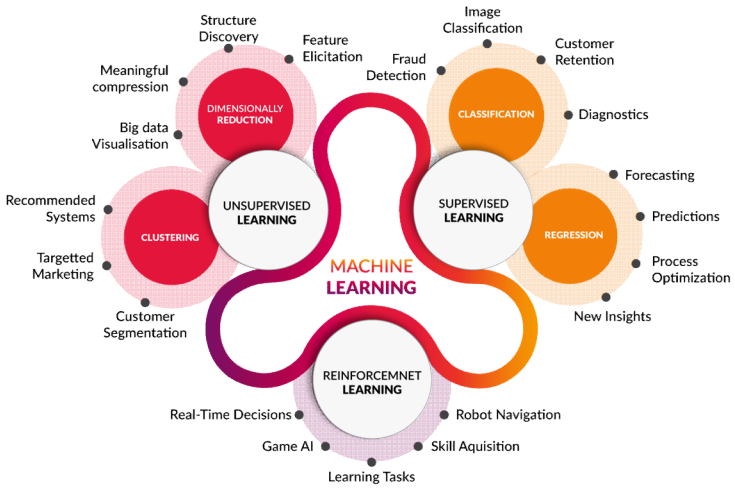
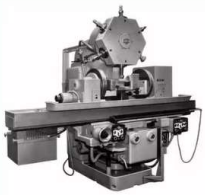
Naïve Bayes Classifier

Machine Learning

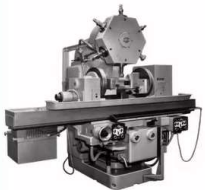
Dr. Adnan Abid



Bayes Theorem



m1 m1 m1 m1 m1 m1 m1 m1 m1 m1 m1 m1 m1



m2 m2 m2 m2 m2 m2 m2 m2 m2 m2

Bayes Theorem

What's the probability?

m2

Find the probability that a wrench has been produced by machine 2 and is defective.

Bayes Theorem

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr

Out of all produced parts:
We can SEE that 1% are defective

Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2

Question:
What is the probability that a part produced by mach2 is defective = ?

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr

-> P(Mach1) = 30/50 = 0.6
-> P(Mach2) = 20/50 = 0.4

Out of all produced parts:
We can SEE that 1% are defective

-> P(Defect) = 1%

Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2

-> P(Mach1 | Defect) = 50%
-> P(Mach2 | Defect) = 50%

Question:
What is the probability that a part produced by mach2 is defective = ?

-> P(Defect | Mach2) = ?

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr

~~-> P(Mach1) = 30/50 = 0.6~~
-> P(Mach2) = 20/50 = 0.4

Out of all produced parts:
We can SEE that 1% are defective

-> P(Defect) = 1%

Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2

~~-> P(Mach1 | Defect) = 50%~~
-> P(Mach2 | Defect) = 50%

We do not need any details related to machine 1. So we can eliminate those details.

Question:
What is the probability that a part produced by mach2 is defective = ?

-> P(Defect | Mach2) = ?

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr
Out of all produced parts:
We can SEE that 1% are defective
Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2
Question:
What is the probability that a part produced by mach2 is defective = ?

-> P(Mach2) = 20/50 = 0.4
-> P(Defect) = 1%
-> P(Mach2 | Defect) = 50%
-> P(Defect | Mach2) = ?

$$P(Defect \mid Mach2) = \frac{P(Mach2 \mid Defect) * P(Defect)}{P(Mach2)}$$

Bayes Theorem

Mach1: 30 wrenches / hr
Mach2: 20 wrenches / hr
Out of all produced parts:
We can SEE that 1% are defective
Out of all defective parts:
We can SEE that 50% came from mach1
And 50% came from mach2
Question:
What is the probability that a part produced by mach2 is defective = ?

-> P(Mach2) = 20/50 = 0.4
-> P(Defect) = 1%
-> P(Mach2 | Defect) = 50%
-> P(Defect | Mach2) = ?

$$P(Defect \mid Mach2) = \frac{0.5 * 0.01}{0.4} = 0.0125$$

It's intuitive!

$$P(\text{Defect} \mid \text{Mach2}) = \frac{P(\text{Mach2} \mid \text{Defect}) * P(\text{Defect})}{P(\text{Mach2})} = 1.25\%$$

Let's look at an example:

- 1000 wrenches
- 400 came from Mach2
- 1% have a defect = 10
- of them 50% came from Mach2 = 5
- % defective parts from Mach2 = 5/400 = 1.25%

It's intuitive!

Obvious question:
If the items are labeled, why couldn't we just count the number of defective wrenches that came from Mach2 and divide by the total number that came from Mach2?

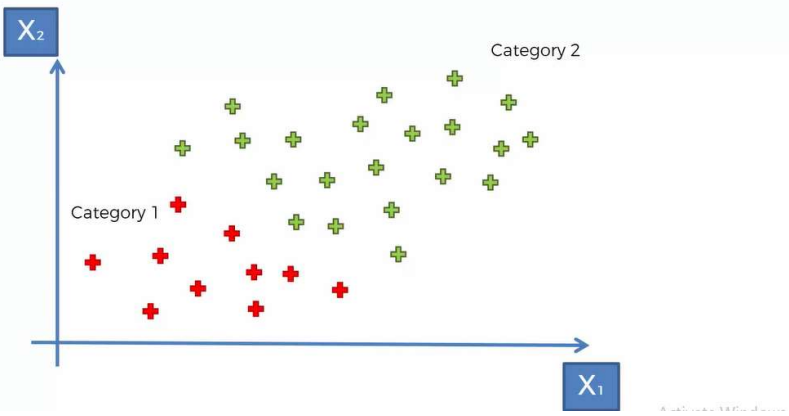
It is time taking and difficult to do this exercise manually

