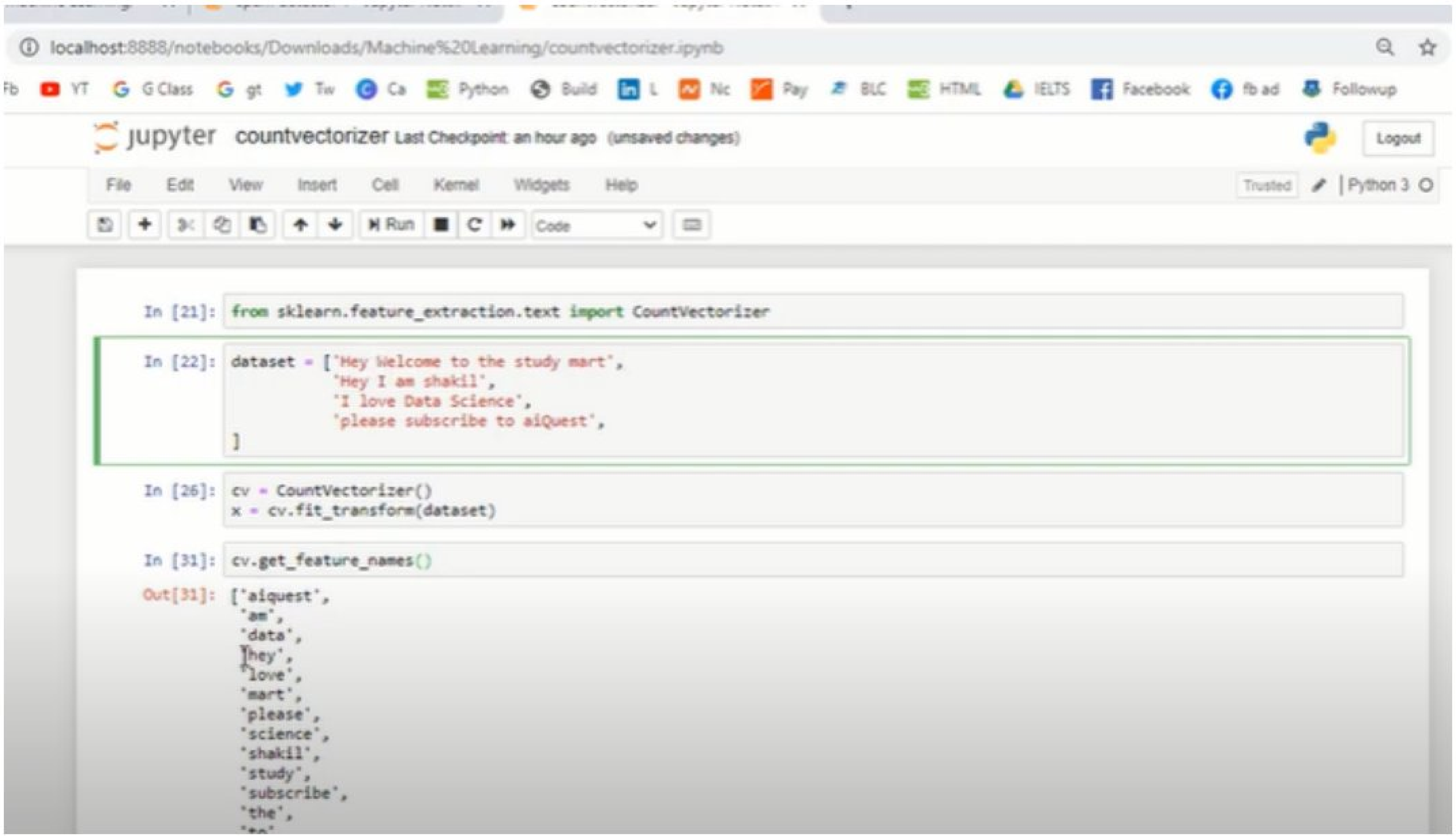
