SPAM EMAIL DETECTION

Problem Definition

A spam detection is a program used to detect unsolicited, unwanted and virus-infected emails and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for specific criteria on which to base its judgments.

Datasets used

Dataset - 1

https://www.kaggle.com/code/surekharamireddy/spam-detection-with-99-accuracy/data

This Dataset Contains Three Columns Email subject, Email message, Label (1 = spam, 0 = not spam)

Dataset - 2

https://www.kaggle.com/datasets/studymart/spam-email-detection-dataset

The csv file composes of two (2) columns: text and spam

Dataset - 3

https://www.kaggle.com/code/mfaisalqureshi/email-spam-detection-98-accuracy/data

This Dataset Contains Two Columns Category (spam or ham) and Message

Dataset - 4

https://www.kaggle.com/datasets/ganiyuolalekan/spam-assassin-email-classification-dataset

The Spam Assassin Dataset is a a selection of mail messages, suitable for use in testing spam filtering systems. This particular set was obtained from the Apache Public Datasets, cleaned and organized into a csv file in a manner which it can be convenient to use.

The csv file composes of two (2) columns: text and target

ML Algorithms

The Algorithms we have used in this project are:

- **Logistic Regression**
- **Random Forest**
- **Support Vector Classifiers (SVCs)**
- **K-nearest Neighbours (KNN)**
- **Bagging Classifier**
- **Decision Tree**

Logistic Regression: it is a simple and widely used algorithm that is often used for binary classification tasks. It can be used to predict whether a student is likely to be accepted or rejected based on their scores and college ranking.

Random Forest: Random forests are an ensemble learning method that combines multiple decision trees to make more accurate predictions. They can be used in the same way as decision trees to predict a student's likelihood of acceptance based on their scores and college ranking.

Support Vector Classifiers(SVCs): These are type of an algorithm that can be used for classification tasks. They work by finding the hyperplane in the feature space that maximally separates the different classes in the data. This allows the algorithm to make predictions based on the distance of new data points to the hyperplane. support vector classifier could potentially be a useful algorithm to consider. It could be trained on the data to predict a student's likelihood of acceptance based on their scores and college ranking

K-nearest Neighbours (KNN): is a classification algorithm that is based on the idea of using the "k" closest data points in the feature space to make predictions for new data. It works by calculating the distances between the new data point and the "k" nearest points in the training set, and then using these distances to determine the class of the new data point. KNN classifier could potentially be a useful algorithm to consider. It could be trained on the data to predict a student's likelihood of acceptance based on their scores and college ranking

Bagging Classifier: A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Decision Tree: Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

Data Pre-processing

Find the columns with only Null values

```
In [2]: data.isnull().sum()

Out[2]: subject    62
        message     0
        label       0
        dtype: int64
```

Count the No of Non-NA cells for each column or row

DATASET 1

```
In [3]: data.count()

Out[3]: subject    2831
        message    2893
        label      2893
        dtype: int64
```

DATASET 2

```
In [3]: data.count()

Out[3]: text     5726
        spam     5726
        dtype: int64
```

DATASET 3

In [3]:
```
data.count()
```
Out[3]:
```
Category    5572
Message     5572
dtype: int64
```

DATASET 4

In [3]:
```
data.count()
```
Out[3]:
```
text      5796
target    5796
dtype: int64
```

List of column in the dataset which has only NULL values

In [4]:
```
Null_Data = data.isnull().sum()
Rows=data.shape[0]
Null_Columns = []
for i in range(len(Null_Data)):
    if Null_Data[i] == Rows - 1 or Null_Data[i] == Rows:
        Null_Columns.append(Column_Names[i])
Null_Columns
```
Out[4]:
```
[]
```

Drop the columns with only Null values

In [5]:
```
for i in Null_Columns:
    del data[i]
data
```

DATASET 1

Out[5]:

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
1	NaN	lang classification grimes , joseph e . and ba...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0
...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0

2893 rows × 3 columns

DATASET 2

Out[10]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1
...	...	...
5721	Subject: re : research and development charges...	0
5722	Subject: re : receipts from visit jim , than...	0
5723	Subject: re : enron case study update wow ! a...	0
5724	Subject: re : interest david , please , call...	0
5725	Subject: news : aurora 5 . 2 update aurora ve...	0