Bayes Theorem

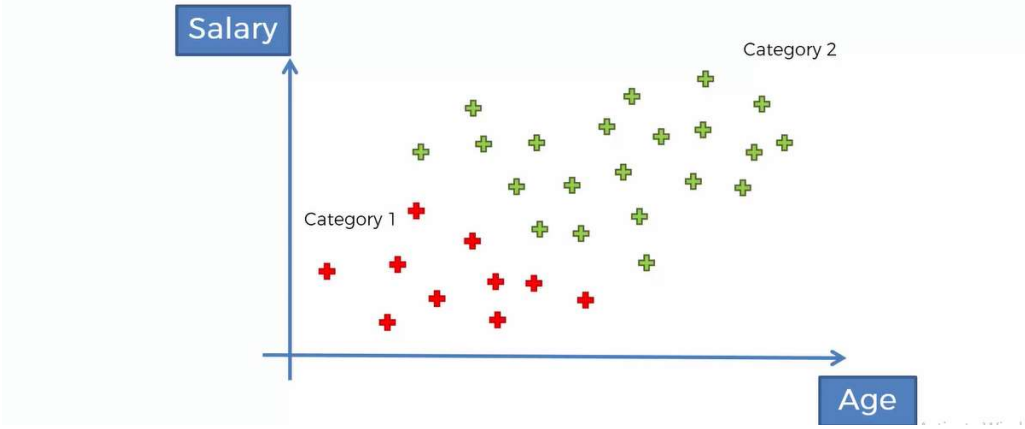
Quick exercise:

P(Defect | Mach1) = ?

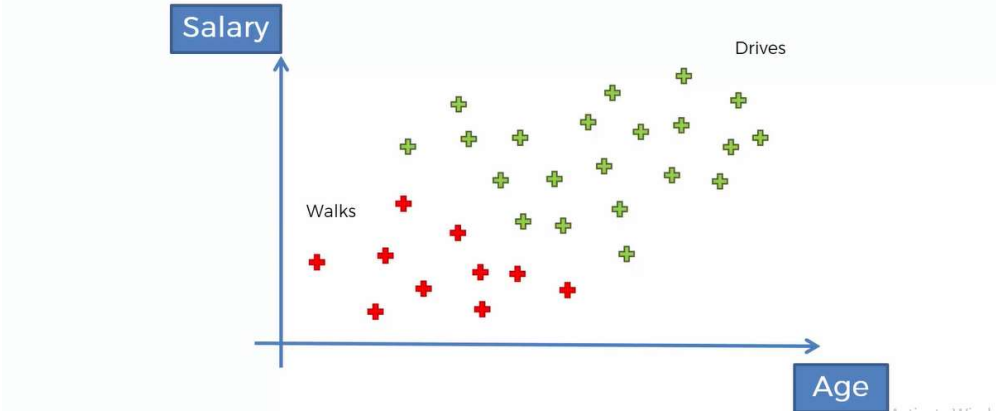
Naïve Bayes



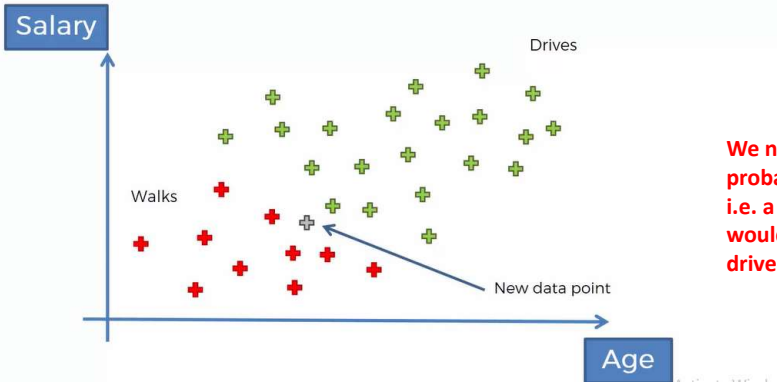
Naïve Bayes



Naïve Bayes



Naïve Bayes



We need to calculate the probability of this new data point i.e. a person with features X, if he would walk to his job or would drive for his job.

Step 1

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

Step 1

#4 Posterior Probability

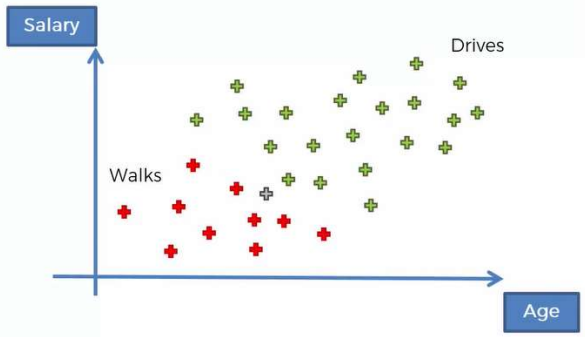
#3 Likelihood

#1 Prior Probability

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

#2 Marginal Likelihood

Naïve Bayes: Step 1

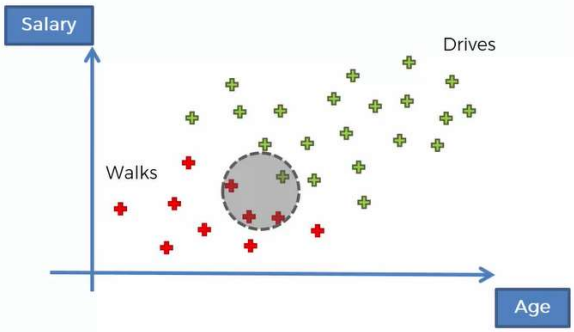


#1. P(Walks)

$$P(Walks) = \frac{Number\ of\ Walkers}{Total\ Observations}$$

$$P(Walks) = \frac{10}{30}$$

Naïve Bayes: Step 1



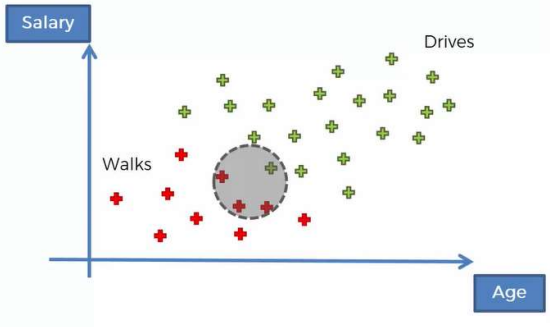
#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

