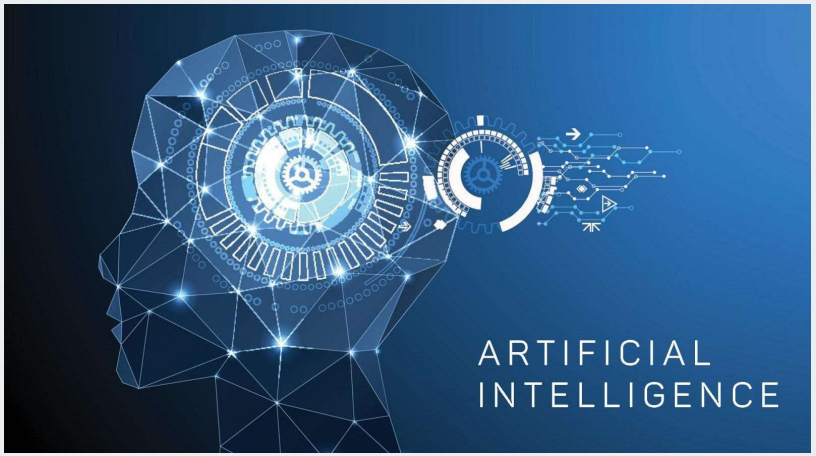
Artificial Intelligence

Project-Stock market analysis using ***Machine Learning*** and ***Artificial Intelligence***

-AYUSH JADIA

What is Artificial intelligence?

Artificial intelligence (**AI**) is intelligence—perceiving, synthesizing, and inferring information—demonstrated by machines, as opposed to intelligence displayed by humans or by other animals. "Intelligence" encompasses the ability to learn and to reason, to generalize, and to infer meaning. You can say that AI is implemented in machines to perform tasks that actually require human intelligence



How can it help?

Artificial intelligence and its applications are endless. A few of them are listed below:

1.Personalized Online Shopping, 2.Smart Cars,

3. Marketing Enhanced Images, 4.Social Media Surveillance, 5.Agriculture, 6.Customer Service,

7.Video Games, 8.Healthcare,

9.Banks, 10.Smart Homes,

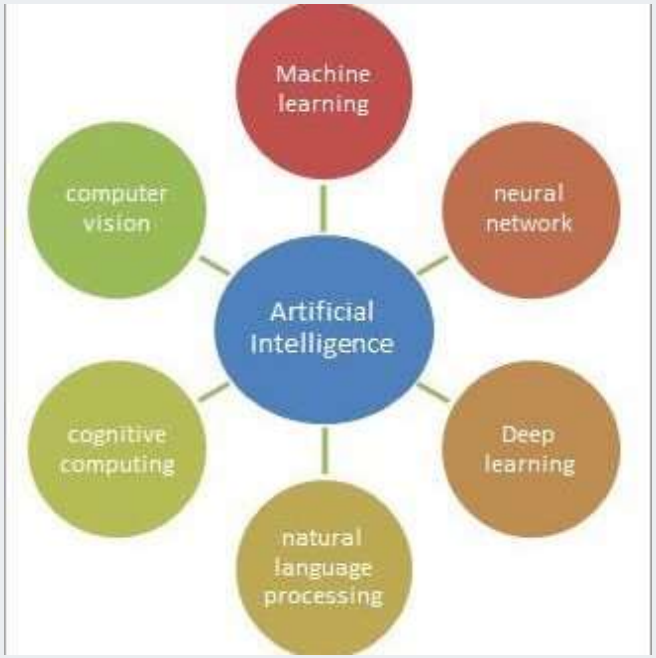
11.Virtual Assistance, 12.Space Exploration,

13.Chatbots.

We will be using AI in our project to predict the stock market.

