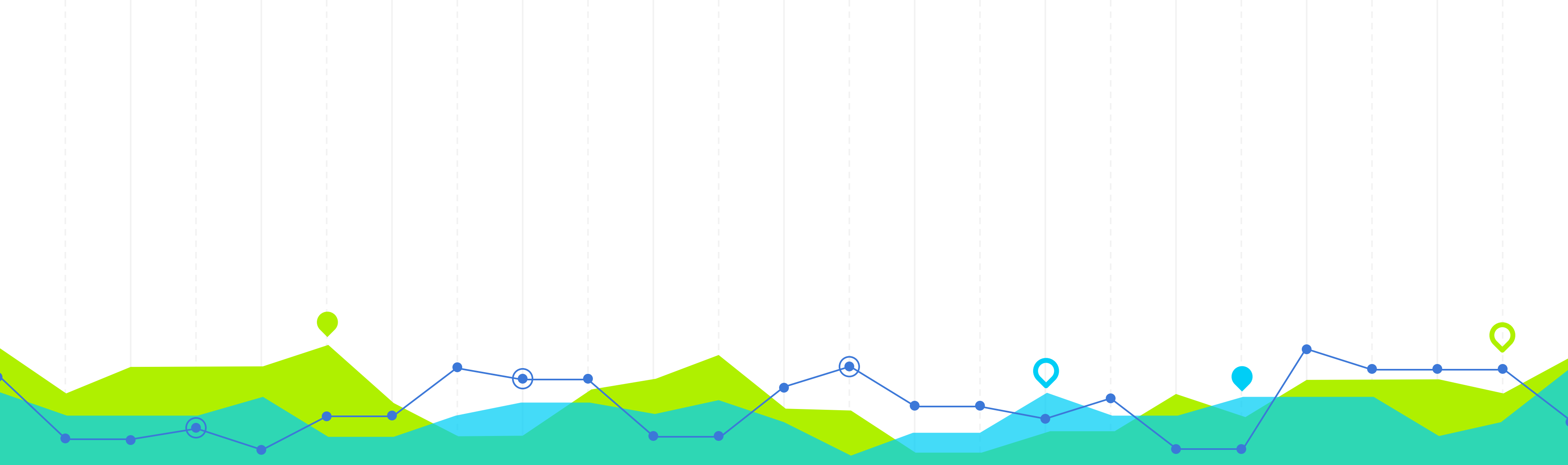
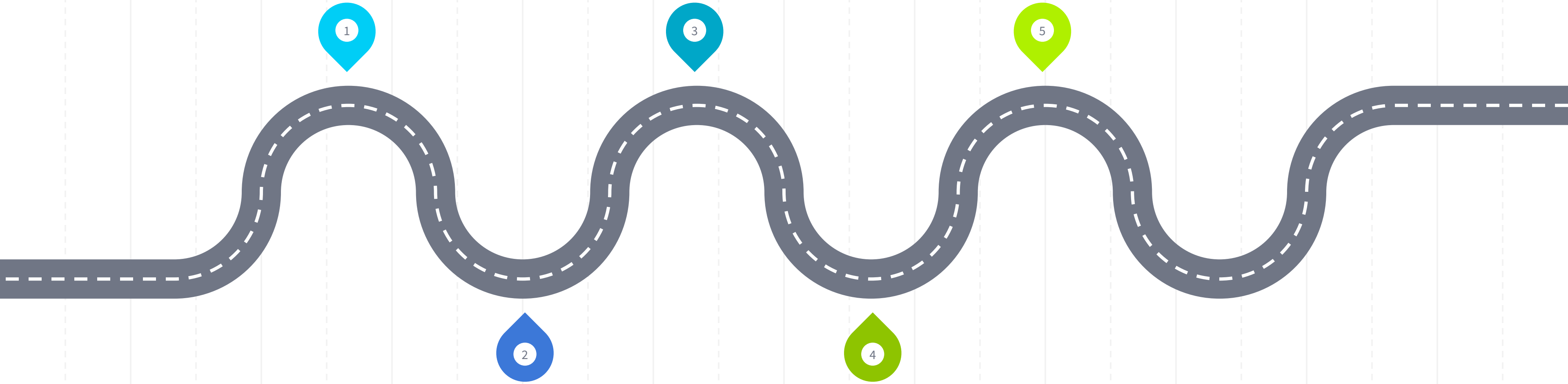


