Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer #considers how common the word occur in the email
from sklearn.naive_bayes import GaussianNB, MultinomialNB #algorithms for classification tasks
from sklearn.metrics import classification_report, confusion_matrix
```

Reading data

```
df = pd.read_csv('/content/spam_ham_dataset.csv')
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

Added name to the first column as "ID"

```
df.rename(columns = {'Unnamed: 0': 'ID'},inplace = True)
```

```
df.head()
```

	ID	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

Generating Profile Report

```
!pip install ydata_profiling
```

```
Collecting ydata_profiling
  Downloading ydata_profiling-4.8.3-py2.py3-none-any.whl (359 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 359.5/359.5 kB 4.2 MB/s eta 0:00:00
Requirement already satisfied: scipy<1.14,>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.11.4)
Requirement already satisfied: pandas!=1.4.0,<3,>1.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (2.0.3)
Requirement already satisfied: matplotlib<3.9,>=3.2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (3.7.1)
Requirement already satisfied: pydantic>=2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (2.7.1)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (6.0.1)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (3.1.4)
Collecting visions[type_image_path]<0.7.7,>=0.7.5 (from ydata_profiling)
  Downloading visions-0.7.6-py3-none-any.whl (104 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 104.8/104.8 kB 4.6 MB/s eta 0:00:00
Requirement already satisfied: numpy<2,>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.25.2)
Collecting htmlmin==0.1.12 (from ydata_profiling)
```

```
    Downloading htmlmin-0.1.12.tar.gz (19 kB)
    Preparing metadata (setup.py) ... done
  Collecting phik<0.13,>=0.11.1 (from ydata_profiling)
    Downloading phik-0.12.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (686 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 686.1/686.1 kB 7.2 MB/s eta 0:00:00
  Requirement already satisfied: requests<3,>=2.24.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (2.31.0)
  Requirement already satisfied: tqdm<5,>=4.48.2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (4.66.4)
  Requirement already satisfied: seaborn<0.14,>=0.10.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.13.1)
  Collecting multimethod<2,>=1.4 (from ydata_profiling)
    Downloading multimethod-1.11.2-py3-none-any.whl (10 kB)
  Requirement already satisfied: statsmodels<1,>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.14.2)
  Collecting typeguard<5,>=3 (from ydata_profiling)
    Downloading typeguard-4.2.1-py3-none-any.whl (34 kB)
  Collecting imagehash==4.3.1 (from ydata_profiling)
    Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 4.8 MB/s eta 0:00:00
  Requirement already satisfied: wordcloud>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (1.9.3)
  Collecting dacite>=1.8 (from ydata_profiling)
    Downloading dacite-1.8.1-py3-none-any.whl (14 kB)
  Requirement already satisfied: numba<1,>=0.56.0 in /usr/local/lib/python3.10/dist-packages (from ydata_profiling) (0.58.1)
  Requirement already satisfied: PyWavelets in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata_profiling) (1.6.0)
  Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (from imagehash==4.3.1->ydata_profiling) (9.4.0)
  Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2<3.2,>=2.11.1->ydata_profiling)
  Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_profilin
  Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_profiling)
  Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_profili
  Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_profili
  Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_profiling
  Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_profilin
  Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib<3.9,>=3.2->ydata_prof
  Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba<1,>=0.56.0->ydata_pr
  Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.4.0,<3,>1.1->ydata_profiling)
  Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=1.4.0,<3,>1.1->ydata_profiling
  Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from phik<0.13,>=0.11.1->ydata_profiling)
  Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2->ydata_profiling)
  Requirement already satisfied: pydantic-core==2.18.2 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2->ydata_profiling)
  Requirement already satisfied: typing-extensions>=4.6.1 in /usr/local/lib/python3.10/dist-packages (from pydantic>=2->ydata_profiling
  Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_p
  Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profiling) (3
  Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profili
  Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata_profili
  Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels<1,>=0.13.2->ydata_profiling)
```

Profile Report

```
from ydata_profiling import ProfileReport
ProfileReport = ProfileReport(df)
ProfileReport
```

```
Summarize dataset: 100%                    14/14 [00:03<00:00,  3.62it/s, Completed]
Generate report structure: 100%                 1/1 [00:04<00:00,  4.91s/it]
Render HTML: 100%                          1/1 [00:00<00:00,  1.36it/s]
```

Overview

Dataset statistics

Number of variables	4
Number of observations	5171
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	161.7 KiB
Average record size in memory	32.0 B