Parts of AI

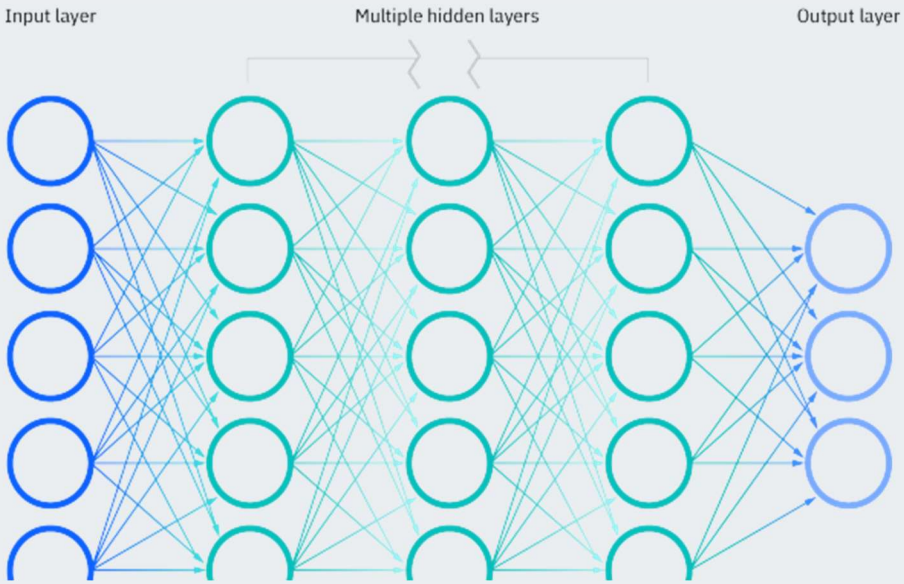
AI can be divided into many parts the main fields are deep learning ,machine learning, neural networks, natural language processing, computer vision, cognitive computing, natural language processing. The most widely used parts are deep learning, nueral network and machine learning . These three will go hand in hand during our project. Lets get a brief introduction about them one by one.



Nueral Network

Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

Neural networks rely on training data to learn and improve their accuracy over time. However, once these learning algorithms are fine-tuned for accuracy, they are powerful tools in computer science and artificial intelligence, allowing us to classify and cluster data at a high velocity.



Types of neural networks:

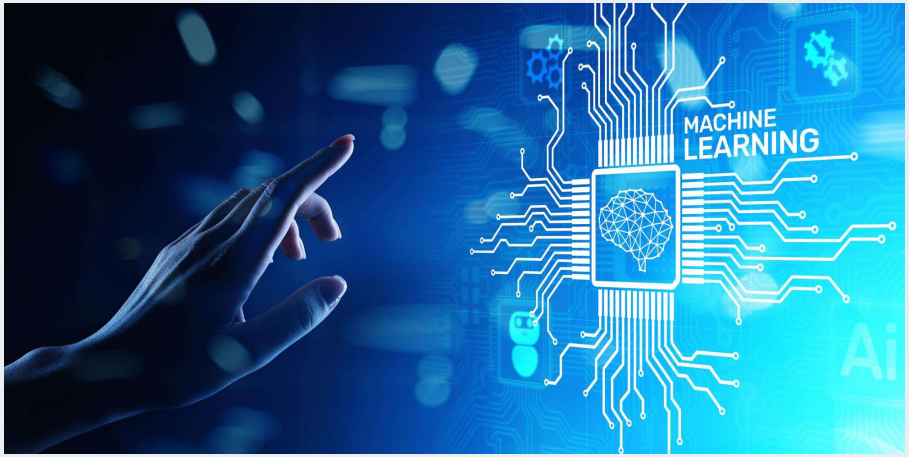
there are different types of neural networks to address specific problems or datasets. Prominents ones are:

Convolutional neural networks (CNNs), used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.

Recurrent neural network (RNNs) are typically used in natural language and speech recognition applications as it leverages sequential or times series data.