5726 rows × 2 columns

DATASET 3

Out[5]:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

DATASET 4

Out[5]:

	text	target
0	From ilug-admin@linux.ie Mon Jul 29 11:28:02 2...	0
1	From gort44@excite.com Mon Jun 24 17:54:21 200...	1
2	From fork-admin@xent.com Mon Jul 29 11:39:57 2...	1
3	From dcm123@btamail.net.cn Mon Jun 24 17:49:23...	1
4	From ilug-admin@linux.ie Mon Aug 19 11:02:47 2...	0
...	...	...
5791	From ilug-admin@linux.ie Mon Jul 22 18:12:45 2...	0
5792	From fork-admin@xent.com Mon Oct 7 20:37:02 20...	0
5793	Received: from hq.pro-ns.net (localhost [127.0...	1
5794	From razor-users-admin@lists.sourceforge.net T...	0
5795	From rssfeeds@jmason.org Mon Sep 30 13:44:10 2...	0

5796 rows × 2 columns

Find the rows with any Null values

In [6]:
```
data.isnull().any()
```

Out[6]:
```
Category    False
Message     False
dtype: bool
```

DATASET 1

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
subject    62
message     0
label       0
dtype: int64
```

DATASET 2

In [12]:
```
data.isnull().sum()
```

Out[12]:
```
text    0
spam    0
dtype: int64
```

DATASET 3

```
In [7]: data.isnull().sum()
```

```
Out[7]: Category    0
        Message     0
        dtype: int64
```

DATASET 4

```
In [7]: data.isnull().sum()
```

```
Out[7]: text      0
        target    0
        dtype: int64
```

Rows which has one or more NULL values in it

DATASET 1

```
In [14]: data[data.isnull().any(axis=1)]
```

Out[14]:

	subject	message	label
1	NaN	lang classification grimes , joseph e . and ba...	0
13	NaN	syntax the antisymmetry of syntax richard s	0
69	NaN	computational ling bengt sigurd (ed) compute...	0
107	NaN	phonology & phonetics burquest , donald a . an...	0
258	NaN	phonology & phonetics leiden in last : hil pho...	0
...	...	...	...
2296	NaN	the latest issue (1994 n01) of etudes de let...	0
2309	NaN	b a r g a i n a i r f a r e s your 1 - stop tr...	1
2555	NaN	data to : = 20 date : fri , 06 feb 1998 22 : 3...	1
2562	NaN	epac . pt , e . carnoali @ genie . com , e . c...	1
2811	NaN	simply send a message to : listserv @ tamvm1	0

62 rows × 3 columns

DATASET 2