DATA ANALYSIS -
- Spam Classification DATASET

ROADMAP



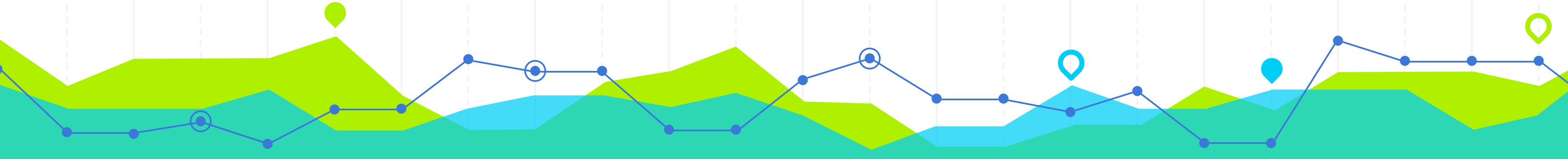
DATA CLEANING

DATA HANDLING

DATA WRANGLING

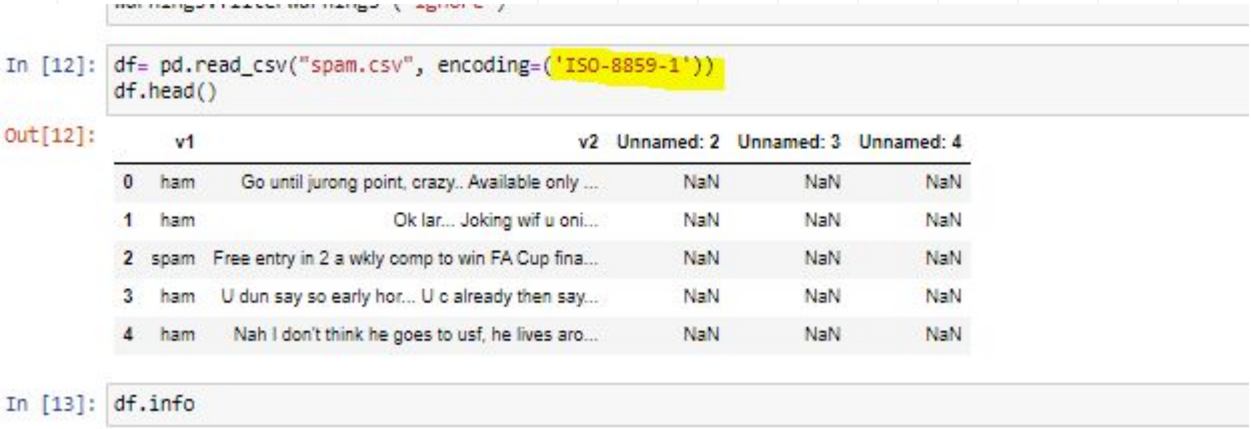
DATA TRANSFORMATION

DATA ANLYSIS &
OBSERVATION



INITIAL DATA CLEANING

Converted to ISO-8859-1 as UTF 8 was creating issues



```
warnings.filterwarnings('ignore')
```