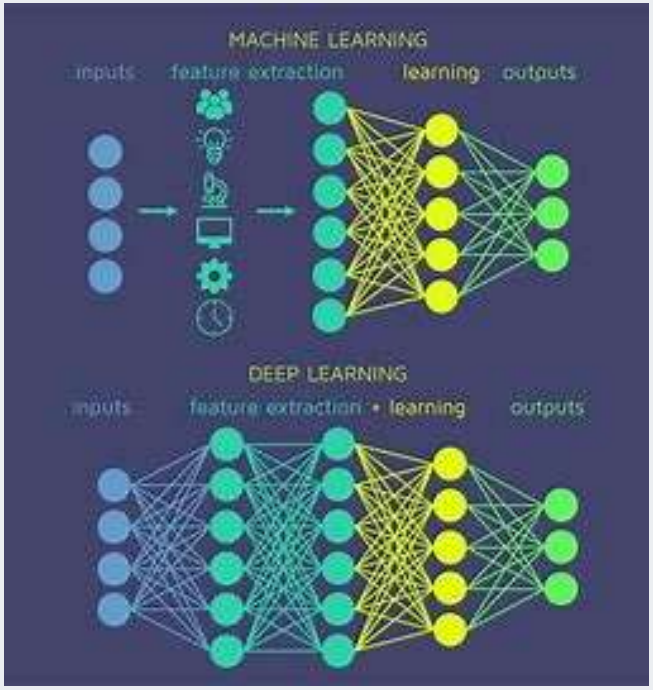
Machine learning

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning is more dependent on human intervention to learn. Human experts determine the set of features to understand the differences between data inputs, usually requiring more structured data to learn.



Deep learning

Deep learning is a essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. The way in which deep learning and machine learning differ is in how each algorithm learns. "Deep" machine learning can use labeled datasets, also known as supervised learning, to inform its algorithm, but it doesn’t necessarily require a labeled dataset. Deep learning can ingest unstructured data in its raw form (e.g., text or images), and it can automatically determine the set of features which distinguish different categories of data from one another



How deep learning works :

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible* layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate

❖ Artificial neural network

Perceptron:

➢ Basic introduction:

Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron. A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time. Perceptron is a type of artificial neural network, which is a fundamental concept in machine learning.

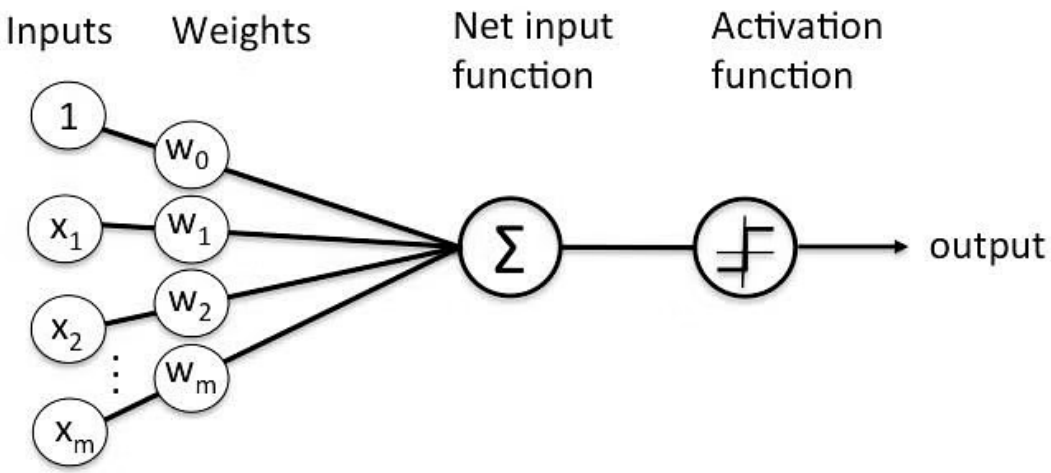
➢ Basic Components of Perceptron

The basic components of a perceptron are:

1. Input Layer: The input layer consists of one or more input neurons, which receive input signals from the external world or from other layers of the neural network.
2. Weights: Each input neuron is associated with a weight, which represents the strength of the connection between the input neuron and the output neuron.
3. Bias: A bias term is added to the input layer to provide the perceptron with additional flexibility in modeling complex patterns in the input data.
4. Activation Function: The activation function determines the output of the perceptron based on the weighted sum of the inputs and the bias term. Common activation functions used in perceptrons include the step function, sigmoid function, and ReLU function.
5. Output: The output of the perceptron is a single binary value, either 0 or 1, which indicates the class or category to which the input data belongs.
6. Training Algorithm: The perceptron is typically trained using a supervised learning algorithm such as the perceptron learning algorithm or

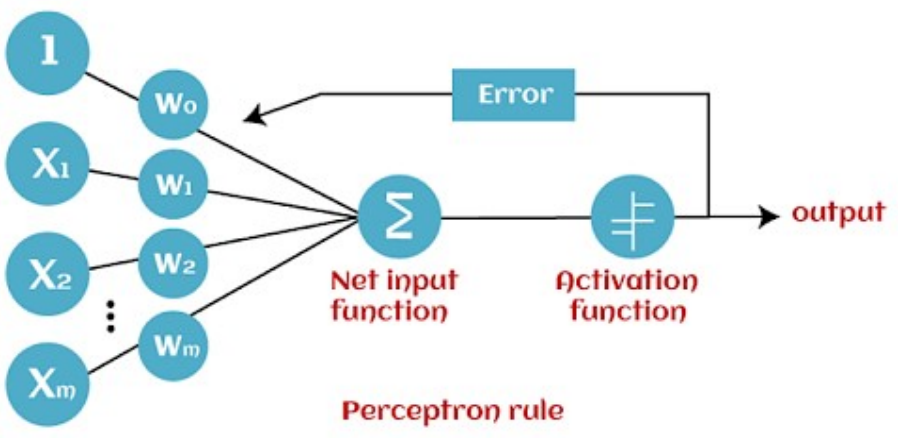
backpropagation. During training, the weights and biases of the perceptron are adjusted to minimize the error between the predicted output and the true output for a given set of training examples.

7. Overall, the perceptron is a simple yet powerful algorithm that can be used to perform binary classification tasks and has paved the way for more complex neural networks used in deep learning today.



How Does Perceptron Work?

Perceptron is considered a single-layer neural link with four main parameters. The perceptron model begins with multiplying all input values and their weights, then adds these values to create the weighted sum. Further, this weighted sum is applied to the activation function ‘f’ to obtain the desired output. This activation function is also known as the step function and is represented by ‘f.’



This step function or Activation function is vital in ensuring that output is mapped between (0,1) or (-1,1). Take note that the weight of input indicates a node's strength. Similarly, an input value gives the ability the shift the activation function curve up or down.

Step 1: Multiply all input values with corresponding weight values and then add to calculate the weighted sum. The following is the mathematical expression of it:

$\sum wi*xi = x1*w1 + x2*w2 + x3*w3+\ldots\ldots x4*w4$

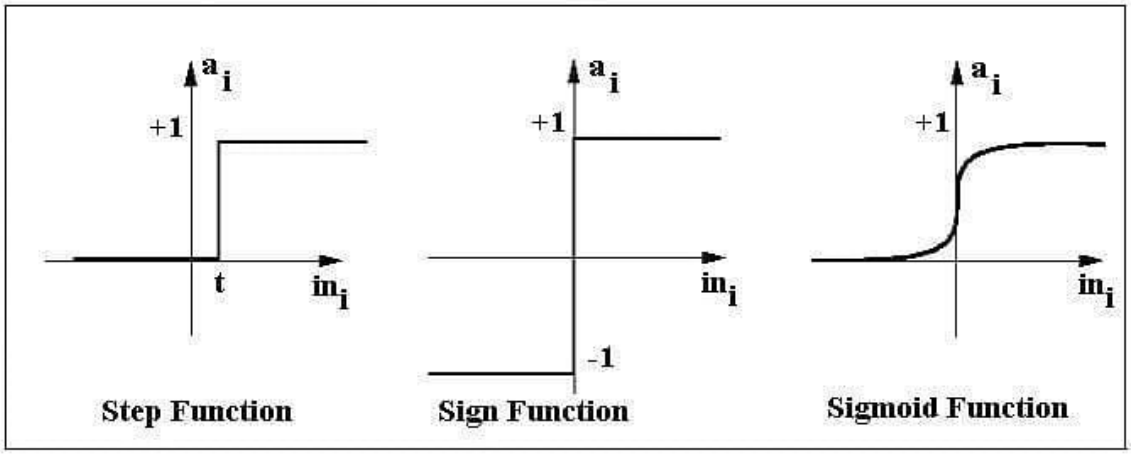
Add a term called bias 'b' to this weighted sum to improve the model's performance.

Step 2: An activation function is applied with the above-mentioned weighted sum giving us an output either in binary form or a continuous value as follows:

$Y=f(\sum wi*xi + b)$

Activation Functions of Perceptron

The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.



For example:

If ∑ wixi> 0 => then final output “o” = 1.

Else, final output “o” = -1.

Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

➢ Use of perceptron:

Mostly preceptron is used in decision making where the output can be true or false. Now perceptrons are able to differentiate between linearly differentiable classes/data only. Since by the algorithm we can see that the we can assign 0(false) and 1(true) depending on the output which is nothing but

∑wi*xi where I foes from 0 to n where w0=1 and x0=b (bias)

Now you can see that this equation is a line in 2-d plane(n=2) plane in 3-d(n=3) and hyperplane for n>=4.

Therefore this perceptron works efficiently when the data is linearly differentiable.

• Loss functions:

The loss function estimates how well a particular algorithm models the provided data. Loss functions are classified into two classes based on the type of learning task

- **Regression Models:** predict continuous values.
- **Classification Models:** predict the output from a set of finite categorical values.

REGRESSION LOSSES

1.) **Mean Squared Error (MSE) / Quadratic Loss / L2 Loss**

- It is the Mean of Square of Residuals for all the datapoints in the dataset. Residuals is the difference between the actual and the predicted prediction by the model.
- Squaring of residuals is done to convert negative values to positive values. The normal error can be both negative and positive. If some positive and negative numbers are summed up, the sum maybe 0. This will tell the model that the net error is 0 and the model is performing well but contrary to that, the model is still performing badly. Thus, to get the actual performance of the model, only positive values are taken to get positive, squaring is done.
- Squaring also gives more weightage to larger errors. When the cost function is far away from its minimal value, squaring the error will penalize the model more and thus helping in reaching the minimal value faster.
- Mean of the Square of Residuals is taking instead of just taking the sum of square of residuals to make the loss function independent of number of datapoints in the training set.
- MSE is sensitive to outliers.

(1)

- Python3

```
import numpy as np
```

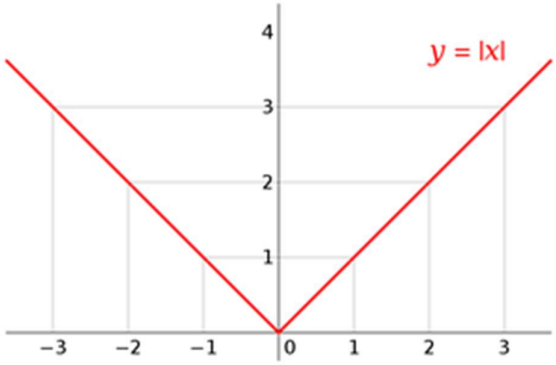
```
# Mean Squared Error

def mse( y, y_pred ) :

    return   np.sum( ( y - y_pred ) ** 2 ) / np.size( y )
```

where,

```
i         - ith training sample in a dataset
n         - number of training samples
y(i)      - Actual output of ith training sample
y-hat(i) - Predicted value of ith training sample
```