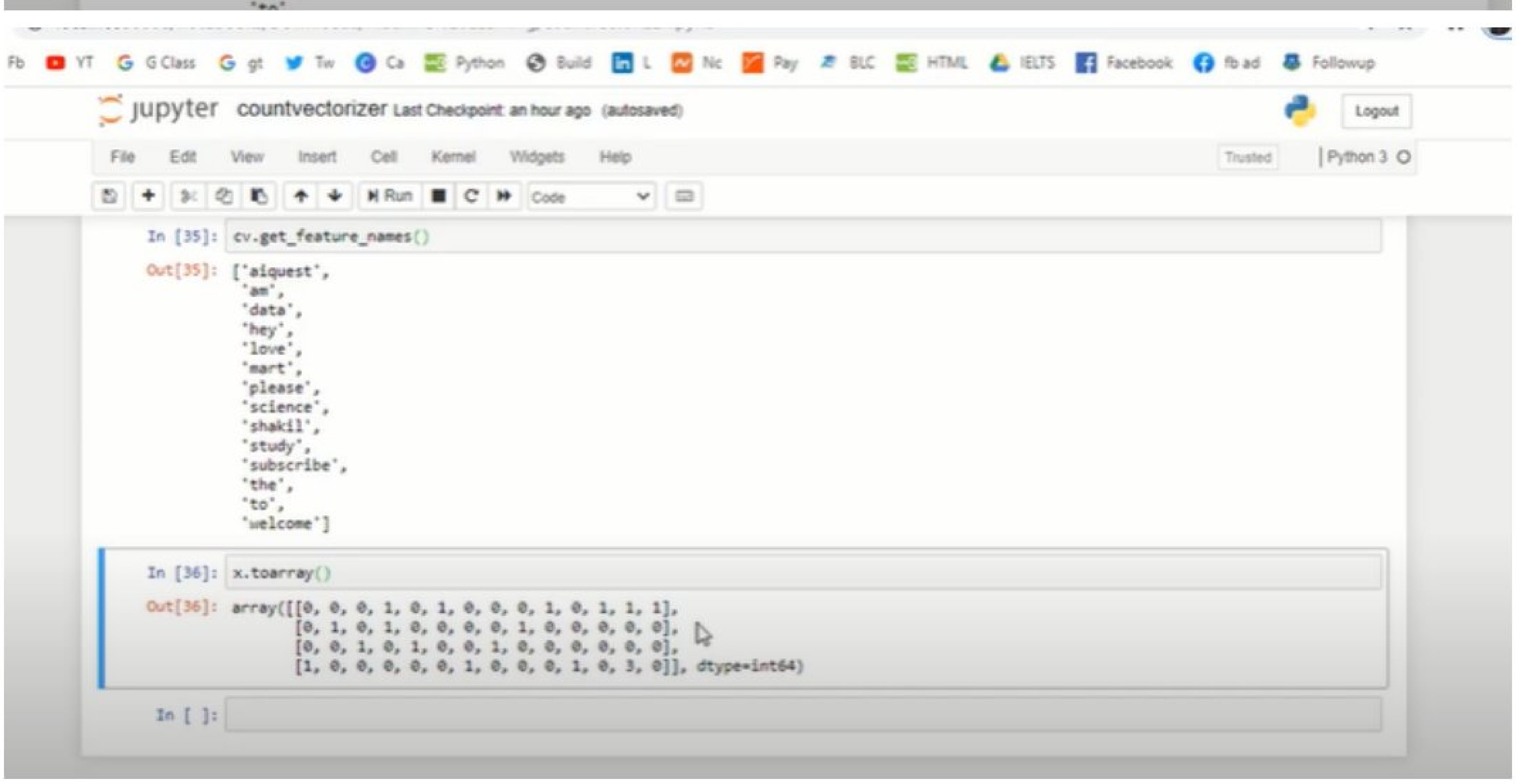

