Machine Learning Assignment - 2 report

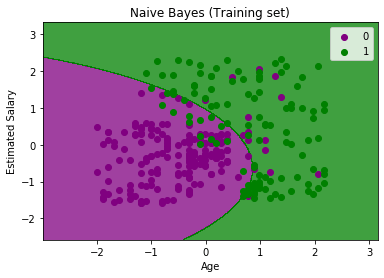
*By:*

*Tushar Dey Biswas (ID: 2020A4PS1987H)*

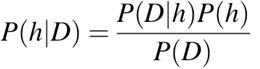
*Anushtup Nandy (ID: 2020A4PS1981H)*

*Sarvaswa Mohata (ID: 2020B1AA2358H)*

## ***NAIVE BAYES:***



Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets. Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features.This assumption is called class conditional independence.



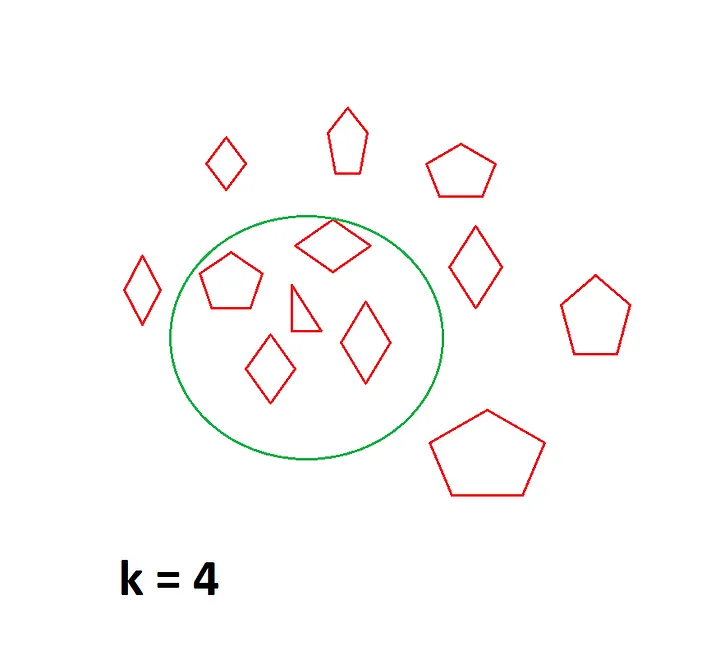
Naive Bayes classifier calculates the probability of an event in the following steps:

* **Step 1**: Calculate the prior probability for given class labels
* **Step 2**: Find Likelihood probability with each attribute for each class
* **Step 3**: Put these values in Bayes Formula and calculate posterior probability.
* **Step 4**: See which class has a higher probability, given the input belongs to the higher probability class

***KNN:***

KNN is one of the simplest forms of machine learning algorithms mostly used for classification. It classifies the data point on how its neighbor is classified.

KNN classifies the new data points based on the similarity measure of the earlier stored data points. For example, if we have a dataset of tomatoes and bananas. KNN will store similar measures like shape and color. When a new object comes it will check its similarity with the color (red or yellow) and shape.



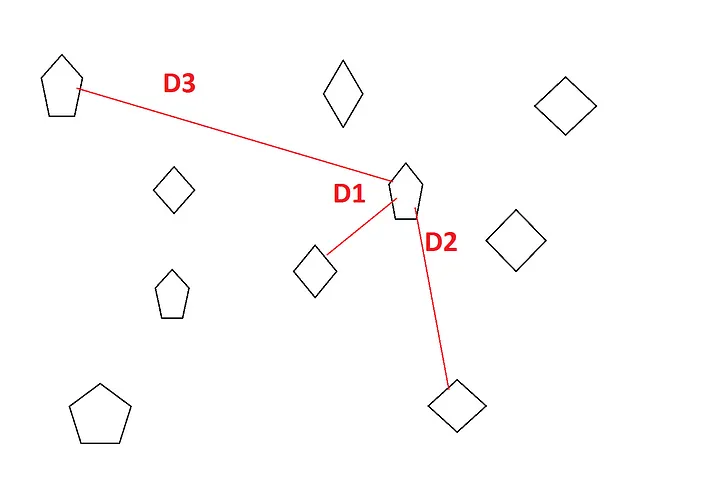
**When to USE KNN:**

1. We have properly labeled data. For example, if we are predicting someone is having diabetes or not the final label can be 1 or 0. It cannot be NaN or -1.

2. Data is *noise-free*. For the diabetes data set we cannot have a Glucose level as 0 or 10000. It’s practically impossible.