- We draw a circle around the new data point and see how many points are similar to this new point.
- We divide this number by total observations.
- Here the size of the circle may vary

Naïve Bayes: Step 1



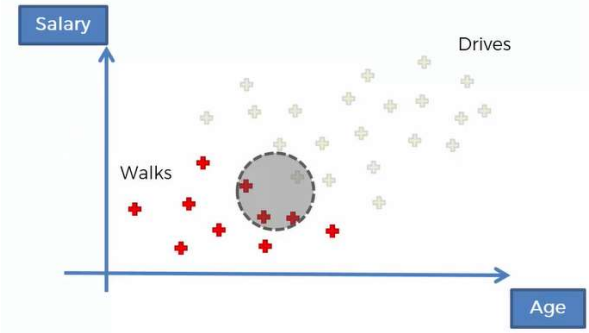
#3. P(X|Walks)

Somebody who WALKs exhibits features X

So we eliminate green dots.

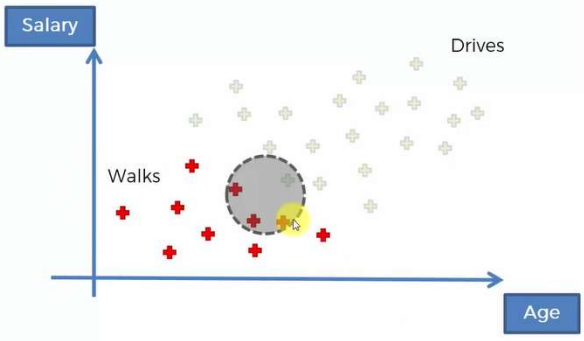
What is the likelihood that a randomly selected data point from the red dots can exhibit properties similar to new data point X.

Naïve Bayes: Step 1



#3. P(X|Walks)

Naïve Bayes: Step 1

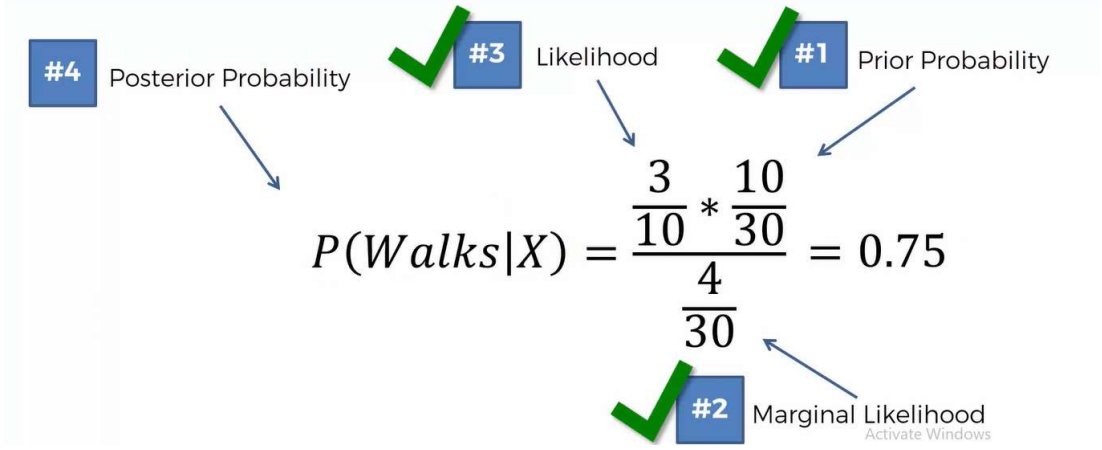


#3. P(X|Walks)

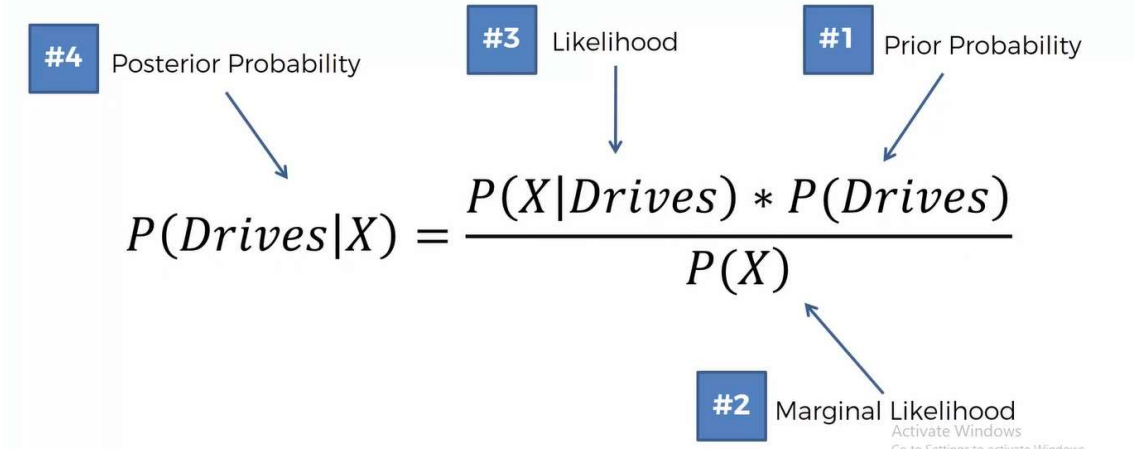
$$P(X|Walks) = \frac{\begin{matrix}Number\ of\ Similar\\ Observations\\ Among\ those\ who\ Walk\end{matrix}}{Total\ number\ of\ Walkers}$$