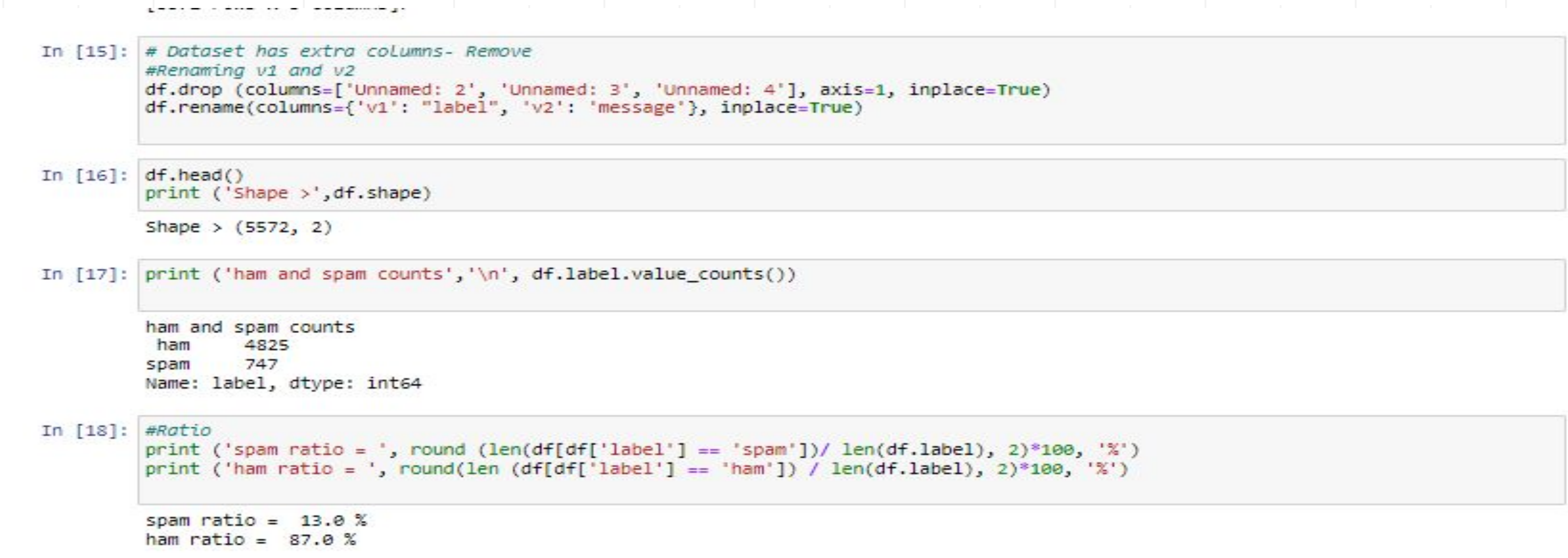
```
In [12]: df= pd.read_csv("spam.csv", encoding=('ISO-8859-1'))
         df.head()
```

Out[12]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

```
In [13]: df.info
```

INITIAL DATA CLEANING

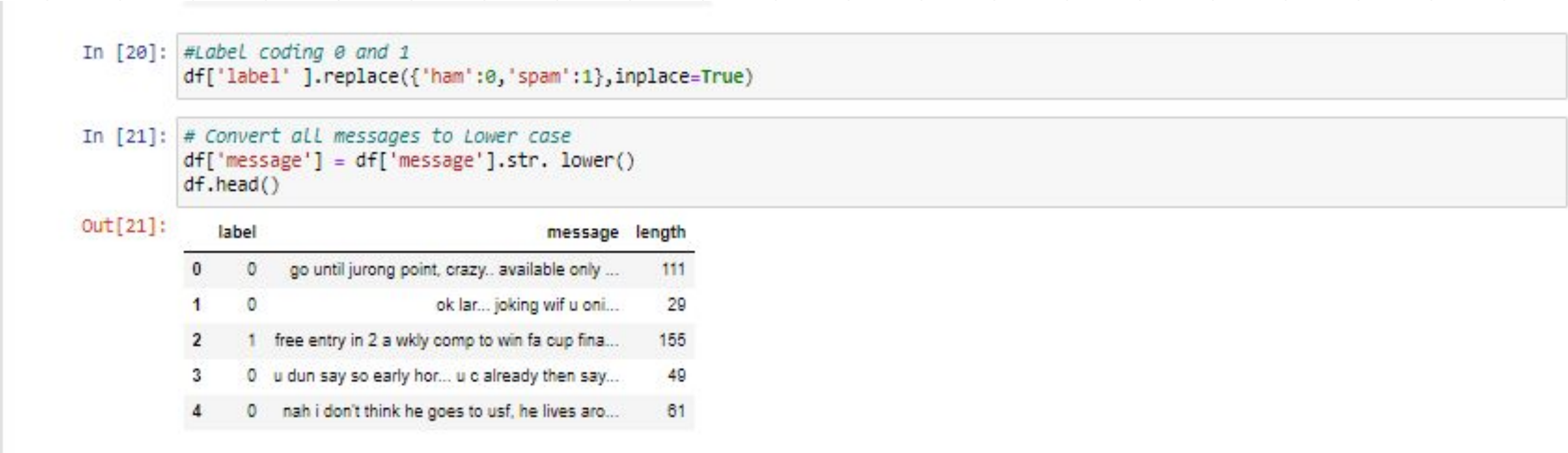


```
[5572 rows x 5 columns]

In [15]: # Dataset has extra columns- Remove
         #Renaming v1 and v2
         df.drop (columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
         df.rename(columns={'v1': "label", 'v2': 'message'}, inplace=True)

In [16]: df.head()
         print ('Shape >',df.shape)

         Shape > (5572, 2)

In [17]: print ('ham and spam counts','\n', df.label.value_counts())

         ham and spam counts
          ham      4825
         spam      747
         Name: label, dtype: int64

In [18]: #Ratio
         print ('spam ratio = ', round (len(df[df['label'] == 'spam'])/ len(df.label), 2)*100, '%')
         print ('ham ratio = ', round(len (df[df['label'] == 'ham']) / len(df.label), 2)*100, '%')

         spam ratio =  13.0 %
         ham ratio =  87.0 %
```

