Chapter 3 : part 5

**Naive Bayes**

Bayes Theorem, Naive Bayes Algorithm

**3.5.1 Bayes Theorem**

* Bayes' Theorem: Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the *probability of a hypothesis* with *prior* *knowledge*. It depends on the conditional probability. The formula for Bayes' theorem is given as:

|  |  |
| --- | --- |
|  | **Naïve Bayes Classifier Algorithm** |

* Probability that ***A*** occurs given ***B*** has occurred ***P(A|B)***. The Bayes Theorem is a method of finding what the ***probability is of something occurring*** (A) given that something ***else has just occurred*** (B).
* ***P(A|B) is Posterior probability:*** Probability of hypothesis **A** on the observed event **B**. How often **A** happens given that **B** happens
* ***P(B|A) is Likelihood probability:*** Probability of the evidence given that the probability of a hypothesis is true. How often ***B*** happens given that ***A*** happens
* ***P(A) is Prior Probability:*** Probability of hypothesis before observing the evidence. ***A*** is on its own.
* ***P(B) is Marginal Probability:*** Probability of Evidence. ***B*** is on its own.
* where and **a**re events and. A and B must be *different* *events*.
* is a conditional probability: The probability of *event* occurring given that is *true*. It is also called the *posterior* *probability* of given .
* is also a conditional probability: the probability of event occurring given that is true. It can also be interpreted as the likelihood of givena fixed because .
* and are the probabilities of observing and respectively without any given conditions; they are known as the marginal probability or prior probability.
* Example 1: Let us say ***P(Fire)*** means how often there is fire, and P(Smoke) means how often we see smoke, then:

***P(Fire|Smoke)*** means how often there is *fire* when we can see *smoke*

***P(Smoke|Fire)*** means how often we can see *smoke* when there is *fire*

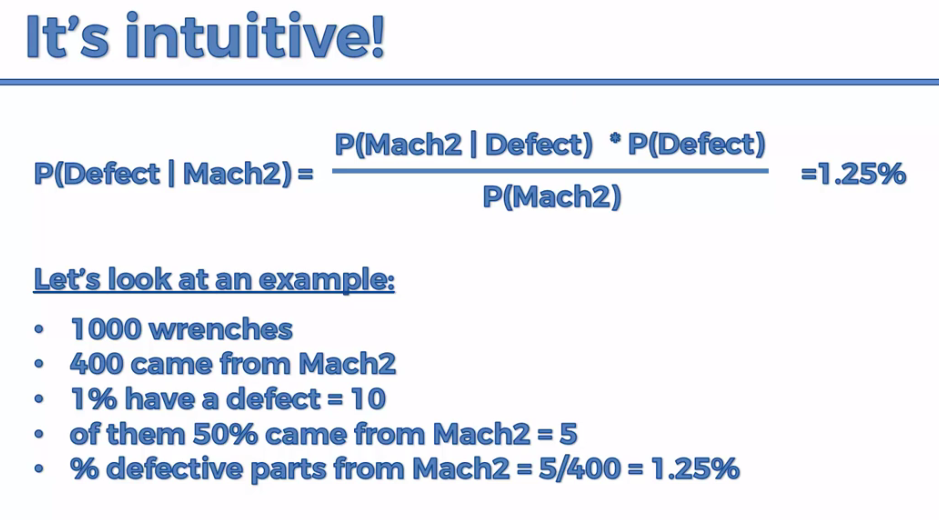
So the formula kind of tells us "forwards" **P(Fire|Smoke)** when we know "backwards" **P(Smoke|Fire)**

* dangerous fires are rare (1%), ***P(Fire)***
* but smoke is fairly common (10%) due to barbecues, ***P(Smoke)***
* and 90% of dangerous fires make smoke, ***P(Smoke|Fire)***

We can then discover the probability of dangerous Fire when there is Smoke:

So it is still worth checking out any smoke to be sure.