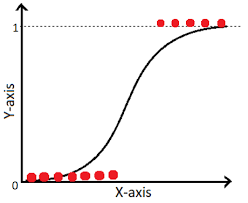
3.. Small dataset.

We usually use Euclidean distance to calculate the nearest neighbor. If we have two points (x, y) and (a, b).



***LOGISTIC REGRESSION:***

It is a classification technique used when the target variable is *categorical* (example 0 or 1).

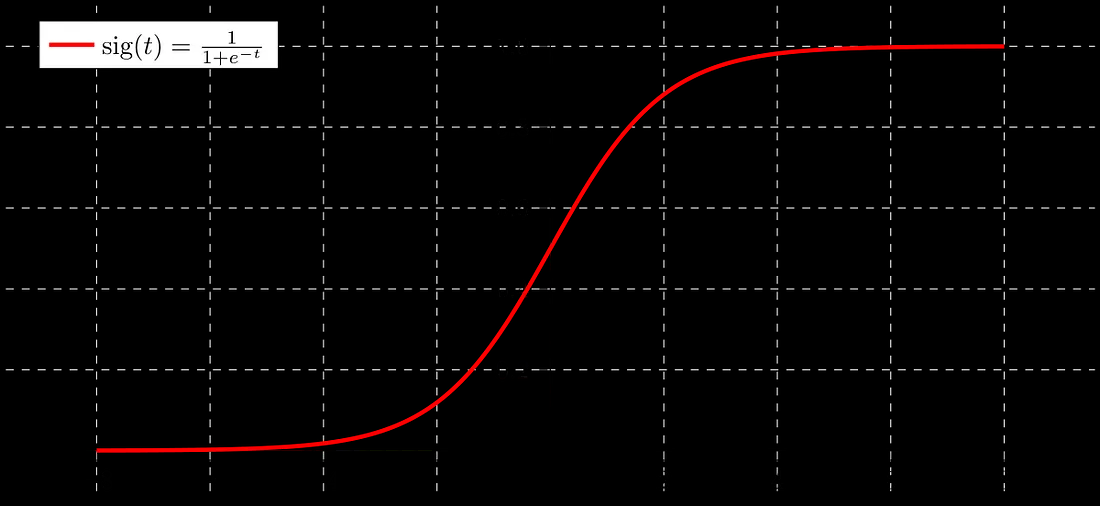


The math:

The hypothesis is ⇒ Z= Wx+B

The output is supposed to be one of 0 or 1

The probability is defined by ⇒ Sigmoid function:



If ‘Z’ goes to infinity, Y(predicted) will become 1 and if ‘Z’ goes to negative infinity, Y(predicted) will become 0.

The hypothesis’ output is considered as the estimated probability.This is used to infer how confident can predicted value be actual value when given an input X.

*- In mathematical format:*

h(x) = P(Y=1 | x) ⇒ probability of Y given x

and P(Y=0 | x) = 1- P(Y=1 | x)

Data is fit into linear regression model, which then be acted upon by a logistic function predicting the target categorical dependent variable.

*- Decision boundary:*

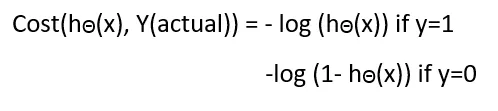
To predict which class a data belongs, a **threshold** can be set. Based upon this threshold, the obtained estimated probability is classified into classes.

If (y\_pred ≥ 0.5) ⇒ then classify 1 else 0.

Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary.

*- Cost Function:*

The cost function is decided by the NLL (or the negative log likelihood).



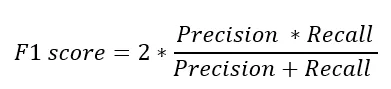
To converge this **gradient descent** is used ⇒ which will only converge if the function is a convex one!.

Negative function is because when we train, we need to maximize the probability by minimizing loss function. Decreasing the cost will increase the maximum likelihood assuming that samples are drawn from an identically independent distribution.

| Naive Bayes | Logistic | KNN |
| --- | --- | --- |
| 76.335 | 76.110 | 76.512 |
| 75.805 | 76.614 | 75.954 |
| 76.596 | 84.050 | 82.058 |
| 76.261 | 83.808 | 82.272 |
| 75.572 | 83.417 | 81.789 |
| 75.544 | 83.752 | 81.705 |
| 77.092 | 82.793 | 81.451 |
| 76.584 | 83.577 | 82.976 |
| 76.817 | 83.791 | 83.455 |
| 77.251 | 84.567 | 83.397 |

***F1 SCORE***

The F1 score is defined as the harmonic **mean of precision and recall**.



