Experiment 3:

1. There are 5572 records in the data set. The distribution of the labels is

Ham	4825
Spam	747

Ham is right skewed; Spam is left skewed and the histogram of sms length is right sk ewed.

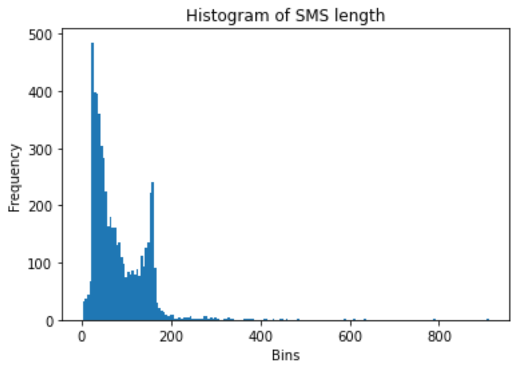
2.

5169 unique sms message. Most frequent sms message is “Sorry, I'll call later” with a frequency of 30.

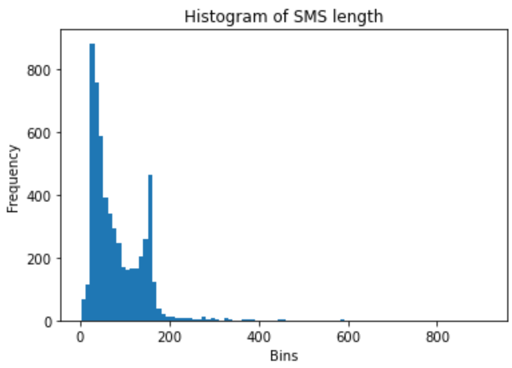
```
In [85]: df_sms.sms.value_counts()

Out[85]: Sorry, I'll call later
         30
         I cant pick the phone right now. Pls send a message
         12
         Ok...
```

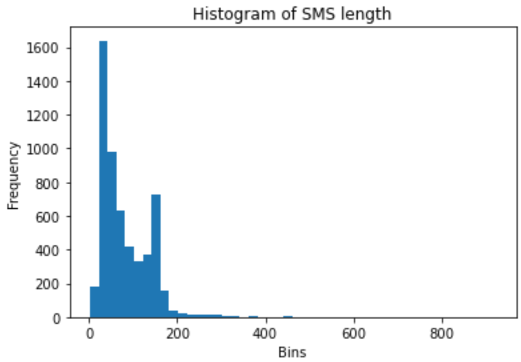
3. Max length of SMS is 910 and Min length of SMS is 2.