* Wrench making machine example: Two machines produce wrenches. We know that:
* Machine 1: Produces 30 Wrenches Per Hour
* Machine 2: Produces 20 Wrenches Per Hour
* *1%* of all wrenches are defective, where *50%* of the wrenches that are defective are from *machine 1* and *50%* from *machine 2*.
* Question: What is the *probability* that a *part* produced by *machine 2* is *defective*?
* Total wrench per hour is 30+20 = 50
* **P(M1)** = probability of being from *machine 1* = 30/50 = 0.6
* **P(M2)** = probability of being from *machine 2* = 20/50 = 0.4
* **P(D)** = probability of being defective = 0.01
* **P(M1 | D)** = Probability of being from *machine 1* from defected wrenches = 0.5, (probability of being from a pile of only defective wrenches).
* **P(M2 | D)** = Probability of being from *machine 2* from defected wrenches = 0.5, (probability of being from a pile of only defective wrenches).
* **P(D | M2)** = *probability* that a *Wrench* produced by *machine 2* is *defective*??? i.e. given that the wrench is from , then we need to find the probability of being it defective. (from pile of wrenches only come from machine 2)
* From Bayes Theorem,
* Its intuitive:



= Number of defective wrenches comes from machine 2

= Number of wrenches comes from machine 2

* **|** : means given in mathematical terms.
* Similarly, *probability* that a *Wrench* produced by *machine 1*  is *defective*

**3.5.2 Naïve Bayes Classifier Algorithm**

* *Naïve* *Bayes* algorithm is a *supervised* learning algorithm, which is based on *Bayes* *theorem* and used for solving *classification* problems (used to classify data with previous known classes).
* It is *mainly used* in *text classification* that includes a *high-dimensional training dataset*.
* *Naïve Bayes Classifier* is one of the simple and most effective *Classification* *algorithms* which helps in building the *fast machine learning models* that can make *quick* *predictions*.
* It is a *probabilistic classifier*, which means it *predicts* on the *basis of the probability* of an *object*.
* Some popular examples of *Naïve Bayes* Algorithm are *spam* filtration, *Sentimental* analysis, and classifying *articles*.
* Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

* Naïve: It is called *Naïve* because it assumes that the *occurrence* of a *certain* *feature* is ***independent of the occurrence of other features***. Such as if the fruit is identified on the bases of color, shape, and taste, then ***red***, ***spherical***, and ***sweet*** fruit is recognized as an Apple.
* Bayes: It is called *Bayes* because it depends on the principle of *Bayes' Theorem*.
* Advantages of Naïve Bayes Classifier:
* Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
* It can be used for Binary as well as Multi-class Classifications.
* It performs well in Multi-class predictions as compared to the other Algorithms.
* It is the most popular choice for text classification problems.
* Disadvantages of Naïve Bayes Classifier:
* Naive Bayes assumes that all features are independent or unrelated, so it *cannot learn the relationship* between *features*.
* Applications of Naïve Bayes Classifier:
* It is used for *Credit* *Scoring*.
* It is used in *medical* *data* *classification*.
* It can be used in *real-time predictions* because *Naïve Bayes Classifier* is an *eager learner*.
* It is used in *Text* *classification* such as *Spam filtering* and *Sentiment* analysis.
* Types of Naïve Bayes Model: There are three types of Naive Bayes Model, which are given below:
* Gaussian: The Gaussian model assumes that features follow a normal distribution. This means if *predictors take continuous values* instead of *discrete*, then the model assumes that these values are *sampled* from the *Gaussian* *distribution*.
* Multinomial: The Multinomial Naïve Bayes classifier is used when the *data* is *multinomial* *distributed*. It is primarily used for *document classification* problems, it means a particular *document* belongs to which *category* such as *Sports*, *Politics*, *education*, etc. The classifier uses the *frequency of words* for the predictors.
* Bernoulli: The *Bernoulli* classifier works similar to the *Multinomial* *classifier*, but the predictor variables are the *independent* *Booleans variables*. Such as if a particular word is present or not in a document. This model is also famous for *document* *classification* tasks.

|  |  |
| --- | --- |
| * How it Works:  1. Apply Bayes Theorem in this example we’ll use it to find the probability that a person walks based on his features (the specific age and salary of that data point). Calculate the probability of each of the components of Bayes Theorem. 2. Apply Theorem again, in this case it would be to find the probability that the new dataset Drives based on its features (age and salary). 3. Compare the two and assign a class to the dataset. | Bayes Theorem Example |

* Here we've got a data set. It has two features x1 (salary) and x2 (age). And there are two categories: *Category 1* which is *RED (Walks to work)* *Category 2* which is *GREEN (Drives to work)*.
* So we are presenting observations or people that are part of a data-set in terms of their age and salary as you can see we have 30 people here on this chart.
* The ML problem is to classify a new data-point. i.e. what happens if we add a *new observation* a *new data poin*t into the set.
* So, this is a supervised machine learning algorithm because we're classifying something based on previously known Classes.
* Here Naïve Bayes Algorithm is going to help us solve this challenge.