$$P(X|Walks) = \frac{3}{10}$$

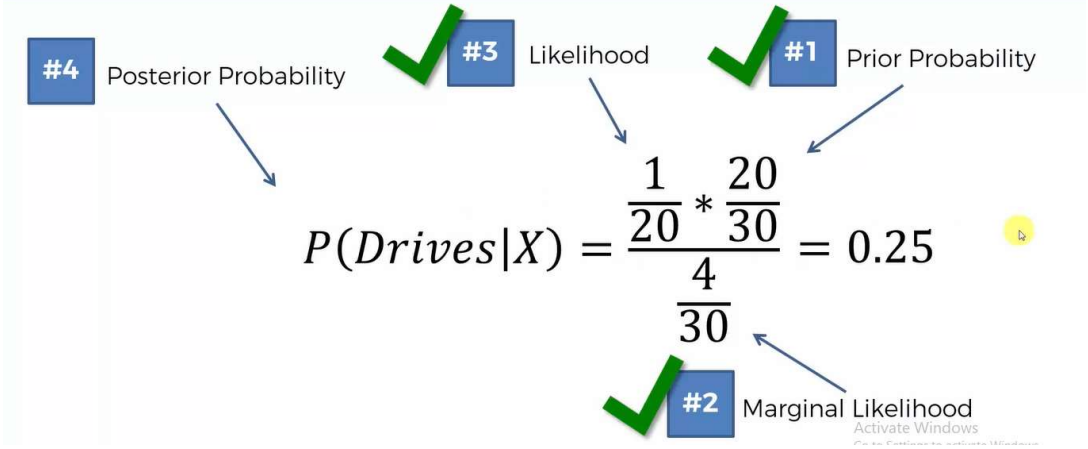
Naïve Bayes: Step 1



Step 2



Naïve Bayes: Step 2



Step 3

$$P(Walks|X) \; v.s. \; P(Drives|X)$$

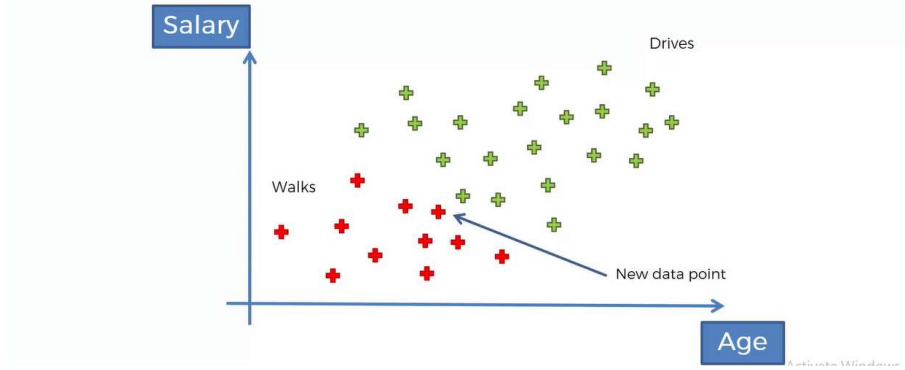
Step 3

$$0.75\ v.s.\ 0.25$$

Step 3

$$P(Walks|X) > P(Drives|X)$$

Naïve Bayes



Step 2

#4 Posterior Probability

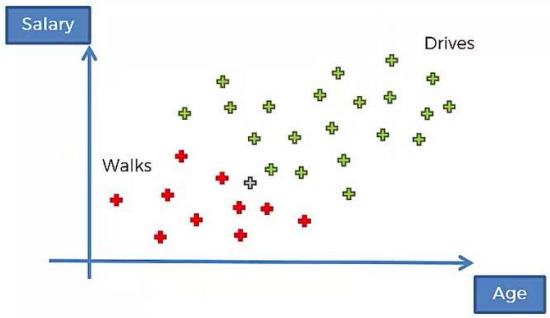
#3 Likelihood

#1 Prior Probability

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

#2 Marginal Likelihood

Naïve Bayes: Step 2

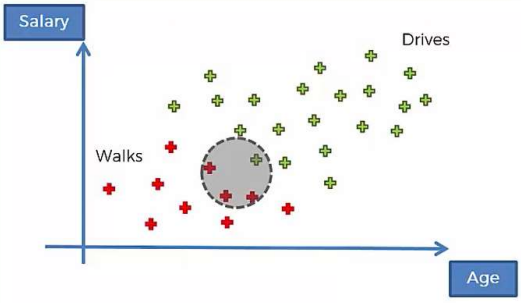


#1. P(Drives)

$$P(Drives) = \frac{Number\ of\ Drivers}{Total\ Observations}$$

$$P(Drives) = \frac{20}{30}$$

Naïve Bayes: Step 2

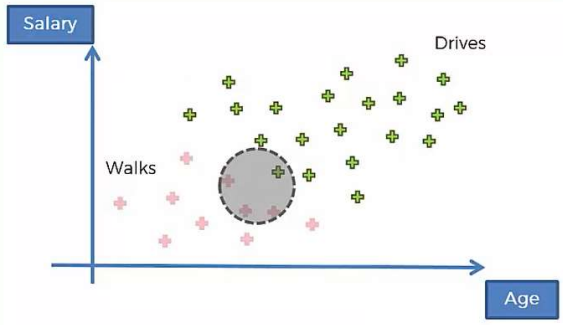


#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

Naïve Bayes: Step 2



#3. P(X|Drives)

$$P(X|Drives) = \frac{\begin{matrix}Number\ of\ Similar \\ Observations \\ Among\ those\ who\ Walk\end{matrix}}{Total\ number\ of\ Walkers}$$

$$P(X|Drives) = \frac{1}{20}$$

Naïve Bayes: Step 2

#4 Posterior Probability

#3 Likelihood

#1 Prior Probability

$$P(Drives|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

#2 Marginal Likelihood

Naïve Bayes

1. Q: Why "Naïve"?
2. P(X)
3. More than 2 features

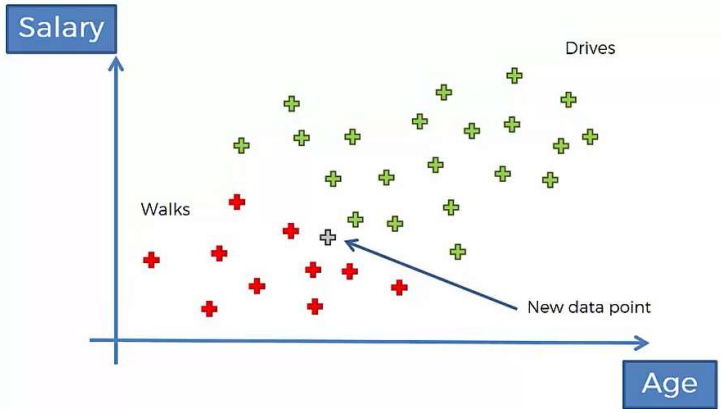
Naïve Bayes

Q: Why "Naïve"?