localhost:8888/notebooks/Downloads/Machine%20Learning/countvectorizer.ipynb
jupyter countvectorizer Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
In [21]: from sklearn.feature_extraction.text import CountVectorizer
In [22]: dataset = ['Hey Welcome to the study mart',
'Hey I am shakil',
'I love Data Science',
'please subscribe to aiQuest',
]
In [26]: cv = CountVectorizer()
x = cv.fit_transform(dataset)
In [31]: cv.get_feature_names()
Out[31]: ['aiquest',
'am',
'data',
'hey',
'love',
'mart',
'please',
'science',
'shakil',
'study',
'subscribe',
'the',

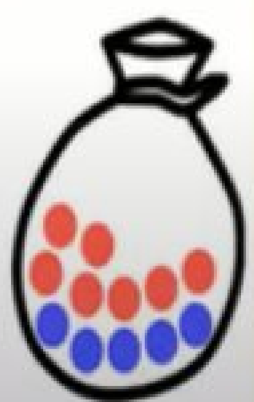

jupyter countvectorizer Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
In [35]: cv.get_feature_names()
Out[35]: ['aiquest',
'am',
'data',
'hey',
'love',
'mart',
'please',
'science',
'shakil',
'study',
'subscribe',
'the',
'to',
'welcome']
In [36]: x.toarray()
Out[36]: array([[0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1],
[0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 3, 0]], dtype=int64)
In [ ]:

Naïve Bayes

Simple Probability

$$\text{Probabilty} = \frac{\text{Favorable outcomes}}{\text{Total outcomes}}$$

Example:



$$P(red) = \frac{7}{12}$$

7 — Number of red marbles

12 — Total number of marbles (sample space)

$$P(blue) = \frac{5}{12}$$

5 — Number of blue marbles

12 — Total number of marbles (sample space)

Experimental Probability

Experimental Probability is found by repeating an experiment and observing the outcomes.

$$P(\text{event}) = \frac{\text{number of times event occurs}}{\text{total number of trials}}$$

Example:
A coin is tossed 10 times.
A head is recorded 7 times
and a tail 3 times.

$$P(\text{head}) = \frac{7}{10} \qquad P(\text{tail}) = \frac{3}{10}$$

Naïve Bayes

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

Likelihood — Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Posterior Probability — Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Above,

- *P(c|x)* is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- *P(c)* is the prior probability of *class*.
- *P(x|c)* is the likelihood which is the probability of *predictor* given *class*.
- *P(x)* is the prior probability of *predictor*.

Naïve Bayes

Step 1: Convert the data set into a frequency table

Step 2: Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

How Naive Bayes algorithm works?

Naïve Bayes

Step 3: Part 1

Problem: Players will play if weather is **Sunny**. Is this statement is **correct**?

We can solve it using above discussed method of posterior probability.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

Here, We have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P(Yes)= 9/14 = 0.64

Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has **Higher** probability.

Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes

Naïve Bayes

How to build a basic model using Naive Bayes in Python ?

Scikit learn (Python Machine Learning library) will help here to build a Naive Bayes model in Python. There are three types of Naive Bayes model under the scikit-learn library:

- **GaussianNB**
- **MultinomialNB**
- **BernoulliNB**

Naïve Bayes

Gaussian: It is used in classification and it assumes that features follow a normal distribution.

Multinomial: It is used for discrete counts. For example, let's say, we have a text classification problem. Here we can consider Bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

Bernoulli: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

Naïve Bayes

Applications of Naive Bayes Algorithms:

- Real time Prediction
- Recommendation System
- Text classification/ Spam Filtering/ Sentiment Analysis:
- Multi class Prediction:

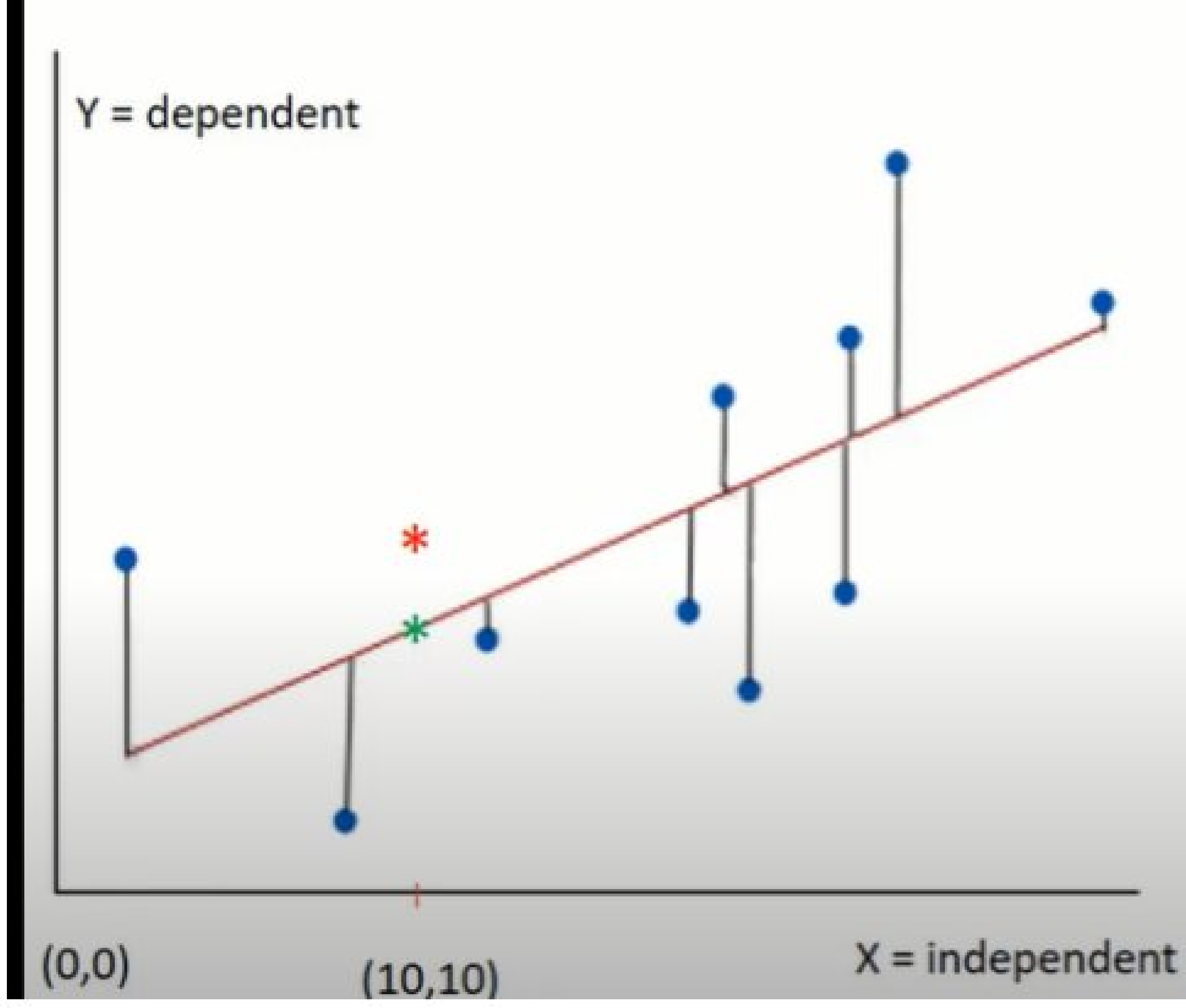
```
xtrain
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AM
26340	26341	150000.0	2	1	2	26.0	0	0	2	0	...	63156.0	57147.0	53383.0	48556.0	485
3877	3878	50000.0	2	2	2	23.0	0	0	0	0	...	42162.0	6765.0	19286.0	9558.0	200
27081	27082	80000.0	1	2	2	31.0	0	0	0	0	...	72368.0	77505.0	78845.0	74182.0	300
5440	5441	120000.0	2	2	2	39.0	0	0	2	0	...	24372.0	24670.0	25617.0	26535.0	240
22164	22165	150000.0	2	1	3	30.0	1	-1	2	-1	...	6527.0	2168.0	-7.0	1373.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
17289	17290	170000.0	2	3	1	40.0	1	2	0	0	...	20619.0	19520.0	5200.0	0.0	
5192	5193	330000.0	2	1	1	41.0	-1	-1	-2	-2	...	0.0	0.0	0.0	0.0	
12172	12173	50000.0	1	2	2	22.0	0	0	0	0	...	35458.0	19778.0	19929.0	19790.0	170

Performance Metrics

- Regression
  - R-Squared Value
- Classification
  - Confusion Matrix
  - Accuracy
  - F1 Measure
  - Recall
  - ROC Curve
- Clustering
  - Elbow Method

R-Squared Value



$$R^2 = \frac{\sum_{i=1}^{n} (y_i - \bar{y})}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Here,

i = 1, to n (integer value)

$\hat{y}_i$ = Predicted Output for i no data

$\bar{y}$ = Mean Value of real outputs

y_i = Real output value for each i data

Confusion Matrix

- Accuracy
- Precision
- ReCall
- F1 Measure

Confusion Matrix

True Positives (TPs)	False Positives (FPs)
False Negatives (FNs)	True Negatives (TNs)

Confusion Matrix

Confusion Matrix and ROC Curve