Variable types

Numeric	1
Categorical	2
Text	1

Alerts

`ID` is highly overall correlated with `label` and 1 other fields (label, label_num)	High correlation

From the data report, we came to know the following things: -- Data has no missing values -- Each column is highly correlated to with each other. -- The number of ham(3672) emails is more than the spam(1499)

Data Statistics

```
df.describe()
```

	ID	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.shape
# 5171 rows and 4 columns
```

```
(5171, 4)
```

Finding Nunique values

```
df.nunique()
```

```
ID           5171
label           2
text         4993
label_num       2
dtype: int64
```

Adding a column length to check the length of messages

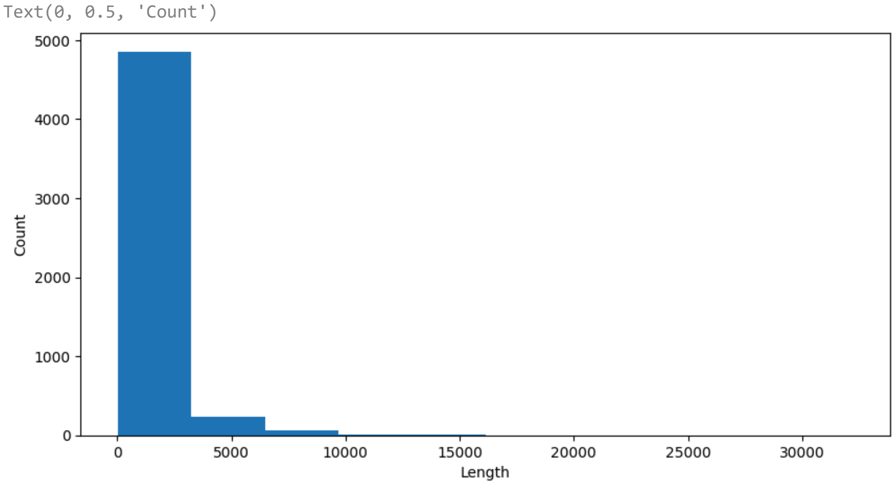
```
df['length'] = df['text'].apply(len)
df.head()
```

	ID	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

Distribution on column length to check which text has highest length

```
plt.figure(figsize = (10,5))
hist_graph = df['length'].plot(kind = 'hist')
hist_graph.set_xlabel("Length")
hist_graph.set_ylabel("Count")
```

```
Text(0, 0.5, 'Count')
```



The graph elaborates that there are some messages who length is above 4000 or upto 5000

With data statistics we are going to find the longest message in this dataset

```
df.length.describe()
```

```
count     5171.000000
mean      1048.390447
std       1528.514135
min         11.000000
25%        244.000000
50%        540.000000
75%       1237.000000
max      32258.000000
Name: length, dtype: float64
```

The highest message length is 32258 which clearly explains that it is a spam, because spam messages tend to have more text than ham data.

Pre Processing

Removing Punctuations

```
import string
text = df['text']
translator = str.maketrans('', '', string.punctuation)
## using maketrans method to gather punctuations
df['text'] = text.apply(lambda x: x.translate(translator))
#applying translate() to remove punctuations
df.head()
```

	ID	label	text	label_num	length
0	605	ham	Subject enron methanol meter 988291\r\nthis...	0	327
1	2349	ham	Subject hpl nom for january 9 2001\r\n see at...	0	97
2	3624	ham	Subject neon retreat\r\nho ho ho we re aroun...	0	2524
3	4685	spam	Subject photoshop windows office cheap mai...	1	414
4	2030	ham	Subject re indian springs\r\nthis deal is to ...	0	336

Data Test and Train split

```
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['label'], test_size=0.3, random_state=42)
# 30% data for testing and 70% for training
```

Vectorization

```
cv = CountVectorizer() # converts the words into tokenization, converts raw text into numerical representation that machine learning underst
X_train = cv.fit_transform(X_train)
X_test = cv.transform(X_test)
```

TF-IDF

```
tfidf_transformer = TfidfTransformer()
X_train = tfidf_transformer.fit_transform(X_train)
X_test = tfidf_transformer.transform(X_test)
# training and transforming the training dataset into TF-IDF
```

Training model on Naive Bayes