2.) **Mean Absolute Error (MAE) / La Loss**

- It is the Mean of Absolute of Residuals for all the datapoints in the dataset. Residuals is the difference between the actual and the predicted prediction by the model.
- The absolute of residuals is done to convert negative values to positive values.
- Mean is taken to make the loss function independent of number of datapoints in the training set.
- One advantage of MAE is that is robust to outliers.
- MAE is generally less preferred over MSE as it is harder to calculate the derivative of the absolute function because absolute function is not differentiable at the minima.



Source: Wikipedia

(2)

- Python3

```
# Mean Absolute Error


def mae( y, y_pred ) :


    return np.sum( np.abs( y - y_pred ) ) / np.size( y )
```

Output

3.) **Mean Bias Error:** It is the same as **MSE**. but less accurate and can could conclude if the model has a positive bias or negative bias.

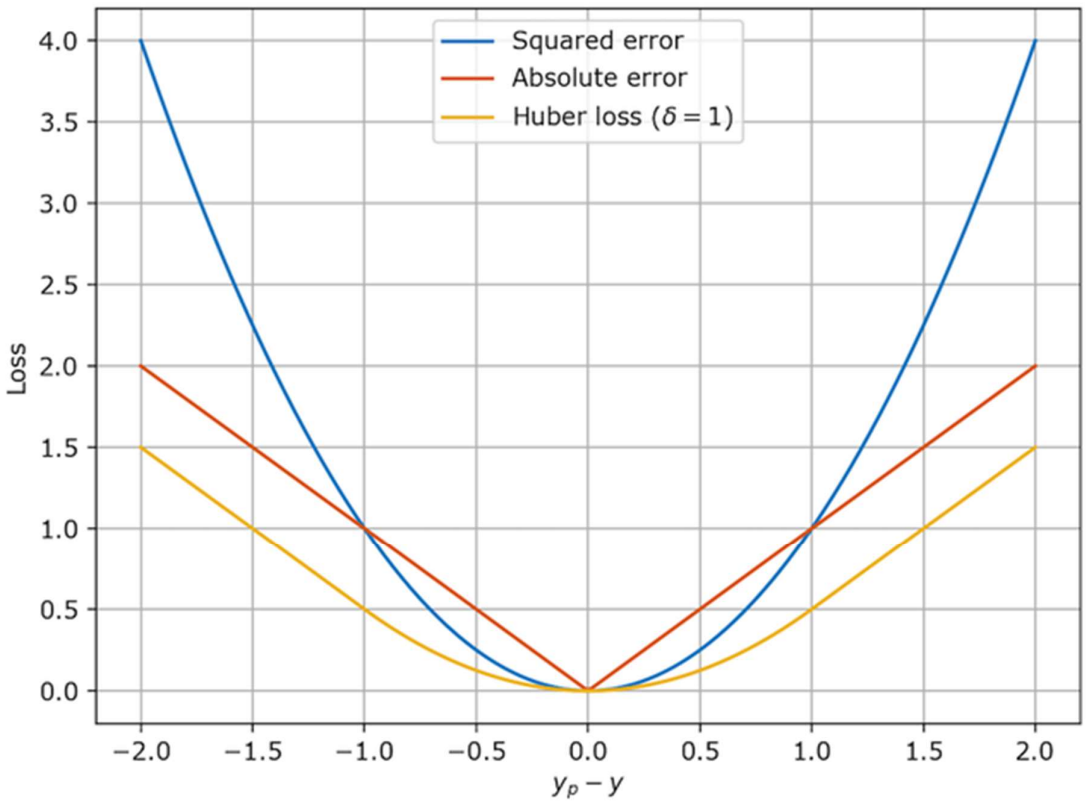
(3)

- Python3

```
# Mean Bias Error


def mbe( y, y_pred ) :


    return np.sum( y - y_pred ) / np.size( y )
```

4.) **Huber Loss / Smooth Mean Absolute Error**

- It is the combination of MSE and MAE. It takes the good properties of both the loss functions by being less sensitive to outliers and differentiable at minima.
- When the error is smaller, the MSE part of the Huber is utilized and when the error is large, the MAE part of Huber loss is used.
- A new hyper-parameter '**????**' is introduced which tells the loss function where to switch from MSE to MAE.