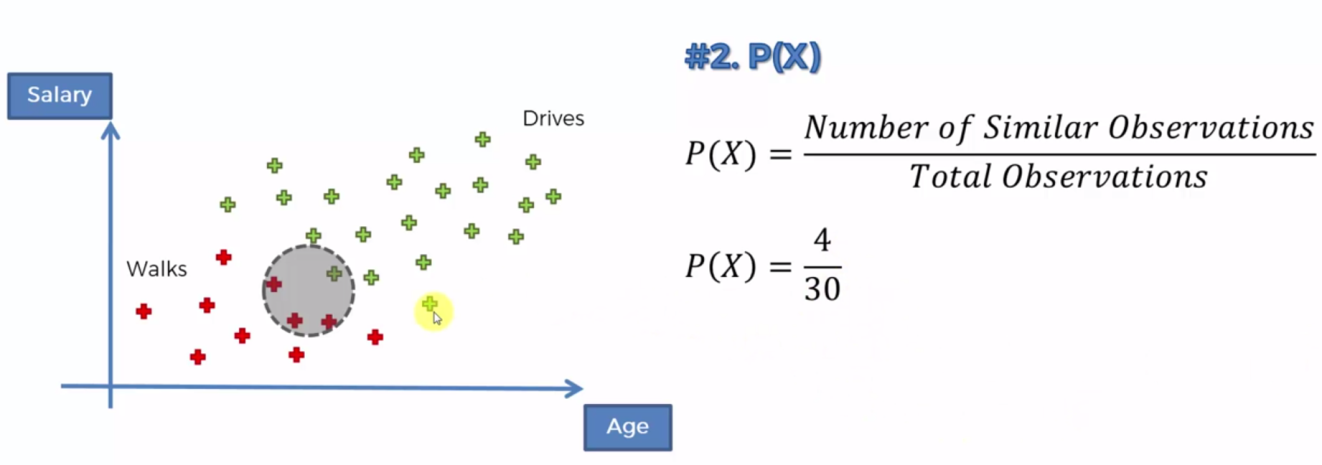
|  |  |
| --- | --- |
| * We're going to take the Bayes Theorem and we can *apply it twice*.  1. Step 1: First, we're going to apply it to find out what is the probability that a *person(new data-point)* ***walks*** *given that his* ***features*** *X* (i.e. age & salary as features). |  |
| Bayes Theorem Example | |
| 1. Step 2: We're going to calculate the probability that somebody drives given those same features X that we see in our *new data-point*. [Probability that a *person(new data-point)* ***drives*** *given that his* ***features*** *X* (i.e. age & salary as features)] |  |
| 1. Step 3: And finally we're going to compare the probability that somebody WALKS given features X versus the probability that somebody DRIVES given features X. Then from that comparison we'll decide which Class to put that new data point in. | |

* You can see that the Naïve Bayes classifier is a probabilistic type of classifier because we're first calculating the probabilities and then based on probabilities we're assigning it to a Class.
* Step 1: We divide each calculation into 4 steps:

|  |  |
| --- | --- |
| 1. Calculate The Prior Probability: We're going to calculate the probability that somebody walks. i.e. we're going to add a new observation to our data-set but we don't know their age and salary (without knowing the features).  * What is the probability that this person that we're adding to our database walks to work. i.e. |  |

* We calculate the *number* of *read* *observations* (which is 10) number of people that actually walk and we divide it by the *number of total observations* (which is 30).
* Hence Prior Probability is:

1. Calculate the Marginal Likelihood: Select a *radius around the data point*. is the *probability* of any *given* *point* to fall within that selected radius.



