Import Lib

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

# importing Stopwords
import nltk
from nltk.corpus import stopwords
import string

# models
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

# train test split
from sklearn.model_selection import train_test_split, GridSearchCV

# Pipeline
from sklearn.pipeline import Pipeline

# score
from sklearn.metrics import confusion_matrix,classification_report,ConfusionMatrixDisplay
from sklearn.metrics import accuracy_score
```

Import Data

```
df=pd.read_csv('/kaggle/input/spam-ham-dataset/spam_dataset.csv')
```

Basic Analysis

```
df.head()
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
df.describe()
```

	Unnamed: 0	label_num
count	5171.000000	5171.000000
mean	2585.000000	0.289886
std	1492.883452	0.453753
min	0.000000	0.000000
25%	1292.500000	0.000000
50%	2585.000000	0.000000
75%	3877.500000	1.000000
max	5170.000000	1.000000

```
df['label'].value_counts()
```

```
label
ham     3672
spam    1499
Name: count, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

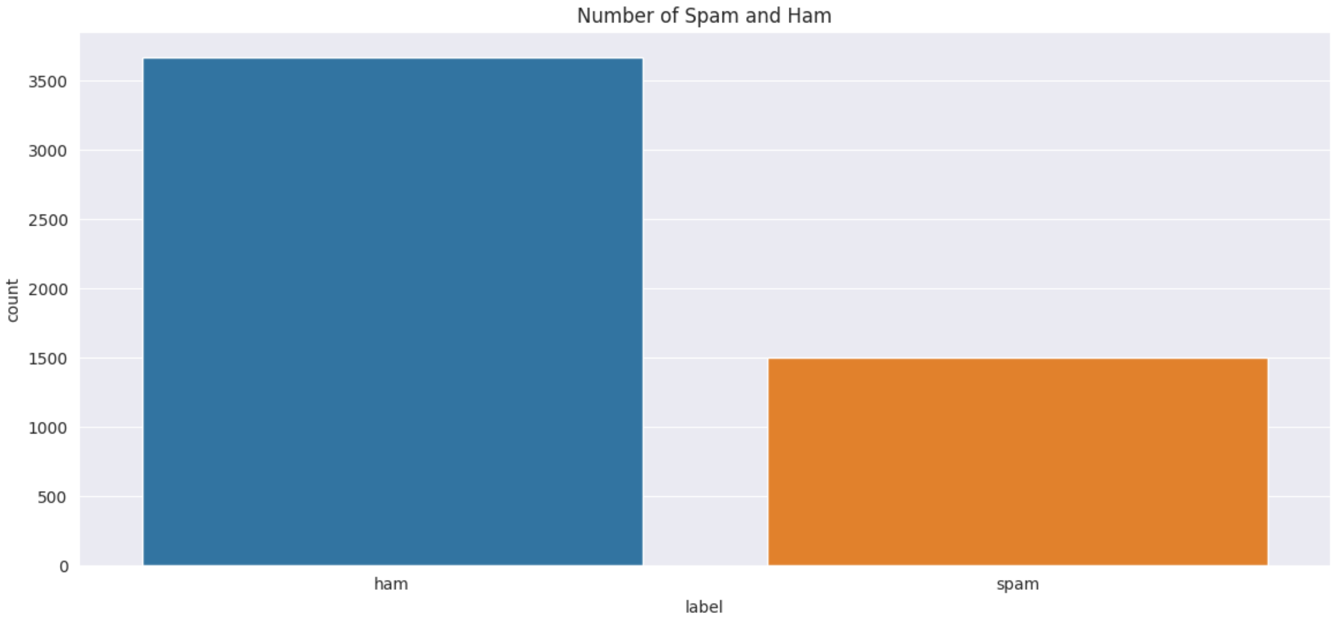
```
# adding new column as length of the text
df['length'] = df['text'].apply(len)
df.head()
```

	Unnamed: 0	label	text	label_num	length
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0	327
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0	97
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0	2524
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1	414
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0	336