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

TN	True Negative
FP	False Positive
FN	False Negative
TP	True Positive

Model Performance

Accuracy	= (TN+TP)/(TN+FP+FN+TP)
Precision	= TP/(FP+TP)

Confusion Matrix

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

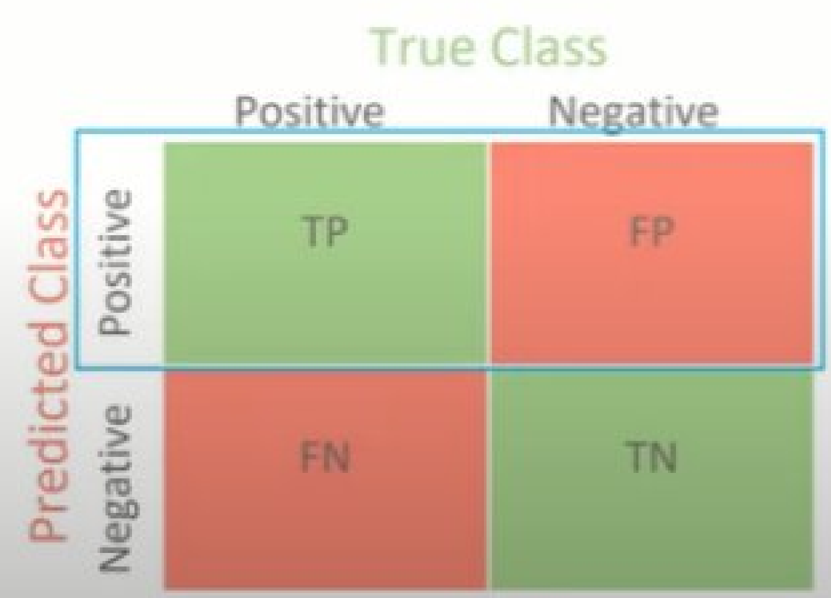
TP = True positive

TN = True negative

FP = False positive

FN = False negative

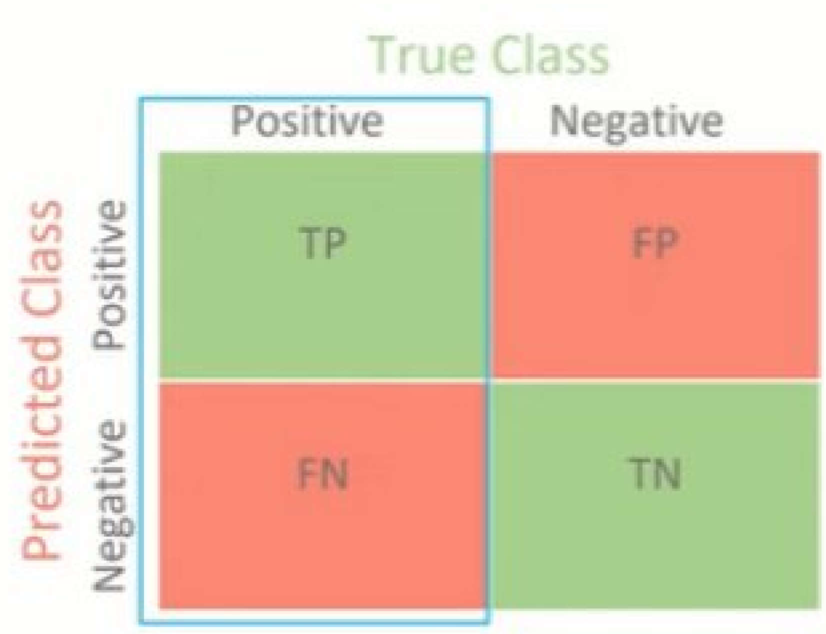
$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Positive}} \Rightarrow \text{Precision} = \frac{\text{True Positive}}{\text{Total Predicted Positive}}$$



Precision is the ratio of number of **True Positive** to the **total number of Predicted Positive**. It measures, out of the total predicted positive, how many are actually positive.

Precision measures the error caused by **False Positives**. Hence it is a good evaluation metric when **False Positive** predictions are critical.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \textbf{False Negative}} \Rightarrow \text{Recall} = \frac{\text{True Positive}}{\text{Total Actual Positive}}$$



Recall is the ratio of number of **True Positive** to the **total number of Actual Positive**. It measures, out of the total actual positive, how many are predicted as True Positive.

Recall measures the error caused by **False Negatives**. Hence it is a good evaluation metric when **False Negative** predictions are critical.