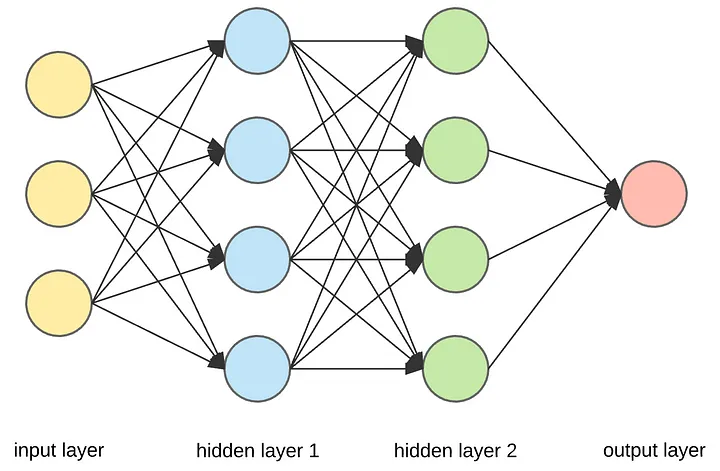
* A model will obtain a **high F1 score** if both Precision and Recall are high
* A model will obtain a **low F1 score** if both Precision and Recall are low
* A model will obtain a **medium F1 score** if one of Precision and Recall is low and the other is high.

***RESULTS***

We obtained an **accuracy score of 76.335%** and an **F1 score of 82.053%**. We applied the Laplace Smoothing technique which enhanced the accuracy score to **76.551%.** The smoothing technique applied basically improves the probability in case of a feature being absent or having no influence in the likelihood function estimate.

***ANN***

Artificial Neural Networks (ANN) are multi-layer fully-connected neural nets.They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers.

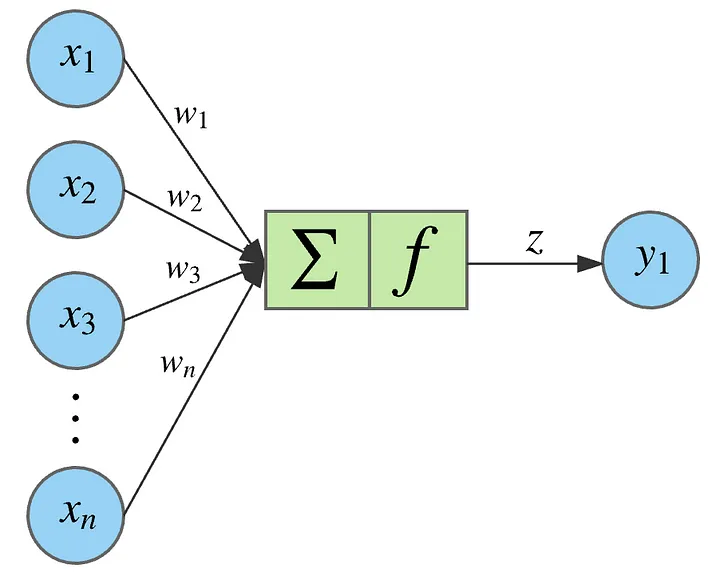


**Hidden Layer**

Then comes the hidden layer. There can be one or more hidden layers in a neural network. Neurons in a hidden layer receive their inputs either from the neurons of the input layer or from the neurons of the previously hidden layer. Each neuron then passes the input to another nonlinear activation function and post that, it sends the output to the next layer neurons. Here also, we multiply the data by edge weights as it is transmitted to the next layer.