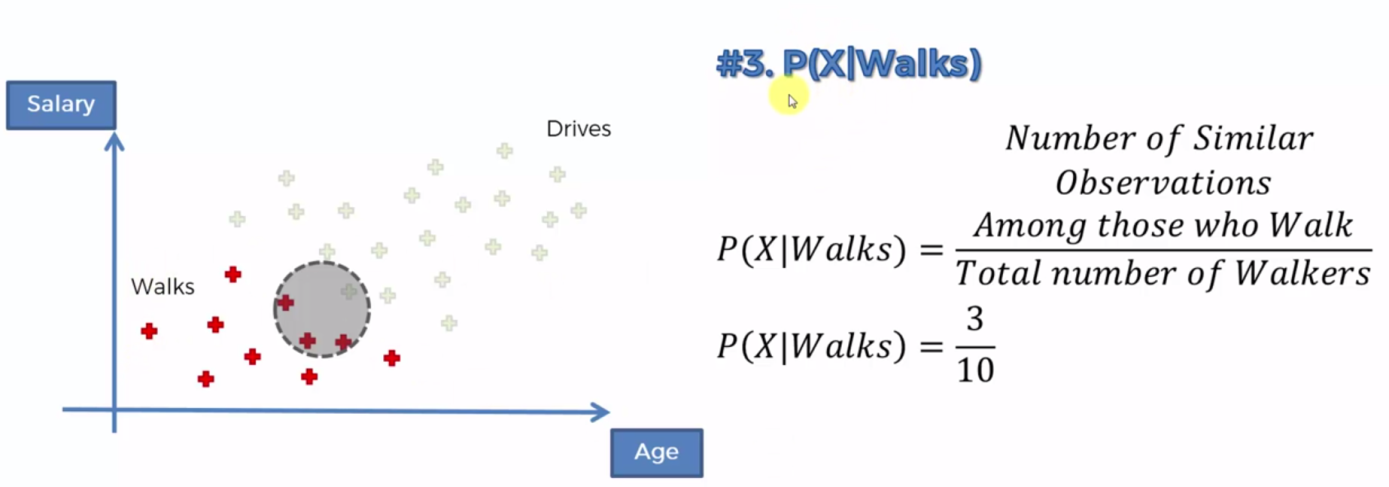
* We're going to select a radius and we're going to draw a circle around our observation. Now this radius you need to select on your own and you need to decide for you.
* We're going to remove our new data-point DOT for now just so that it's not confusing us.
* And then we're going to look at all the points that are *inside* this *circle* and we're going to consider them to be *similar in terms of features* to the new data-point that we had (age 25 and salary of $30000 per year).
* Let's say anybody between the ***age*** of ***20 to 30*** and in the ***salaries*** of ***$25000 to $35000*** are the ***features*** for the ***circled*** ***area***. Anybody that falls in that circle is going to be deemed similar to the new data point that we're adding to our data set.
* Marginal Likelihood: The probability of X, is the *probability of a new point that we add to our data set being* similar *in* features *to the point that we actually are adding to it*. So basically it's a *probability* of that *new* *point* that we're adding (or like any random point) that *fall into this circle*.

it tells us what is the likelihood(possibility) of any new random variable that we add to this data set Falling inside the CIRCLE. And is calculated as:

* We have 3 red dots 1 green dot in the circle. i.e.

1. Likelihood: Same radius. : Probability of the data point would be in this circle given that that data-point Walks.

* What is the likelihood that somebody who walks exhibits features **X**?
* We're going to draw the *same* *circle* again and once again anything that *falls* *inside* the *circle* is deemed to be similar to the point that we're adding.
* So, what is the *likelihood* that a *randomly* *selected* *data* *point* will be fall in this circle given that a person Walks.
* Other way to think about this is we're only working with people who walk to work. So we're only working with the red dots which represent people who walk to work. So let's forget about the green dots.
* So the question becomes: ***"Given that we're only working with the red dots, then what is the likelihood that a randomly selected data-point from the red dots is (exhibits features similar to the point that we are adding to our Dataset) falling into this circle"***.



1. Calculate Posterior Probability:

***As summery:***

1. Prior Probability:
2. Calculate the Marginal Likelihood: Select a radius around the data point. P(X) is the probabilty of any given point to fall within that selected radius.
3. Likelihood: Same radius. : Probability of the data point would be in this circle given that that datapoint walks.
4. Calculate Posterior Probability:

* Step 2:

We consider the new data-point and same circle. Then:

|  |  |
| --- | --- |
| 1. Prior Probability: 2. Calculate the Marginal Likelihood: 3. Likelihood: : Considering only the green points |  |