EDA

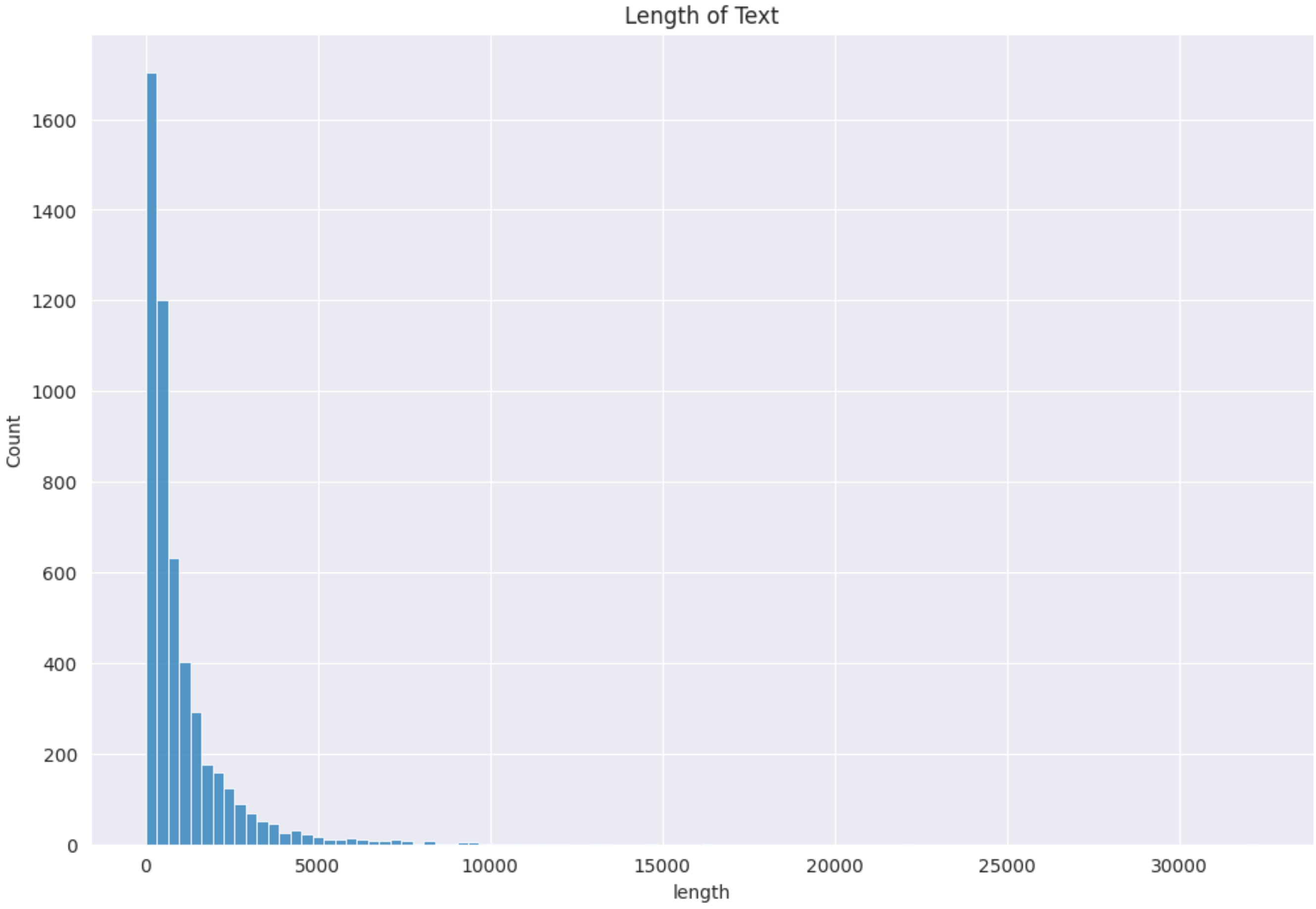
```
# plot for count of spam and ham in data
plt.figure(figsize=(14,6))
sns.set_style('darkgrid')
sns.countplot(x='label',data=df)
plt.title('Number of Spam and Ham')
```

```
Text(0.5, 1.0, 'Number of Spam and Ham')
```



```
# Plot for distribution lenth of text
plt.figure(figsize=(12,8))
sns.histplot(x='length',data=df,bins=100)
plt.title('Length of Text')
```

```
Text(0.5, 1.0, 'Length of Text')
```

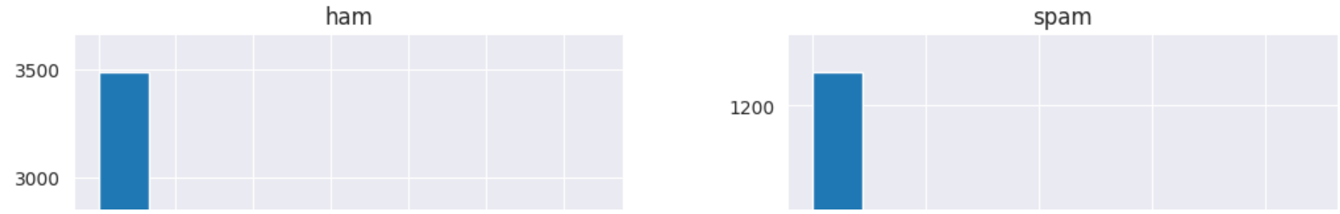


```
# maximum lenth text
df[df['length']==df['length'].max()]['text']
```

```
949    Subject: fw : " red , white and blue out "\r\n...
Name: text, dtype: object
```

```
# distribution of spam and ham by length of text
df.hist(column='length',by='label',figsize=(12,8))
```

```
array([<Axes: title={'center': 'ham'}>, <Axes: title={'center': 'spam'}>],
      dtype=object)
```