Labeling the data, finding out the spam and ham message ratio

INITIAL DATA CLEANING

Used label encoding to bifurcate into ham and spam messages. Also converted the messages to lower case.



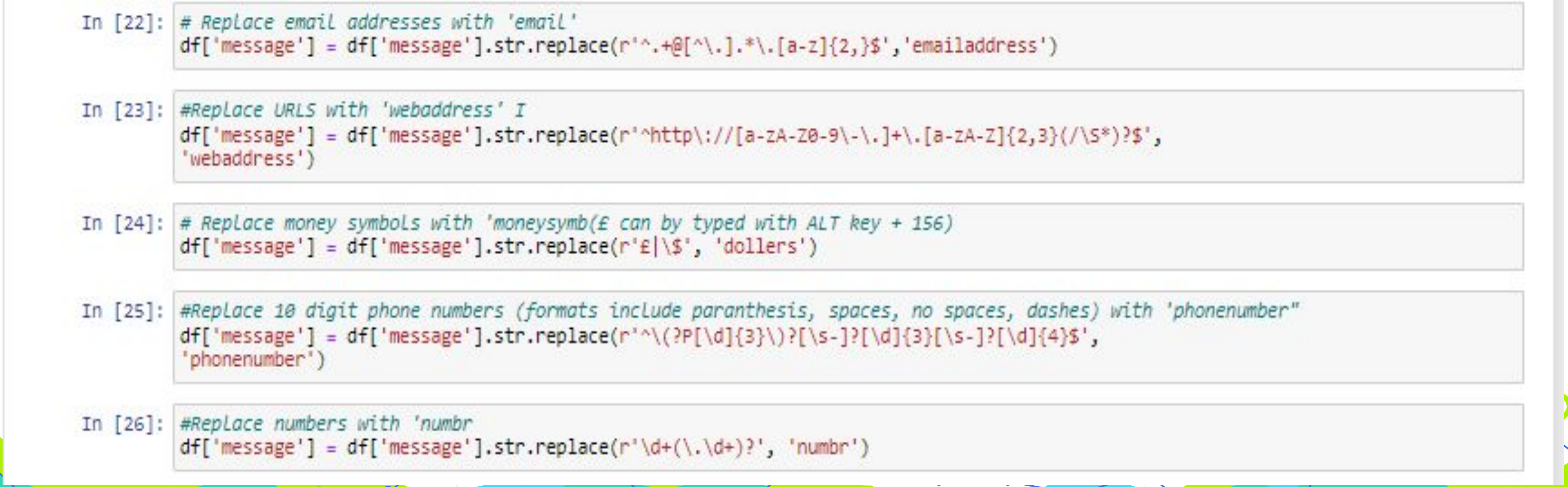
```
In [20]: #Label coding 0 and 1
         df['label' ].replace({'ham':0,'spam':1},inplace=True)

In [21]: # Convert all messages to Lower case
         df['message'] = df['message'].str. lower()
         df.head()

Out[21]:
```

	label	message	length
0	0	go until jurong point, crazy.. available only ...	111
1	0	ok lar... joking wif u oni...	29
2	1	free entry in 2 a wkly comp to win fa cup fina...	155
3	0	u dun say so early hor... u c already then say...	49
4	0	nah i don't think he goes to usf, he lives aro...	61