A: Independence assumption

The independence assumption states that the variab**les/features are independent from one another**

Naïve Bayes



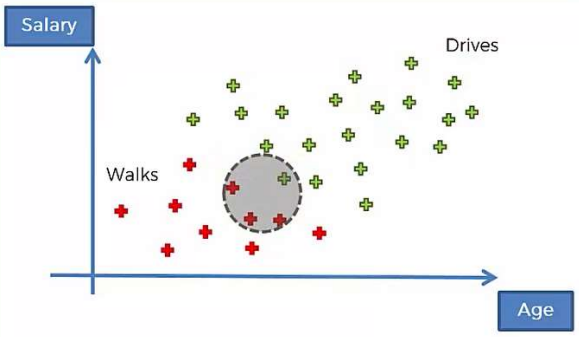
The independence assumption states that the variables/features are independent from one another.

However, in this case we can see that the variables are not independent, as the age grows salary also grows and there is a correlation (may not be super strong correlation) but still it is there.

Naïve Bayes

$$P(X)$$

Naïve Bayes: Step 2



#2. P(X)

$$P(X) = \frac{Number\ of\ Similar\ Observations}{Total\ Observations}$$

$$P(X) = \frac{4}{30}$$

This values does not change for any scenario, i.e. it is same for persons driving or walking to work.

As it is based on the number of observations similar to the new one, divided by total observations.

Step 3

$$P(Walks|X) \quad v.s. \quad P(Drives|X)$$

Step 3

$$\frac{P(X|Walks) * P(Walks)}{P(X)} \quad v.s. \quad \frac{P(X|Drives) * P(Drives)}{P(X)}$$

Step 3

$$\frac{P(X|Walks) * P(Walks)}{\cancel{P(X)}} \quad v.s. \quad \frac{P(X|Drives) * P(Drives)}{\cancel{P(X)}}$$

