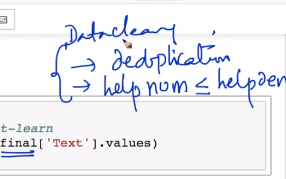
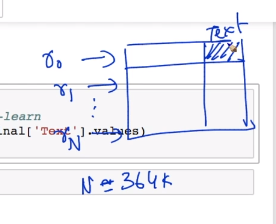
**After Data cleaning step :**

i.e, Removing duplicate entries and cleaning data where helpfulness numerator was > helpfulness denominator which is absolutely wrong.



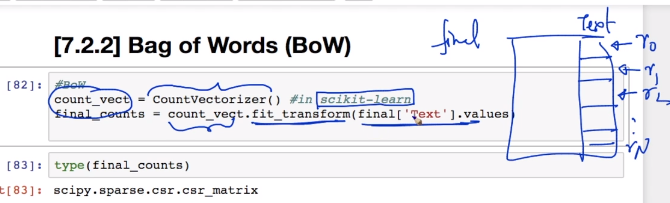
The data we had was **“final”**

And in that **“final” dataframe** we had a column called **“text”** which contains actually text reviews given by customer for any particular product.



Now we need to convert all the **364k rows** or entries to vector.(we can write code for it but we already have function for it)

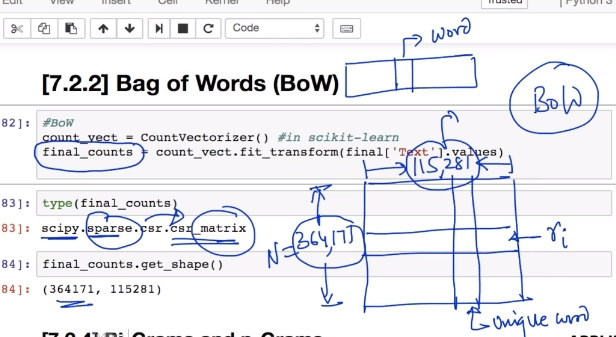
We have a function or class in scikit-learn called ***countVectorizer().***



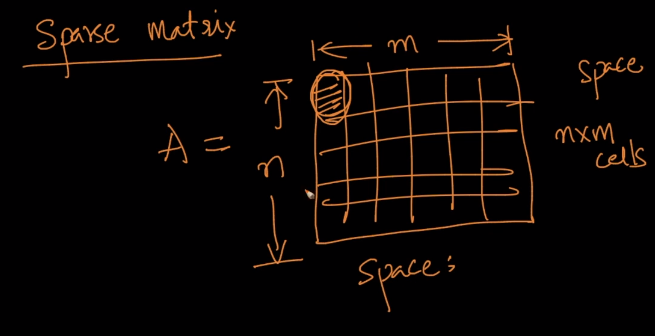
All we have to do is using this class we have to do fit\_transform for values of ‘text’ in final dataframe.

And returns us an **“Scipy\_Sparse\_Matrix”**

This sparse matix has 364171 rows and 115281 columns where each row is a review and each column is a unique word in whole corpus.



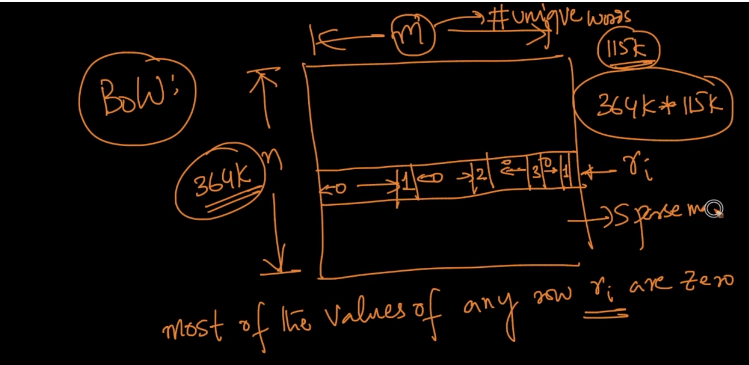
\_\_\_Now to understand what sparse matrix is we need to understand what is normal matrix.



So in above matrix we need to store a value for every index and so the complexity will be **o(n\*m)**

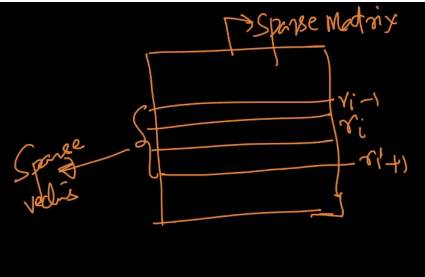
But when we take case of sparse matrix it is not worth to store each and every data because for any review **most of columns are 0** as we can see in below image because every sentence or say every will not contain every unique word.