DATA CLEANING & WRANGLING

Replaced the url in the messages with -’webaddress’

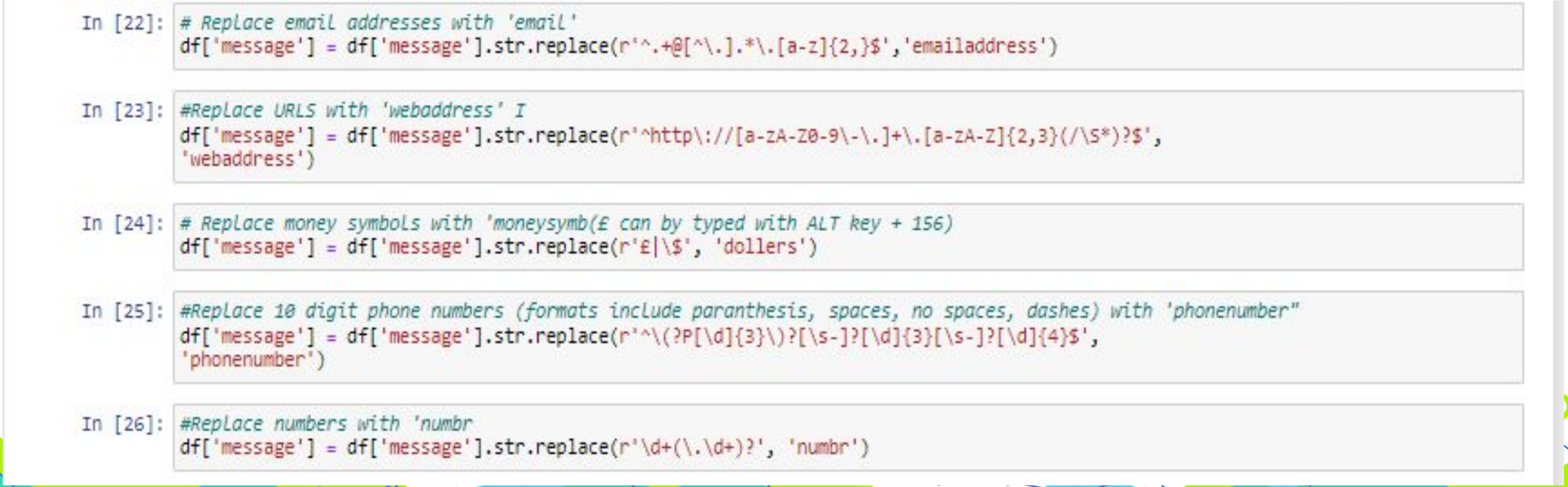
Replaced numbers with ‘numbr’



```
In [22]: # Replace email addresses with 'email'
         df['message'] = df['message'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

In [23]: #Replace URLS with 'webaddress' I
         df['message'] = df['message'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
         'webaddress')

In [24]: # Replace money symbols with 'moneysymb(£ can by typed with ALT key + 156)
         df['message'] = df['message'].str.replace(r'£|\$', 'dollers')

In [25]: #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber"
         df['message'] = df['message'].str.replace(r'^\(?P[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
         'phonenumber')

In [26]: #Replace numbers with 'numbr
         df['message'] = df['message'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

DATA CLEANING & WRANGLING

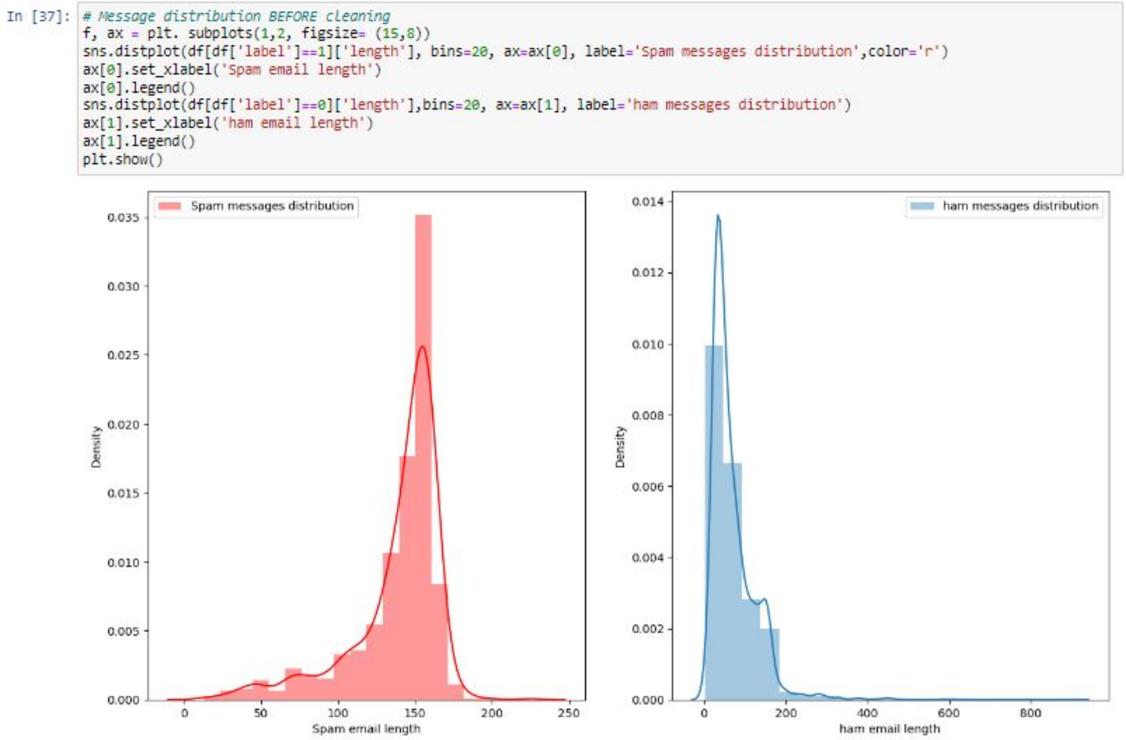
Remove stop words with nltk library