Naïve Bayes

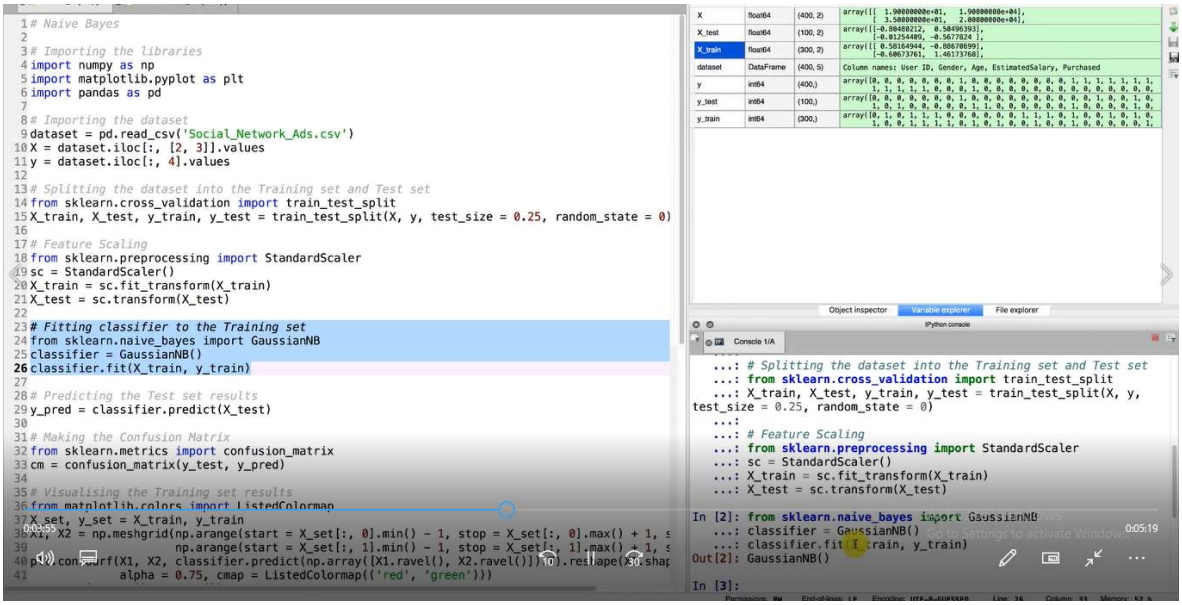
More than 2 classes

Step 3

$$P(Walks|X)\ v.s.\ P(Drives|X)$$

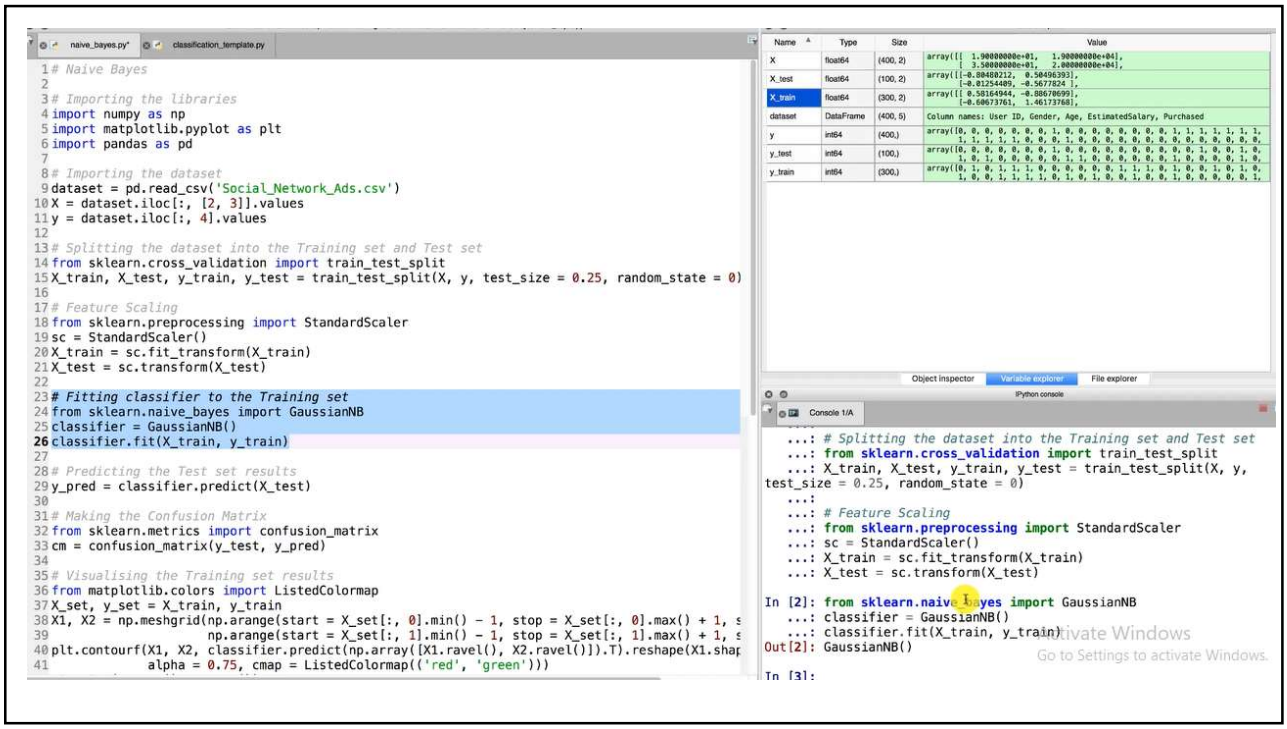
Step 3

$$0.75\ v.s.\ 0.25$$



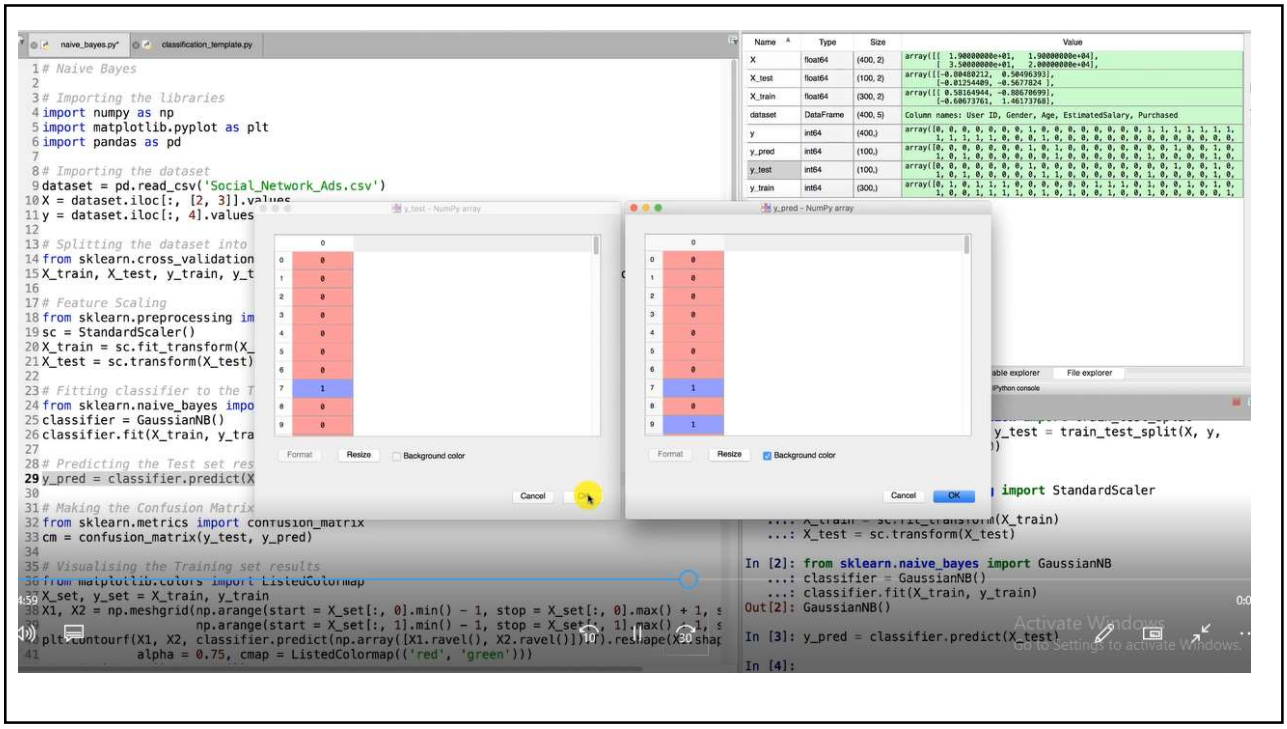
```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation 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
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# 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, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()])).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
```

```
...: # Splitting the dataset into the Training set and Test set
...: from sklearn.cross_validation 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
...: sc = StandardScaler()
...: X_train = sc.fit_transform(X_train)
...: X_test = sc.transform(X_test)

In [2]: from sklearn.naive_bayes import GaussianNB
...: classifier = GaussianNB()
...: classifier.fit(X_train, y_train)
Out[2]: GaussianNB()

In [3]:
```