```
Gnb = GaussianNB()   #ensures that the probability calculated is normal
Mnb = MultinomialNB()
Gnb_model = Gnb.fit(X_train.toarray(), y_train) #The toarray() method is used on sparse matrices((when there are many 0s in model) in scikit
Mnb_model = Mnb.fit(X_train.toarray(), y_train)
```

Prediction

```
print("Prediction for Gaussian Naive Bayes:", Gnb_model.predict(X_test.toarray()))
```

```
Predicted: ['ham' 'spam' 'ham' ... 'ham' 'spam' 'ham']
```

```
print("Prediction for Multinomial Naive Bayes:", Mnb_model.predict(X_test.toarray()))
```

```
Prediction for Multinomial Naive Bayes: ['ham' 'spam' 'ham' ... 'ham' 'ham' 'ham']
```

Classification Report

Gaussian Naivr Bayes

```
print(classification_report(y_test, Gnb_model.predict(X_test.toarray())))
```

```
              precision    recall  f1-score   support

         ham       0.96      0.97      0.97      1121
        spam       0.92      0.90      0.91       431

    accuracy                           0.95      1552
   macro avg       0.94      0.94      0.94      1552
weighted avg       0.95      0.95      0.95      1552
```

Multinomial Algorithm

```
print(classification_report(y_test, Mnb_model.predict(X_test.toarray())))
```

```
              precision    recall  f1-score   support

         ham       0.85      1.00      0.92      1121
        spam       1.00      0.52      0.69       431

    accuracy                           0.87      1552
   macro avg       0.92      0.76      0.80      1552
weighted avg       0.89      0.87      0.85      1552
```

Overall the accuracy of Gaussian Naive Bayes is more than Multinomial Bayes i.e 95% but we cannot rely on accuracy only so moving forward to analyze th confusion matrix

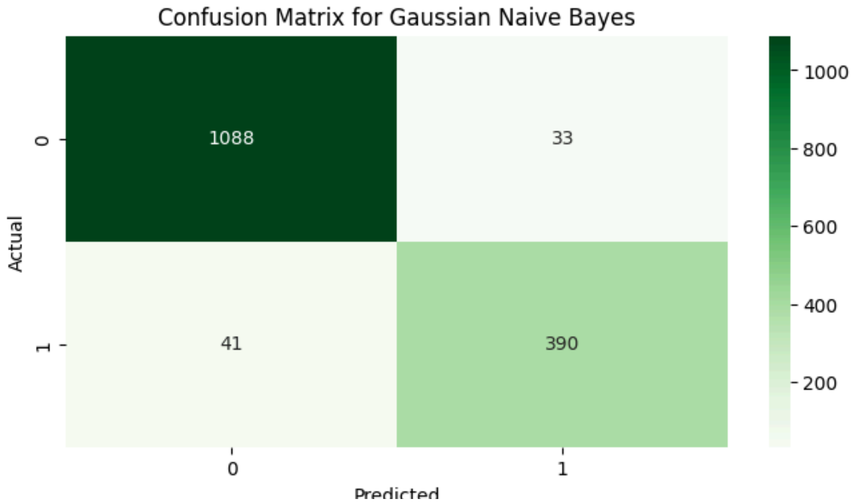
Confusion Matrix for Gnb

```
confusion_matrix(y_test, Gnb_model.predict(X_test.toarray()))
```

```
array([[1088,   33],
       [  41,  390]])
```

Plotting Confusion matrix

```
plt.figure(figsize = ( 8,4))
sns.heatmap(confusion_matrix(y_test, Gnb_model.predict(X_test.toarray())), annot = True, fmt = 'd', cmap = 'Greens')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix for Gaussian Naive Bayes')
plt.show()
```



- 1088 instances are correctly classified as "Ham"(True Negative)
- 33 instances were incorrectly classified as "Ham (False Positive)
- 41 were incorrectly classified "Spam"
- 390 were correctly classified "Spam"

Confusion matrix for multinomial bayes

```
confusion_matrix(y_test, Mnb_model.predict(X_test.toarray()))
```

```
array([[1121,    0],
       [ 205,  226]])
```

Plotting

```
plt.figure(figsize= (8,4))
sns.heatmap(confusion_matrix(y_test, Mnb_model.predict(X_test.toarray())), annot = True, fmt = 'd', cmap = 'Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('Actual Labels')
plt.title('Confusion Matrix for Multinomial Naive Bayes')
plt.show()
```