1. Calculate Posterior Probability:

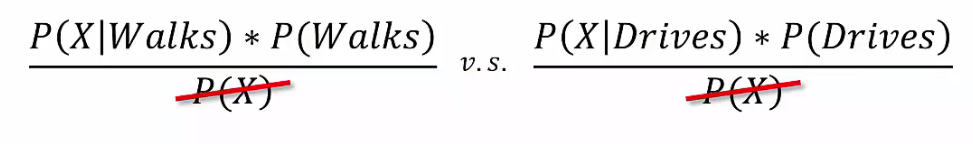
* Step 3: We're going to compare the probability that somebody WALKS given features X versus the probability that somebody DRIVES given features X.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |

* Since in this case the *probability* of the data-point *walking* is greater than *driving* we can say that the data point is assigned to walking.
* Therefore it is more likely that that person with given features X is going to be a person who *walks to work* than the person who drives to work.

That is how the Naïve Bayes Algorithm in ML works.

* Why Naïve?: Because we assume that the features (age and Salary are independent).
* We can actually drop P(X). But it is good to follow full calculation. Some times the shortcut is used. It is ok when we are comparing but when we calculate the Posterior Probability values, we need to follow the full calculations.



* More than 2 classes: We need to compare all possible pair of cases.

**3.5.3 Python Implementation of the Naïve Bayes algorithm:**

|  |  |
| --- | --- |
| * The template is same as before, We only change the algorithm name, and Implement the classifier: * We don’t need any parameters for our classifier. * Confusion matrix and prediction: |  |

Practiced version

#*----------- Naive Bayes model ----------*

#*Library*

**import** pandas **as** pd

**import** matplotlib**.**pyplot **as** pLt

**import** numpy **as** np

#*Data Extract*

dataSet = pd**.read\_csv**("Social\_Network\_Ads.csv")

X = dataSet**.**iloc[:, [2,3]]**.**values

y = dataSet**.**iloc[:, 4]**.**values

        #*Feature-Scaling after Data Split*

#*Data Split*

**from** sklearn**.**model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test = **train\_test\_split**(X, y, test\_size= 0.25, random\_state = 0)

#*Feature-Scaling*

**from** sklearn**.**preprocessing **import** StandardScaler

st\_x= **StandardScaler**()

X\_train= st\_x**.fit\_transform**(X\_train)

X\_test= st\_x**.transform**(X\_test)

#*Fit train set to Naïve Bayes classifier: No parmeter is needed*

**from** sklearn**.**naive\_bayes **import** GaussianNB

clsFier = **GaussianNB**()

clsFier**.fit**(X\_train, y\_train) #*fit the dataset*

#*Predict*

y\_prd = clsFier**.predict**(X\_test)

#*Making the confusion matrix use the function "confusion\_matrix"*

**from** sklearn**.**metrics **import** confusion\_matrix

cm = **confusion\_matrix**(y\_true= y\_test, y\_pred= y\_prd)

#*parameters of cm: y\_true: Real values, y\_pred: Predicted value*

#*Visualising the Training set results*

**from** matplotlib**.**colors **import** ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np**.meshgrid**(np**.arange**(start = X\_set[:, 0]**.min**() - 1, stop = X\_set[:, 0]**.max**() + 1, step = 0.01),

                     np**.arange**(start = X\_set[:, 1]**.min**() - 1, stop = X\_set[:, 1]**.max**() + 1, step = 0.01))

pLt**.contourf**(X1, X2, clsFier**.predict**(np**.array**([X1**.ravel**(), X2**.ravel**()])**.**T)**.reshape**(X1**.**shape),

             alpha = 0.5, cmap = **ListedColormap**(('red', 'green')))

pLt**.xlim**(X1**.min**(), X1**.max**())

pLt**.ylim**(X2**.min**(), X2**.max**())

**for** i, j **in** **enumerate**(np**.unique**(y\_set)):

    pLt**.scatter**(X\_set[y\_set **==** j, 0], X\_set[y\_set **==** j, 1],

                c = **ListedColormap**(('red', 'green'))(i), label = j)

pLt**.title**('Naïve Bayes (Training set)')

pLt**.xlabel**('Age')

pLt**.ylabel**('Estimated Salary')

pLt**.legend**()

pLt**.show**()