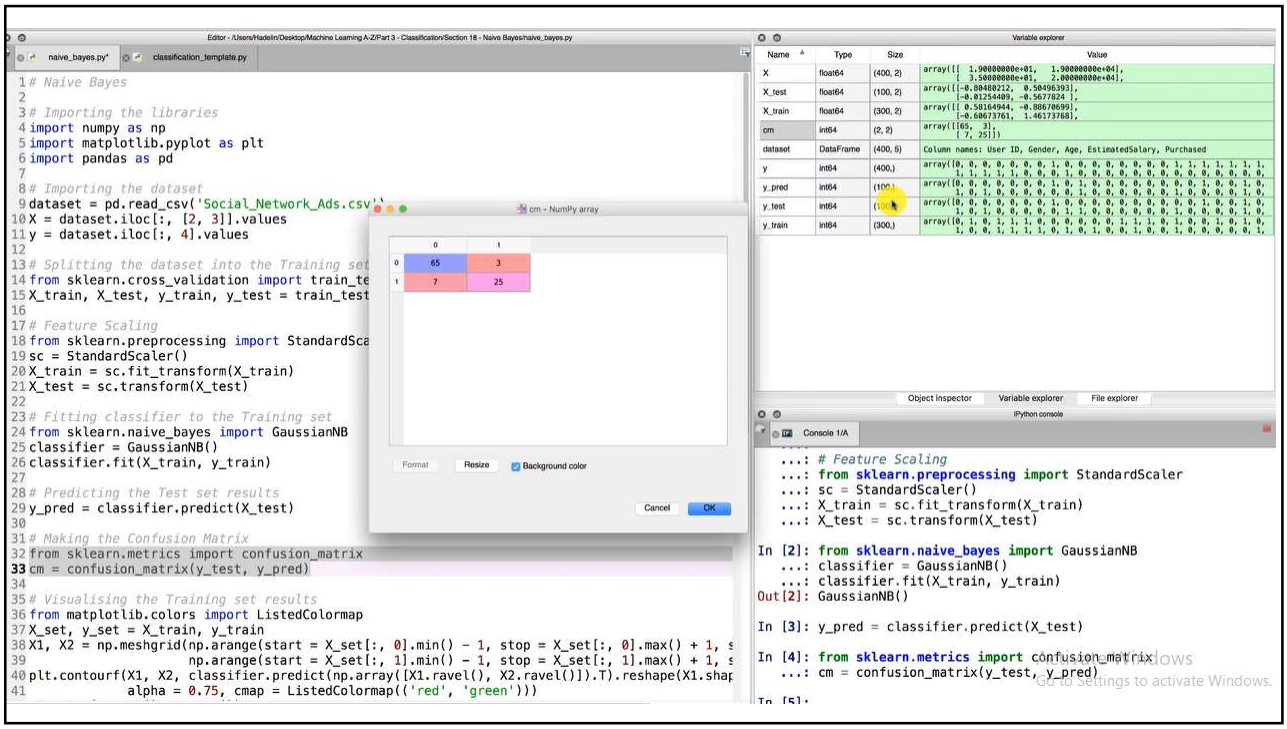
```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.cross_validation 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
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting classifier to the Training set
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# 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, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()])).T).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
```

```
In [2]: from sklearn.naive_bayes import GaussianNB
   ...: classifier = GaussianNB()
   ...: classifier.fit(X_train, y_train)
Out[2]: GaussianNB()

In [3]:
```

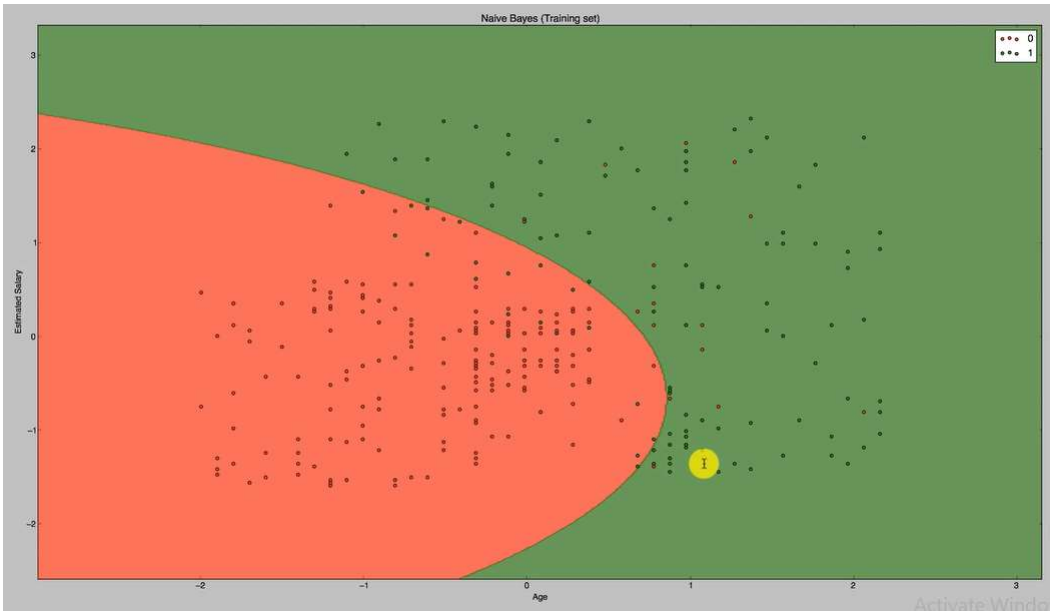


```
# Naive Bayes

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into
from sklearn.cross_validation
X_train, X_test, y_train, y_t

# Feature Scaling
from sklearn.preprocessing im
sc = StandardScaler()
X_train = sc.fit_transform(X_
X_test = sc.transform(X_test)

# Fitting classifier to the T
from sklearn.naive_bayes impo
classifier = GaussianNB()
classifier.fit(X_train, y_tra

# Predicting the Test set res
y_pred = classifier.predict(X

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)

# 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, s
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, s
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()])).T).reshape(X1.shap
             alpha = 0.75, cmap = ListedColormap(('red', 'green')))
```