```
In [22]: # Replace email addresses with 'email'
         df['message'] = df['message'].str.replace(r'^.+@[^\.].*\.[a-z]{2,}$','emailaddress')

In [23]: #Replace URLS with 'webaddress' I
         df['message'] = df['message'].str.replace(r'^http\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(/\S*)?$',
         'webaddress')

In [24]: # Replace money symbols with 'moneysymb(£ can by typed with ALT key + 156)
         df['message'] = df['message'].str.replace(r'£|\$', 'dollers')

In [25]: #Replace 10 digit phone numbers (formats include paranthesis, spaces, no spaces, dashes) with 'phonenumber"
         df['message'] = df['message'].str.replace(r'^\(?P[\d]{3}\)?[\s-]?[\d]{3}[\s-]?[\d]{4}$',
         'phonenumber')

In [26]: #Replace numbers with 'numbr
         df['message'] = df['message'].str.replace(r'\d+(\.\d+)?', 'numbr')
```

DATA CLEANING & WRANGLING

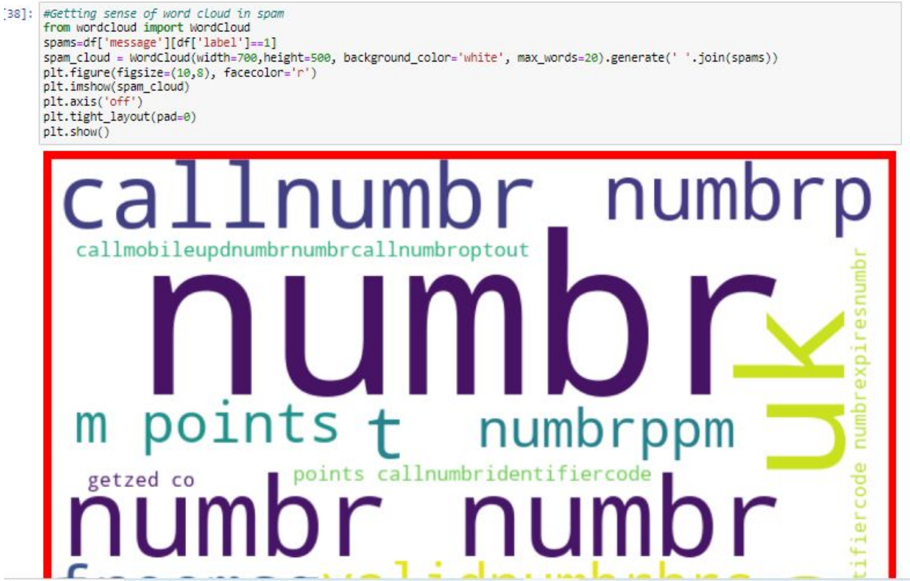
Original length vs clean length