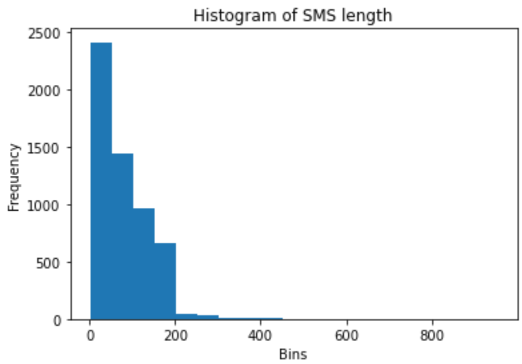
Bin size - 5



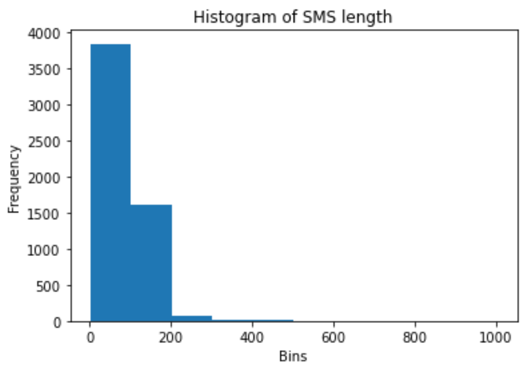
Bin size - 10



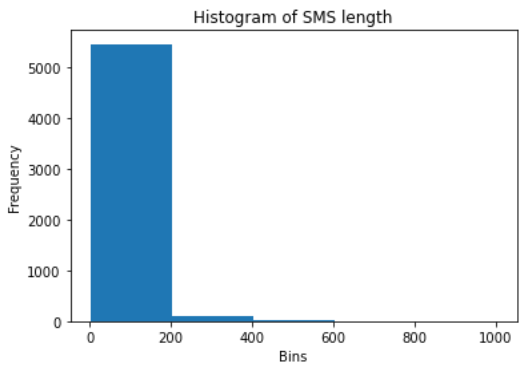
Bin size – 20



Bin size -50



Bin size – 100

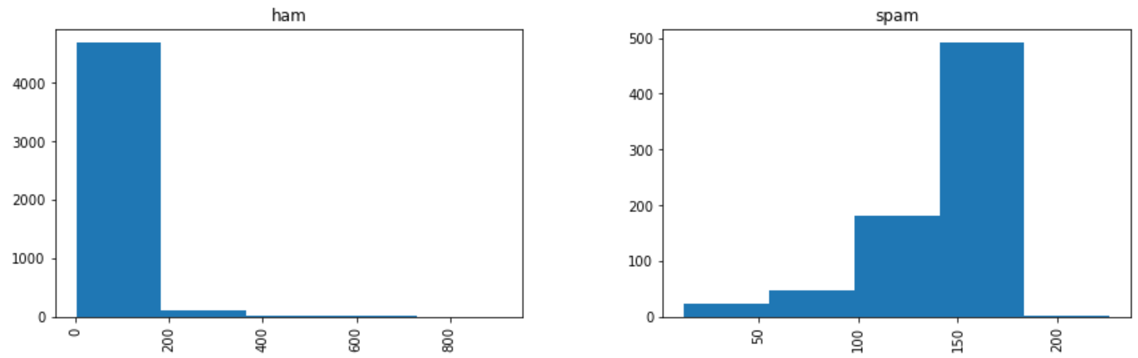


Bin size – 200

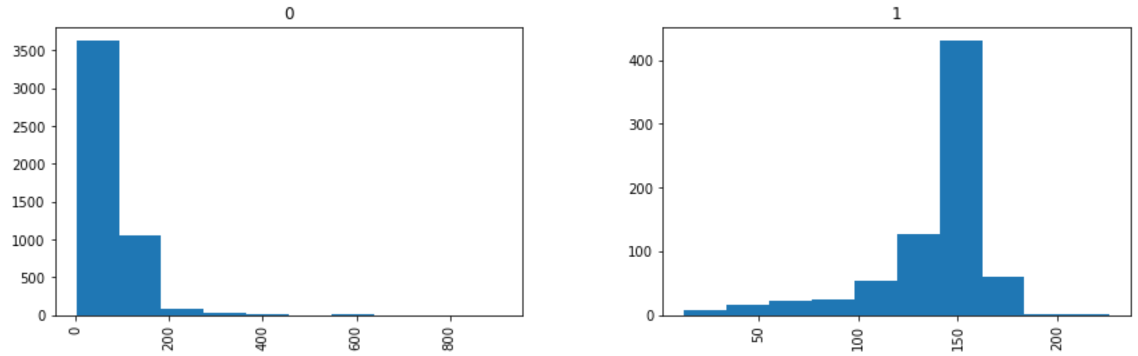
The bin width changes the ability of a histogram to identify local regions of higher incidence and the granularity of the data. So as the bin size increases, the histogram will lose the granularity of data. In particular here, we can observe that as the size increases, we are losing any indication of a second peak.

4.