- Python3

```
def Huber(y, y_pred, delta):

    condition = np.abs(y - y_pred) < delta

    l = np.where(condition, 0.5 * (y - y_pred) ** 2,
                 delta * (np.abs(y - y_pred) - 0.5 * delta))

    return np.sum(l) / np.size(y)
```

CLASSIFICATION LOSSES

Cross-Entropy Loss: Also known as **Negative Log Likelihood**. It is the commonly used loss function for classification. Cross-entropy loss progress as the predicted probability diverges from the actual label.

- Python3

```
# Binary Loss



def cross_entropy(y, y_pred):


    return - np.sum(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))
/ np.size(y)
```

5.) **Hinge Loss**: Also known as Multi-class **SVM Loss**. Hinge loss is applied for maximum-margin classification, prominently for support vector machines. It is a convex function used in the convex optimizer.

- Python3

```
# Hinge Loss



def hinge(y, y_pred):


    l = 0


    size = np.size(y)


    for i in range(size):


        l = l + max(0, 1 - y[i] * y_pred[i])


    return l / size
```

We can define all of the loss function in general way as:

Given a set of training examples $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbf{R}^m$ and $y_i \in \mathcal{R}$ ($y_i \in -1, 1$ for classification), our goal is to learn a linear scoring function $f(x) = w^T x + b$ with model parameters $w \in \mathbf{R}^m$ and intercept $b \in \mathbf{R}$. In order to make predictions for binary classification, we simply look at the sign of $f(x)$. To find the model parameters, we minimize the regularized training error given by

$$E(w, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w)$$

where L is a loss function that measures model (mis)fit and R is a regularization term (aka penalty) that penalizes model complexity; $\alpha > 0$ is a non-negative hyperparameter that controls the regularization strength.

Different choices for L entail different classifiers or regressors:

- Hinge (soft-margin): equivalent to Support Vector Classification. $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))$.
- Perceptron: $L(y_i, f(x_i)) = \max(0, -y_i f(x_i))$.
- Modified Huber: $L(y_i, f(x_i)) = \max(0, 1 - y_i f(x_i))^2$ if $y_i f(x_i) > 1$, and $L(y_i, f(x_i)) = -4 y_i f(x_i)$ otherwise.
- Log Loss: equivalent to Logistic Regression. $L(y_i, f(x_i)) = \log(1 + \exp(-y_i f(x_i)))$.
- Squared Error: Linear regression (Ridge or Lasso depending on R). $L(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$.
- Huber: less sensitive to outliers than least-squares. It is equivalent to least squares when $|y_i - f(x_i)| \leq \varepsilon$, and $L(y_i, f(x_i)) = \varepsilon |y_i - f(x_i)| - \frac{1}{2}\varepsilon^2$ otherwise.
- Epsilon-Insensitive: (soft-margin) equivalent to Support Vector Regression. $L(y_i, f(x_i)) = \max(0, |y_i - f(x_i)| - \varepsilon)$.

Resources Used :

(1) IBM newsletters
(2) GeeksforGeeks
(3) https://scikit-learn.org/stable/

Tentative deadlines:

(1) Week8 (till July 2):
ANN ,MLP,
(2) Week 9 (till July 9):
backpropogation ,Keras tuner
(3) Week 10 (till July 13)
CNN,RNN.
(4) Week 11(till July 14):
Completion of project testing and debugging.