```
In [37]: # Message distribution BEFORE cleaning
         f, ax = plt. subplots(1,2, figsize= (15,8))
         sns.distplot(df[df['label']==1]['length'], bins=20, ax=ax[0], label='Spam messages distribution',color='r')
         ax[0].set_xlabel('Spam email length')
         ax[0].legend()
         sns.distplot(df[df['label']==0]['length'],bins=20, ax=ax[1], label='ham messages distribution')
         ax[1].set_xlabel('ham email length')
         ax[1].legend()
         plt.show()
```

DATA ANALYSIS

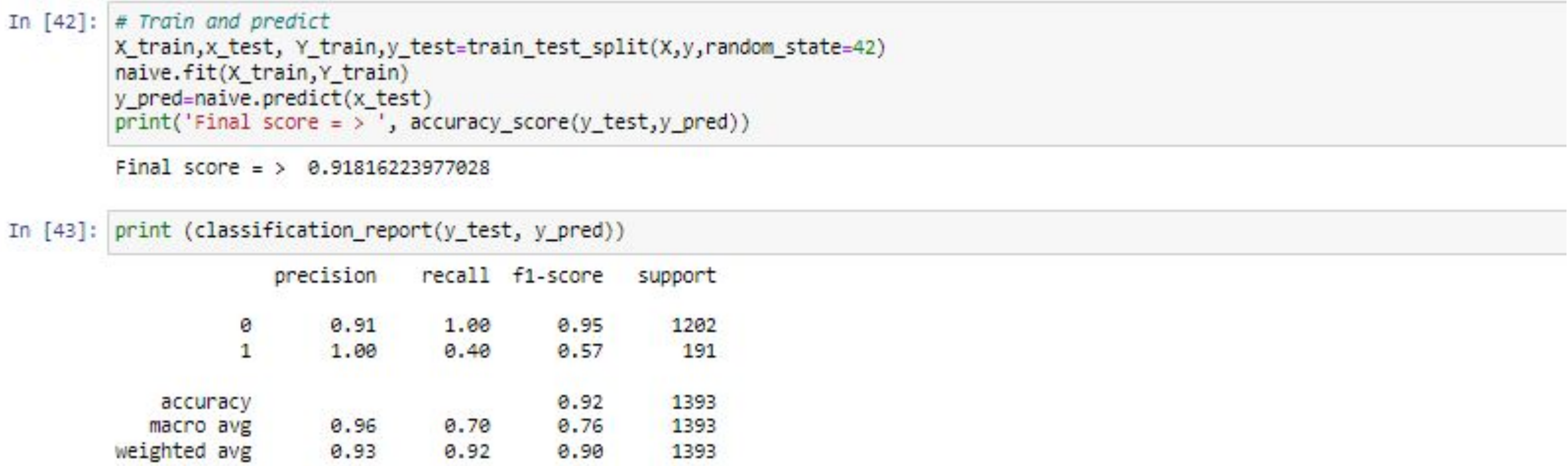
Some of the common spam words revealed with word cloud



```
[38]: #Getting sense of word cloud in spam
from wordcloud import WordCloud
spams=df['message'][df['label']==1]
spam_cloud = WordCloud(width=700,height=500, background_color='white', max_words=20).generate(' '.join(spams))
plt.figure(figsize=(10,8), facecolor='r')
plt.imshow(spam_cloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