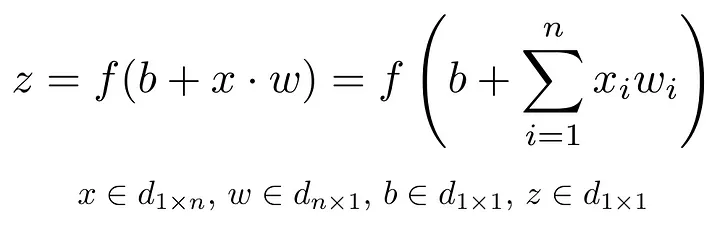
**Activation function**

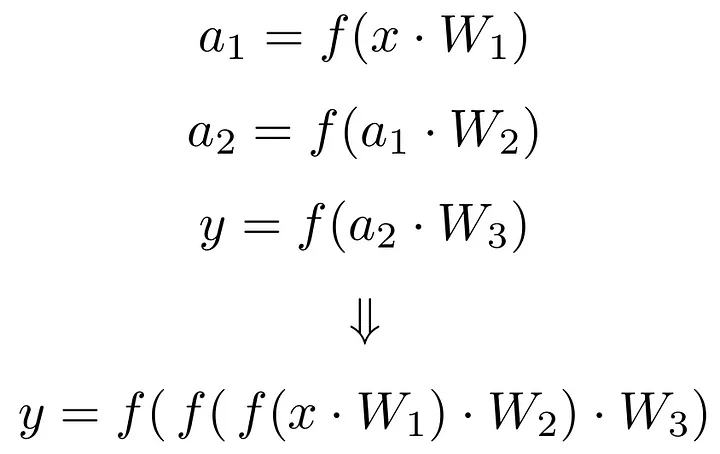
An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron’s input to the network is important or not in the process of prediction using simpler mathematical operations. In the given examplewe have used sigmoid, Relu , Tanh and softmax function as activation functions.In the output layer we should primarily use softmax function as the activation function.



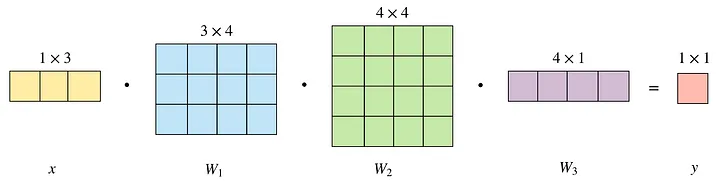
A given node takes the weighted sum of its inputs, and passes it through a non-linear activation function. This is the output of the node, which then becomes the input of another node in the next layer. The signal flows from left to right, and the final output is calculated by performing this procedure for all the nodes. Training this deep neural network means learning the weights associated with all the edges.

The weighted sum of its inputs passed through a non-linear activation function. It can be represented as a vector dot product, where *n* is the number of inputs for the node.It allows to shift the result of the activation function to the left or right. It also helps the model to train when all the input features are 0. If this sounds complicated right now you can safely ignore the bias terms.

