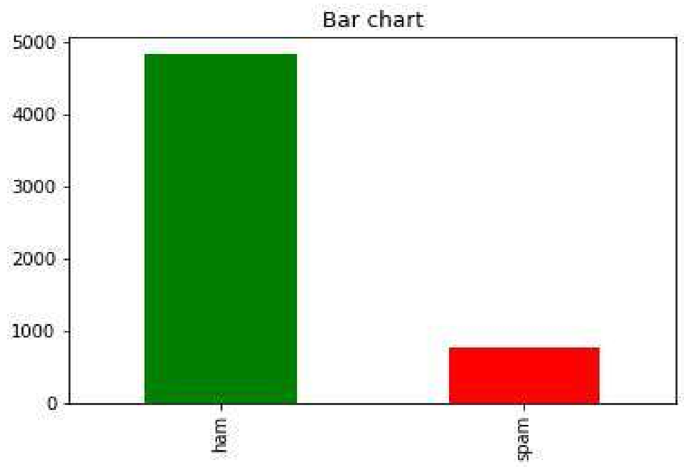
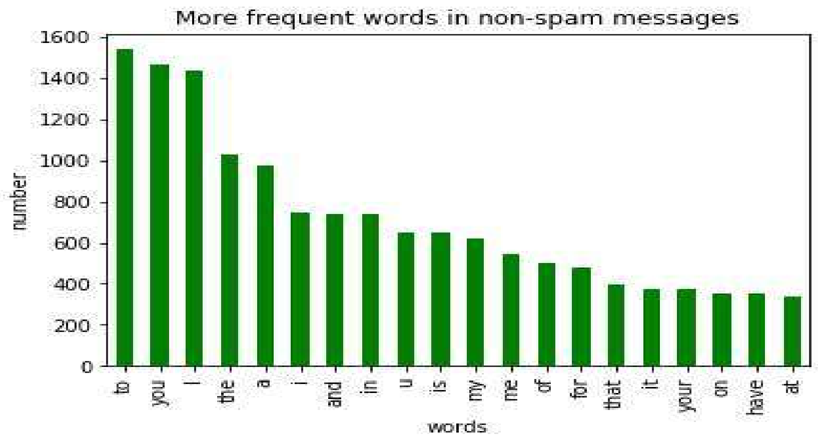
## DISTRIBUTION SPAM/NON-SPAM PLOTS

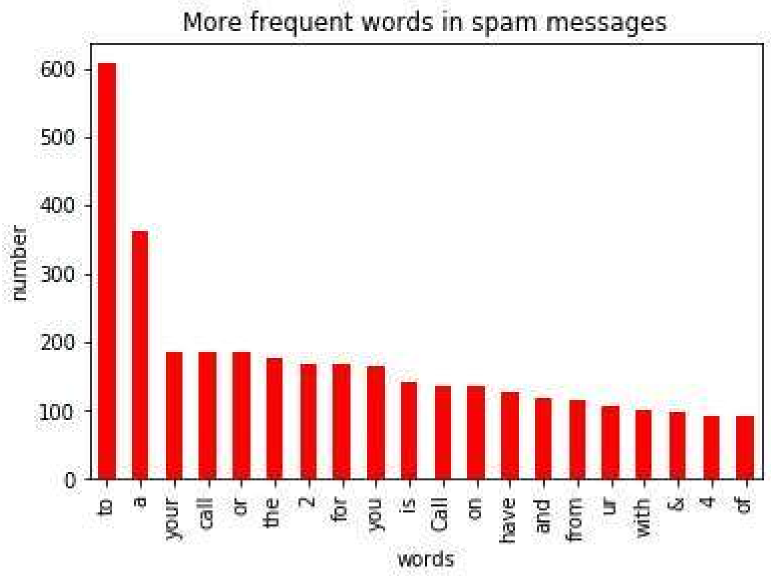
```
count_Class=pd.value_counts(data["v1"], sort= True)
count_Class.plot(kind= 'bar', color= ["Green", "red"])
plt.title('Bar chart')
plt.show()
```



```
[15] df1.plot.bar(legend = False, color='green')
     y_pos = np.arange(len(df1["words in non-spam"]))
     plt.xticks(y_pos, df1["words in non-spam"])
     plt.title('More frequent words in non-spam messages')
     plt.xlabel('words')
     plt.ylabel('number')
     plt.show()
```



```
[16] df2.plot.bar(legend = False, color = 'red')
     y_pos = np.arange(len(df2["words in spam"]))
     plt.xticks(y_pos, df2["words in spam"])
     plt.title('More frequent words in spam messages')
     plt.xlabel('words')
     plt.ylabel('number')
     plt.show()
```



```
[20]
    list_alpha = np.arange(1/100000, 20, 0.11)
    score_train = np.zeros(len(list_alpha))
    score_test = np.zeros(len(list_alpha))
    recall_test = np.zeros(len(list_alpha))
    precision_test= np.zeros(len(list_alpha))
    count = 0
```

```
[21] for alpha in list_alpha:
         bayes = naive_bayes.MultinomialNB(alpha=alpha)
         bayes.fit(X_train, y_train)
         score_train[count] = bayes.score(X_train, y_train)
         score_test[count]= bayes.score(X_test, y_test)
         recall_test[count] = metrics.recall_score(y_test, bayes.predict(X_test))
         precision_test[count] = metrics.precision_score(y_test, bayes.predict(X_test))
         count = count + 1
```