#*Visualising the Test set results*

**from** matplotlib**.**colors **import** ListedColormap

X\_set, y\_set = X\_test, y\_test

X1, X2 = np**.meshgrid**(np**.arange**(start = X\_set[:, 0]**.min**() - 1, stop = X\_set[:, 0]**.max**() + 1, step = 0.01),

                     np**.arange**(start = X\_set[:, 1]**.min**() - 1, stop = X\_set[:, 1]**.max**() + 1, step = 0.01))

pLt**.contourf**(X1, X2, clsFier**.predict**(np**.array**([X1**.ravel**(), X2**.ravel**()])**.**T)**.reshape**(X1**.**shape),

             alpha = 0.5, cmap = **ListedColormap**(('red', 'green')))

pLt**.xlim**(X1**.min**(), X1**.max**())

pLt**.ylim**(X2**.min**(), X2**.max**())

**for** i, j **in** **enumerate**(np**.unique**(y\_set)):

    pLt**.scatter**(X\_set[y\_set **==** j, 0], X\_set[y\_set **==** j, 1],

                c = **ListedColormap**(('red', 'green'))(i), label = j)

pLt**.title**('Naïve Bayes (Test set)')

pLt**.xlabel**('Age')

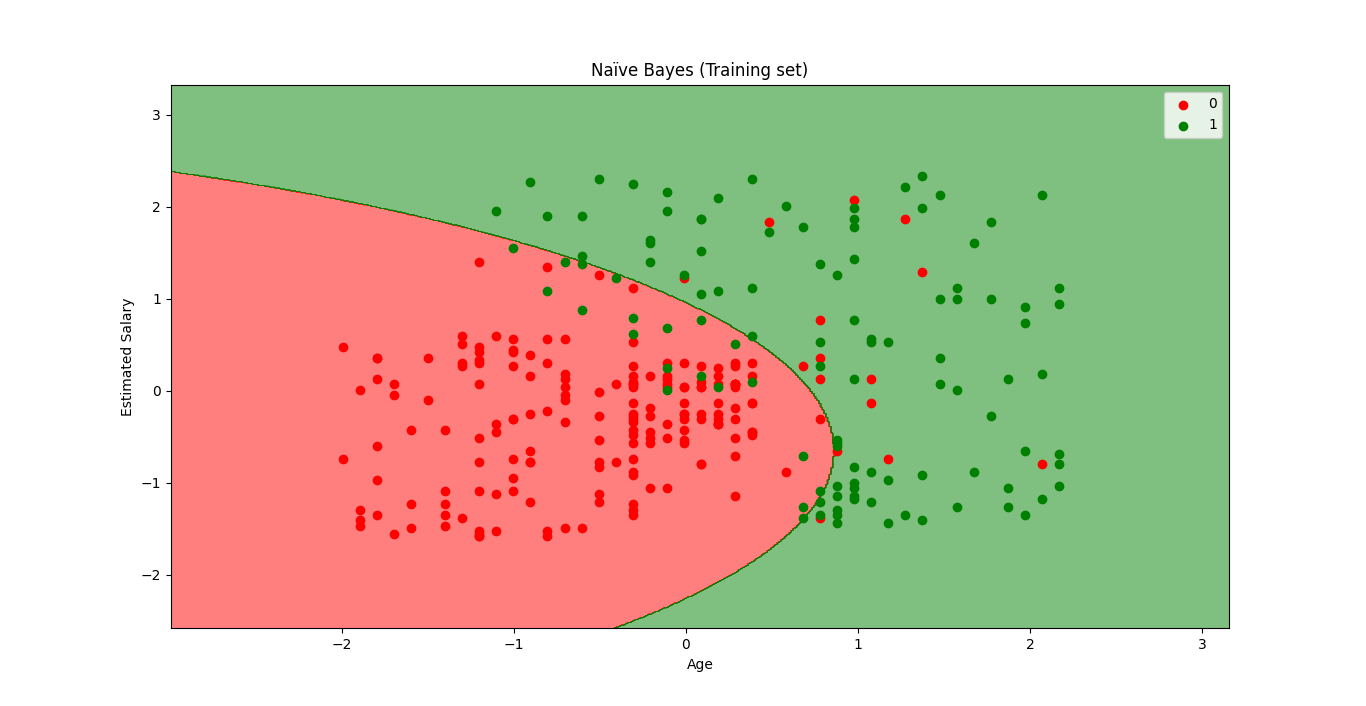
pLt**.ylabel**('Estimated Salary')

pLt**.legend**()

pLt**.show**()

#*python prctc\_niv\_bys.py*

Training Data-set plot



Test Data-set plot