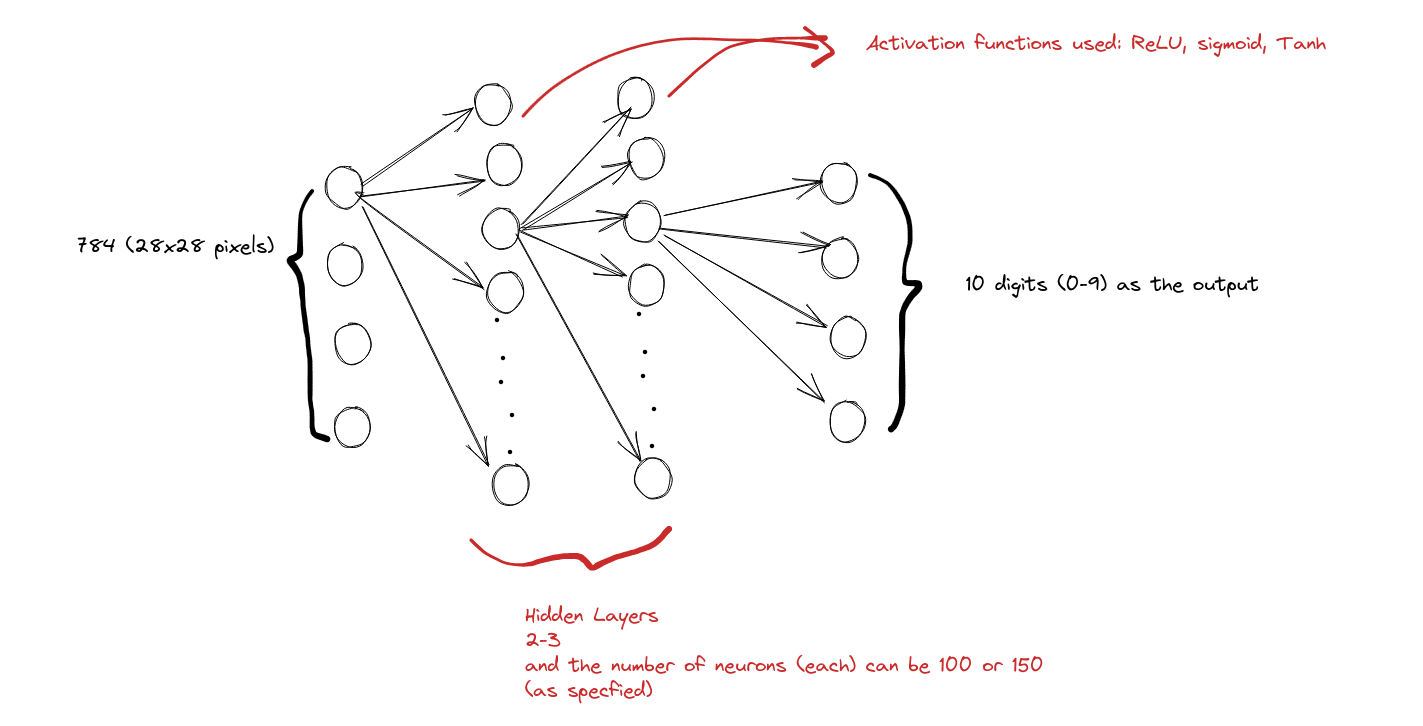




The entire thing can be intuitively looked upon as vector transformations of matrices:



***Our Model***:

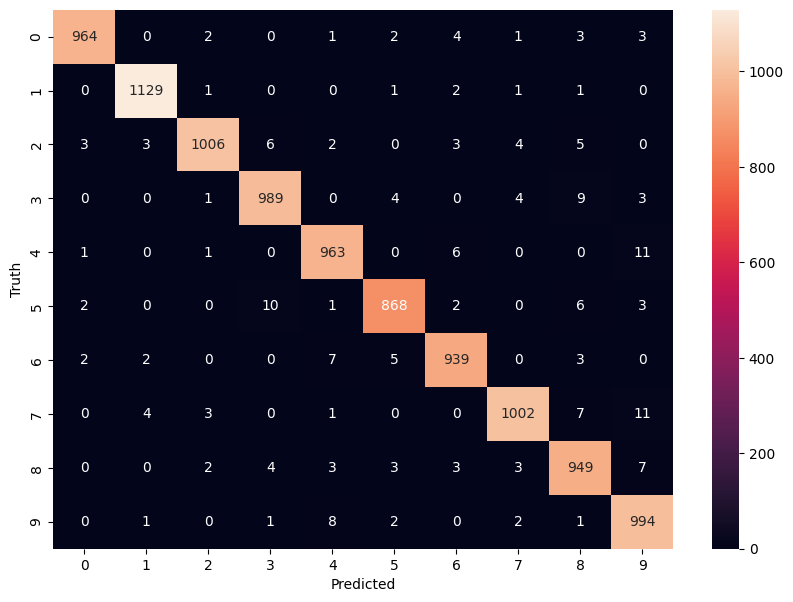


We have implemented 15 such models:

| SNo | Number of Hidden Layers | Neurons | Activation | Accuracy |
| --- | --- | --- | --- | --- |
| 1 | 2 | 150,100 | ReLU,ReLU sigmoid | 0.9726 |
| 2 | 2 | 100,100 | ReLU, tanh, sigmoid | 0.9802 |
| 3 | 2 | 100,150 | ReLU | 0.098 |
| 4 | 2 | 100,100 | tanH, tanh, Sigmoid | 0.975 |
| 5 | 2 | 100, 150 | Tanh,tanh ,sigmoid | 0.974 |
| 6 | 2 | 100,150 | Tanh, sigmoid sigmoid | 0.976 |
| 7 | 3 | 150, 150, 150 | Tanh x3, sigmoid | 0.974 |
| 8 | 3 | 150, 150, 150 | ReLU x3, sigmoid | 0.9778 |
| 9 | 3 | 150, 150, 150 | Sigmoid x3 | 0.974 |
| 10 | 3 | 150, 150, 100 | Sigmoid, relu, tanh, sigmoid | 0.972 |
| 11 | 3 | 150, 100, 100 | Sigmoid, tanh, tanh, softmax | 0.9714 |
| 12 | 3 | 150, 100, 100 | Sigmoid, ReLU, ReLU, softmax | 0.9714 |
| 13 | 3 | 100, 100, 100 | Tanh, tanh, tanh, softmax | 0.975 |
| 14 | 3 | 150, 100, 100 | Tanh, tanh, tanh, softmax | 0.974 |
| 15 | 3 | 150, 150, 150 | Tanh, ReLU, tanh, softmax | 0.976 |

Best performing model is: MODEL-2 with:

1. 2 hidden layers
2. 100 neurons per layer
3. Activation functions: ReLU, tanh, sigmoid
4. Accuracy of: 0.982



The model which performs poorest is MODEL 3. Since, we have used ReLU as the activation for the output and it is unable to scale itself to classify the outputs properly.

