toe:- <https://github.com/Anant-Upadhyay/Tictactoe>
* Github profile:-<https://github.com/Anant-Upadhyay>
* Completion of 1st course on Coursera:- <https://coursera.org/share/38b6e1ab1e5d6b6ef796f0df070c337b>
* Completion of 2nd course on Coursera:-

<https://www.coursera.org/account/accomplishments/certificate/LVZBLU7NC9NP>

* **Why should I be selected for this Cyberlabs?**

I am a keen learner with a childhood interest in machine learning.I look forward for pursuing a career in machine learning and artificial intelligence as these topics have always fascinated me.Being part of Cyberlabs will allow me to explore these fields deeper and explore topics like Computer vision,NLP,Supervised Learning,Unsupervised Learning etc.Being in cyberlabs will provide me with a team of like-minded people which will help in my personal growth,as well as the growth of the cyberlabs team.I would also like to participate in hackathons and discuss research papers on machine learning as I have always been fascinated by machine learning.