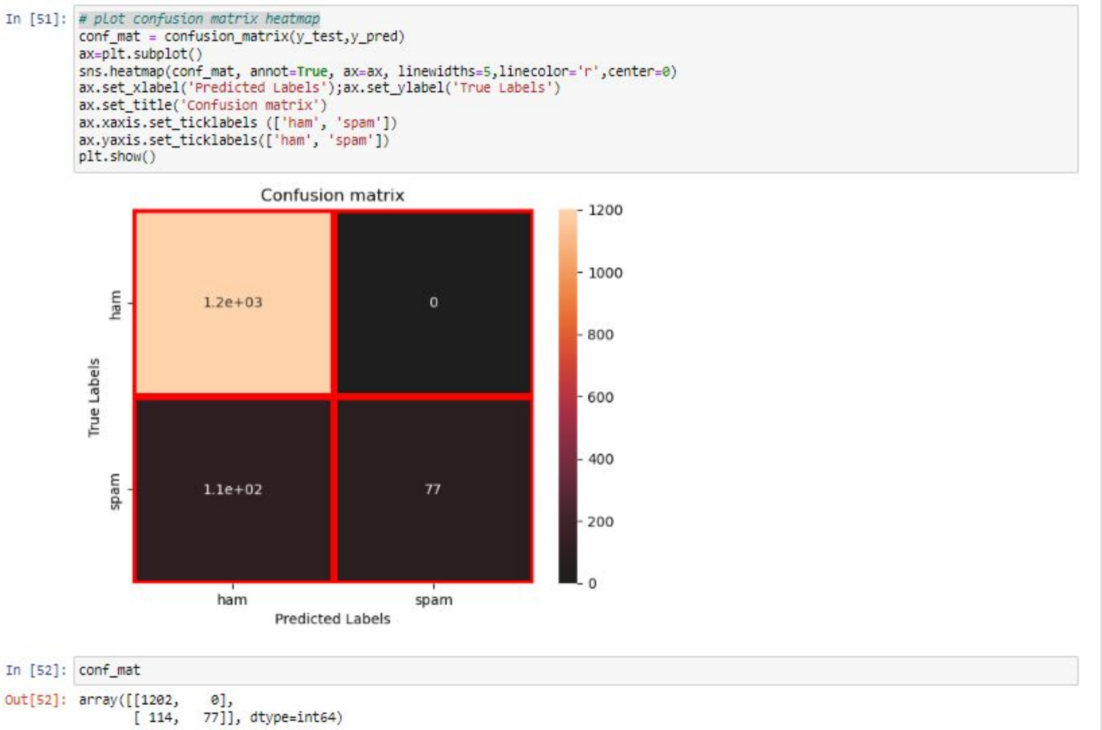
DATA ANALYSIS

Accuracy score and classification report



```
In [42]: # Train and predict
         X_train,x_test, Y_train,y_test=train_test_split(X,y,random_state=42)
         naive.fit(X_train,Y_train)
         y_pred=naive.predict(x_test)
         print('Final score = > ', accuracy_score(y_test,y_pred))

         Final score = >  0.91816223977028

In [43]: print (classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       0.91      1.00      0.95      1202
                    1       1.00      0.40      0.57       191

             accuracy                           0.92      1393
            macro avg       0.96      0.70      0.76      1393
         weighted avg       0.93      0.92      0.90      1393
```

DATA ANALYSIS-CORRELATION MATRIX



```
In [51]: # plot confusion matrix heatmap
         conf_mat = confusion_matrix(y_test,y_pred)
         ax=plt.subplot()
         sns.heatmap(conf_mat, annot=True, ax=ax, linewidths=5,linecolor='r',center=0)
         ax.set_xlabel('Predicted Labels');ax.set_ylabel('True Labels')
         ax.set_title('Confusion matrix')
         ax.xaxis.set_ticklabels (['ham', 'spam'])
         ax.yaxis.set_ticklabels(['ham', 'spam'])
         plt.show()
```

```
In [52]: conf_mat

Out[52]: array([[1202,    0],
                [ 114,   77]], dtype=int64)
```

THANKS!