Feature Enginering

```
# function to remove punctuation and stopwords
def text_process(text):
    non_punc = [char for char in text if char not in string.punctuation]
    non_punc=''.join(non_punc)
    return [word for word in non_punc.split() if word not in stopwords.words('english')]
```

Train Test Split

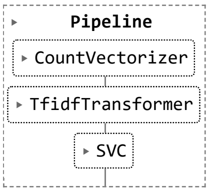
```
# define X(features),y(target)
X= df['text']
y=df['label']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Modles

```
# creating a pipline to model the data
# pipeline for MultinomialNB
pipe_mnb = Pipeline([
    ('bow',CountVectorizer(analyzer=text_process)),
    ('tf',TfidfTransformer()),
    ('classifier',MultinomialNB())
])

# pipeline for Random Forest Classifier
pipe_rf =Pipeline([
    ('bow',CountVectorizer(analyzer=text_process)),
    ('tf',TfidfTransformer()),
    ('classifier',RandomForestClassifier())
])

# pipeline for Random Forest Classifier
pipe_svc =Pipeline([
    ('bow',CountVectorizer(analyzer=text_process)),
    ('tf',TfidfTransformer()),
    ('classifier',SVC())
])
```

```
# fit the data
pipe_mnb.fit(X_train,y_train)
pipe_rf.fit(X_train,y_train)
pipe_svc.fit(X_train,y_train)
```



```
# predict the target feature
pred_mnb = pipe_mnb.predict(X_test)
pred_rf = pipe_rf.predict(X_test)
pred_svc = pipe_svc.predict(X_test)
```

Prediction Accuracy

```
print('The accuracy for Multinomial Classifer:',accuracy_score(y_test,pred_mnb)*100)
print('The accuracy for Random_forest Classifer:',accuracy_score(y_test,pred_rf)*100)
print('The accuracy for SVC:',accuracy_score(y_test,pred_svc)*100)
```

```
The accuracy for Multinomial Classifer: 91.73989455184535
The accuracy for Random_forest Classifer: 97.4223784417106
The accuracy for SVC: 98.82835383714118
```

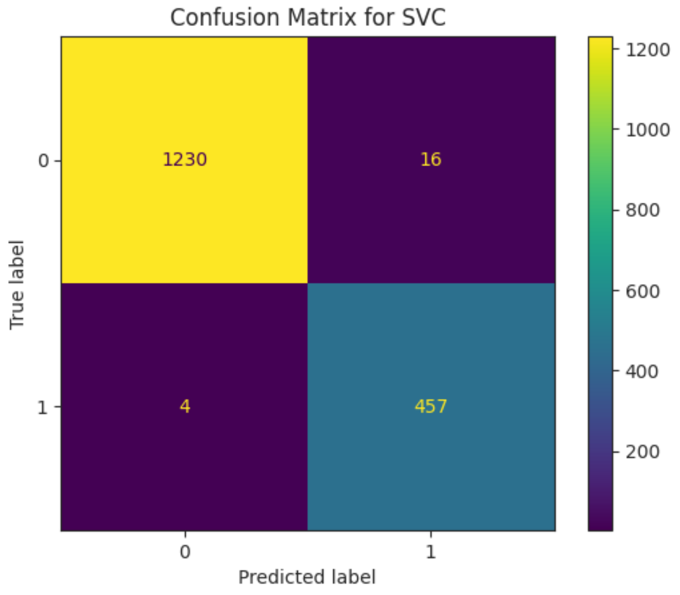
The SVC predicts better tham Random Forest Model and Multinomial.

```
# print confusion matrix and classification report
print ('Classification report on SVC:')
print('\n')
print(classification_report(y_test,pred_svc))
```

```
Classification report on SVC:


              precision    recall  f1-score   support

         ham       1.00      0.99      0.99      1246
        spam       0.97      0.99      0.98       461

    accuracy                           0.99      1707
   macro avg       0.98      0.99      0.99      1707
weighted avg       0.99      0.99      0.99      1707
```

```
# Display confusioni matrix for SVC

sns.set_style('ticks')
ConfusionMatrixDisplay(confusion_matrix(y_test,pred_svc)).plot()
plt.title("Confusion Matrix for SVC")
```

```
Text(0.5, 1.0, 'Confusion Matrix for SVC')
```



```
from sklearn.model_selection import cross_val_score

# Number of folds
k = 5

# Initialize the SVC model in the pipeline
pipe_svc.set_params(classifier=SVC())

# Perform k-fold cross-validation
cv_scores = cross_val_score(pipe_svc, X, y, cv=k)

# Output the results
print(f'CV Scores for each fold: {cv_scores}')
print(f'Average CV Score: {np.mean(cv_scores)}')
```

```
    CV Scores for each fold: [0.98937198 0.98742747 0.99323017 0.98549323 0.99129594]
    Average CV Score: 0.9893637578373934
```