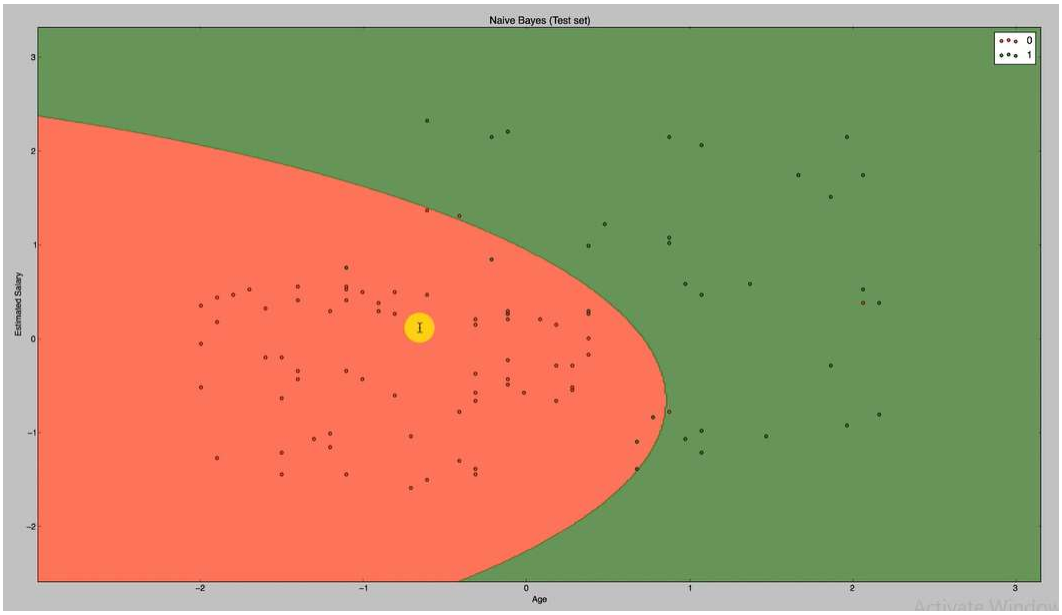
```
y_test = train_test_split(X, y,

import StandardScaler

X_train)
X_test = sc.transform(X_test)

In [2]: from sklearn.naive_bayes import GaussianNB
   ...: classifier = GaussianNB()
   ...: classifier.fit(X_train, y_train)
Out[2]: GaussianNB()

In [3]: y_pred = classifier.predict(X_test)

In [4]:
```



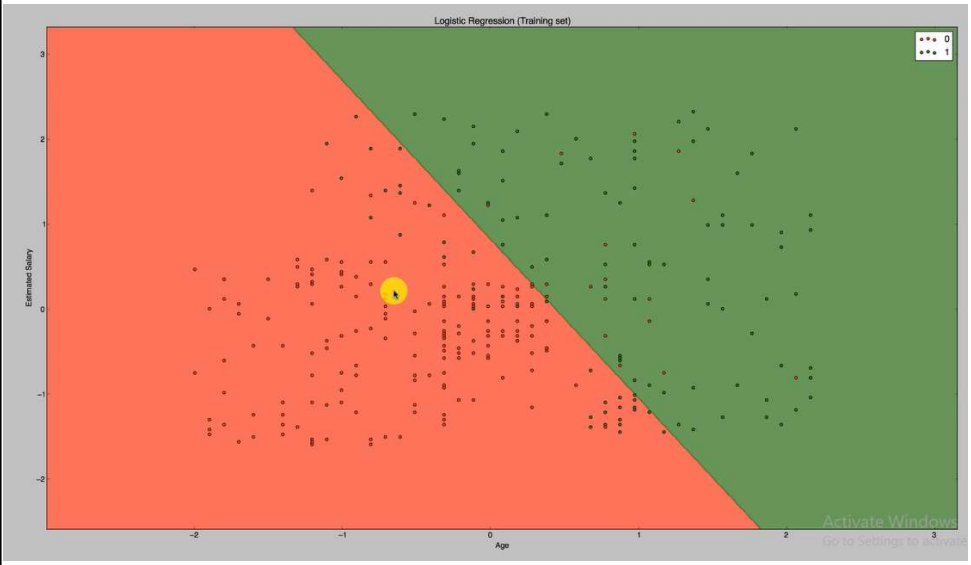
NAIVE BAYES CLASSIFIER (Training Set)



NAIVE BAYES CLASSIFIER (Test Set)



LOGISTIC REGRESSION CLASSIFIER



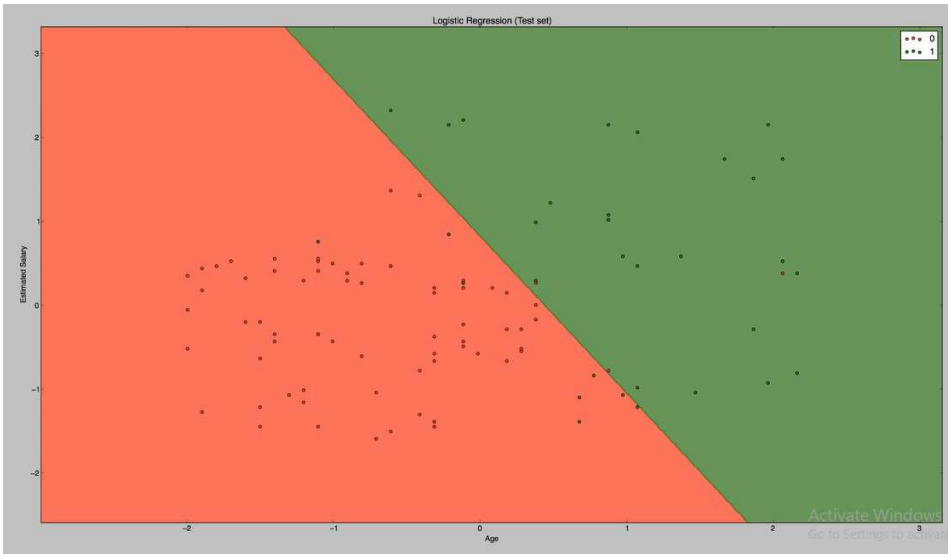
TRAINING SET PLOT

Boundary is a straight line, as logistic regression is a linear classifier. In higher dimensions it will be plane or hyperplane.

The regions are well fitted according to the training data set, though we have some incorrect plots.

Thus, fulfilling the goad of plotting right users in right categories.

LOGISTIC REGRESSION CLASSIFIER



TEST SET PLOT
Test set is reflecting the result of the confusion matrix.