**And in our case if we multiply 364k \* 115k it is very large.**



And Each row here is called **“Sparse Vector” .**

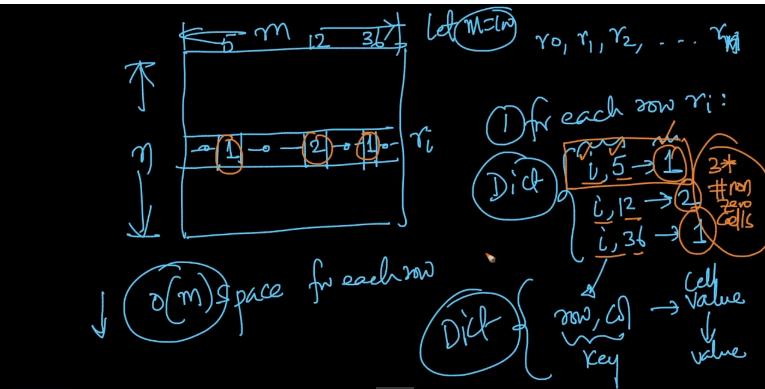
**In a nut shell if we have lots of sparse vector appended upon each other than the matrix formed is called sparse matrix.**



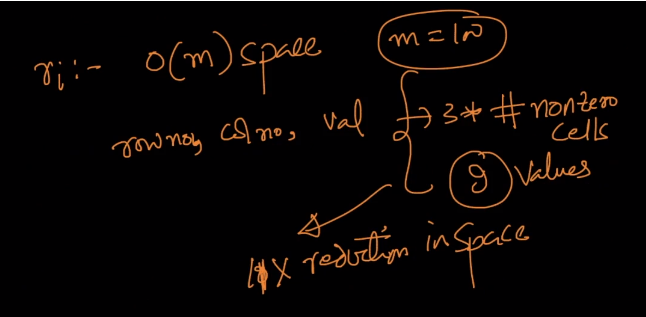
Since I have information that most of the elements in my sparse matrix are zero so we can reduce our space complexity.

So if store every element of a row space complexity will be **o(m)** and since we know that **most of the element in row is 0** so what we can do is for every row we can just store

**(row,column) as key and it value as value in python dictionary and so by doing this we just need to store (3\*# of non zero cells)**



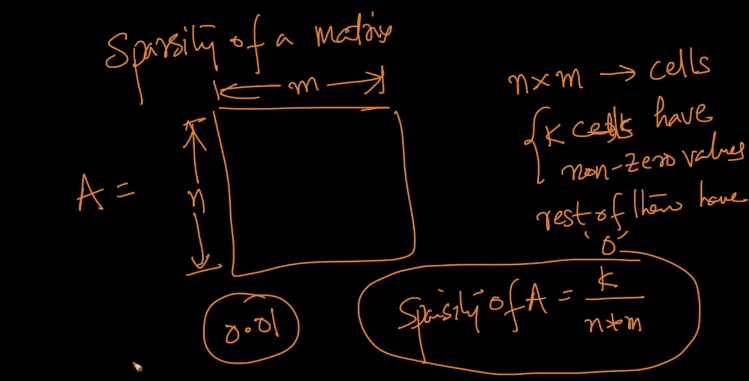
So what we did is instead of storing all 100 values we just stored 9 values which is almost **11 times reduction .**



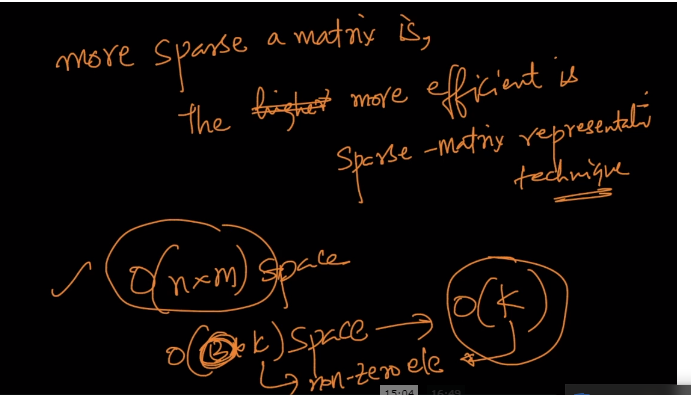
**Sparsity Of a matrix:**

Suppose we have k cells with non zero value in any sparse matrix so sparsity of matrix will be **(k/(n\*m)).**

And suppose if the value of sparsity of matrix is **.01** that means that only **1% of the values are non zero.**



The more sparse a matrix is ,more efficient is sparse matrix representation because we are reducing the space complexity here and less the value of k more the reduction in space complexity.



**NOTE: Please check documentation for countVectorizer() so to understand all the parameters it has.**