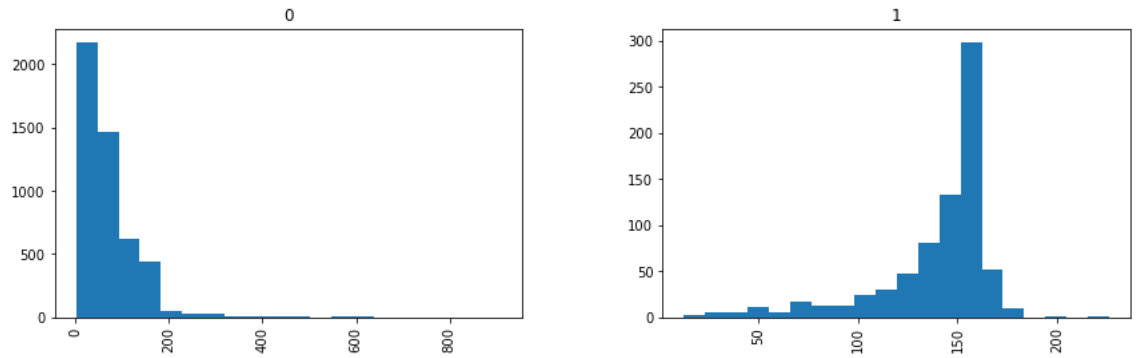
Bins = 5



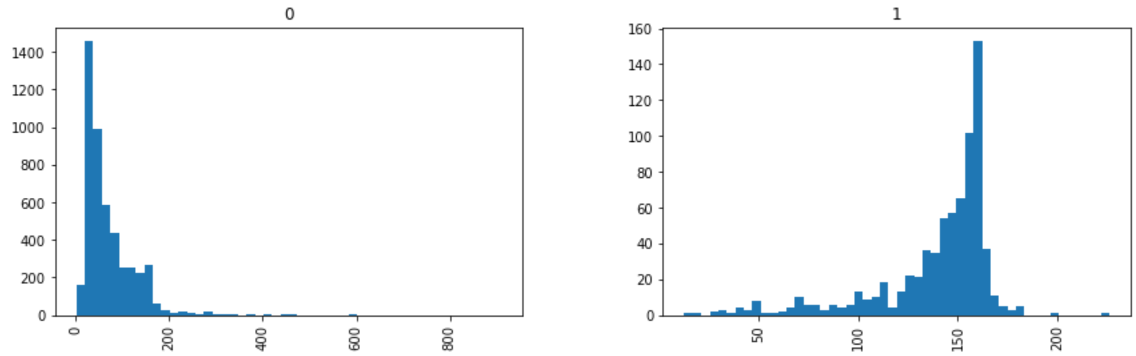
Bins = 10



Bins = 20



Bin s = 50



The number of bins increase with decrease in bin width. As bins increase the granularity of the data improves.

5. We convert the words to lowercase to normalize, to avoid duplicate words due to variation in latter case. Say the words "Is" and "is", due to difference in letter case the two words would be considered as different. Converting all upper-case letters would also work to fulfill the original goal.

6. Count vectorization does the following:
   i. It tokenizes the string (separates the string into individual words) and gives an integer ID to each token.
   ii. It then counts the occurrence of each of those tokens.

   If we set stop words to English then all the words in the document set that matches the list of the stop words defined by scikit learn will be removed because they are insignificant. Examples of stop words are: 'a', 'as', 'at', 'be', 'by'

7. We first separate the data into train/test and then generate a document-term matrix based on the training dataset and afterward generate a matrix for the test set.

   We first initialize the Count Vectorizer method and then fit the training data using "fit_transform" to get the document matrix. We first separate the train and test dataset and then generate the document term matrix because we set the parameters through training and use these settings to generate the document matrix.

8.

```
documents1 =['Hi, how are you?', 'Win money, win from home. Call now.', 'Hi., Call you now or tomorrow?']
count_vector.fit(documents1)
count_vector.get_feature_names_out()
doc_array = count_vector.transform(documents).toarray()
frequency_matrix = pd.DataFrame(doc_array, columns = count_vector.get_feature_names_out())
frequency_matrix
```

	are	call	from	hi	home	how	money	now	or	tomorrow	win	you
0	1	0	0	1	0	1	0	0	0	0	0	1
1	0	1	1	0	1	0	1	1	0	0	2	0
2	0	1	0	1	0	0	0	1	1	1	0	1

9. There are 7777 features created while making the document term matrix for the SMS dataset.

The features have to be combined or irrelevant features have to be excluded to reduce the number of features. The advantage is the complexity of the model decreases and the disadvantage is certain features might have their part in the model settings after training.

10. The input data here is discrete (word counts for text classification), hence we should use multinomial Naive Bayes implementation as it is suitable for classification with discrete features