```
In [13]: data[data.isnull().any(axis=1)]
```

Out[13]:

text	spam

DATASET 3

```
In [8]: data[data.isnull().any(axis=1)]
```

Out[8]:

Category	Message

DATASET 4

In [13]:
```
data[data.isnull().any(axis=1)]
```

Out[13]:

text	spam

Drop the rows with any Null values

DATASET 1

In [9]:
```
data.dropna(inplace=True)
data.isnull().any()
```

Out[9]:
```
subject    False
message    False
label      False
dtype: bool
```

DATASET 2

In [14]:
```
data.dropna(inplace=True)
data.isnull().any()
```

Out[14]:
```
text    False
spam    False
dtype: bool
```

DATASET 3

In [9]:
```
data.dropna(inplace=True)
data.isnull().any()
```

Out[9]:
```
Category    False
Message     False
dtype: bool
```

DATASET 4

In [9]:
```
data.dropna(inplace=True)
data.isnull().any()
```

Out[9]:
```
text      False
target    False
dtype: bool
```

DATASET 1

In [10]:
```
print(data.isnull().sum())
```
```
subject    0
message    0
label      0
dtype: int64
```

DATASET 2

In [15]:
```
print(data.isnull().sum())
```
```
text    0
spam    0
dtype: int64
```

DATASET 3

In [10]:
```
print(data.isnull().sum())
```
```
Category    0
Message     0
dtype: int64
```

DATASET 4

In [10]:
```
print(data.isnull().sum())
```
```
text      0
target    0
dtype: int64
```

Drop the Duplicate rows

Shape

DATASET 1

In [11]:
```
data.shape
```
Out[11]:
```
(2831, 3)
```

DATASET 2

In [16]:
```
data.shape
```
Out[16]:
```
(5726, 2)
```

DATASET 3

In [11]:
```
data.shape
```
Out[11]:
```
(5572, 2)
```

DATASET 4

In [11]:
```
data.shape
```
Out[11]:
```
(5796, 2)
```

Check if there is any Duplicate Rows

DATASET 1

In [18]:
```
duplicate = data[data.duplicated()]
print("Number of Duplicate rows: ", duplicate.shape)
data.count()
```
```
Number of Duplicate rows:  (17, 3)
```
Out[18]:
```
subject    2831
message    2831
label      2831
dtype: int64
```

DATASET 2

In [17]:
```
duplicate = data[data.duplicated()]
print("Number of Duplicate rows: ", duplicate.shape)
data.count()
```
```
Number of Duplicate rows:  (33, 2)
```
Out[17]:
```
text    5726
spam    5726
dtype: int64
```

DATASET 3

In [12]:
```
duplicate = data[data.duplicated()]
print("Number of Duplicate rows: ", duplicate.shape)
data.count()
```
```
Number of Duplicate rows:  (415, 2)
```
Out[12]:
```
Category    5572
Message     5572
dtype: int64
```

DATASET 4

In [12]:
```
duplicate = data[data.duplicated()]
print("Number of Duplicate rows: ", duplicate.shape)
data.count()
```

```
Number of Duplicate rows:  (467, 2)
```

Out[12]:
```
text      5796
target    5796
dtype: int64
```

Drop all the Duplicate Rows

DATASET 1

In [19]:
```
data = data.drop_duplicates()
data.count()
```

Out[19]:
```
subject    2814
message    2814
label      2814
dtype: int64
```

DATASET 2

In [18]:
```
data = data.drop_duplicates()
data.count()
```

Out[18]:
```
text     5693
spam     5693
dtype: int64
```

DATASET 3

In [13]:
```
data = data.drop_duplicates()
data.count()
```

Out[13]:
```
Category    5157
Message     5157
dtype: int64
```

DATASET 4

```
In [13]: data = data.drop_duplicates()
         data.count()

Out[13]: text      5329
         target    5329
         dtype: int64
```

Data Summarization

Descriptive Statistics

❖ Descriptive statistics analysis helps to describe the basic features of dataset and obtain a brief summary of the data.

❖ The describe() method in Pandas library helps us to have a brief summary of the dataset.

❖ It automatically calculates basic statistics for all numerical variables excluding NaN (we will come to this part later) values.

Display First 5 Records

DATASET 1

```
In [15]: data.head()
```

Out[15]:

	subject	message	label
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0
3	risk	a colleague and i are researching the differin...	0
4	request book information	earlier this morning i was on the phone with a...	0
5	call for abstracts : optimality in syntactic t...	content - length : 4437 call for papers is the...	0

DATASET 2

```
In [2]: data.head()
```

Out[2]:

	text	spam
0	Subject: naturally irresistible your corporate...	1
1	Subject: the stock trading gunslinger fanny i...	1
2	Subject: unbelievable new homes made easy im ...	1
3	Subject: 4 color printing special request add...	1
4	Subject: do not have money , get software cds ...	1

DATASET 3

```
In [16]: data.head()
```

Out[16]:

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

DATASET 4

```
In [14]: data.head()
```

Out[14]:

	text	target
0	From ilug-admin@linux.ie Mon Jul 29 11:28:02 2...	0
1	From gort44@excite.com Mon Jun 24 17:54:21 200...	1
2	From fork-admin@xent.com Mon Jul 29 11:39:57 2...	1
3	From dcm123@btamail.net.cn Mon Jun 24 17:49:23...	1
4	From ilug-admin@linux.ie Mon Aug 19 11:02:47 2...	0

Brief summary of Data Frame

DATASET 1

```
In [16]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2814 entries, 0 to 2892
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   subject  2814 non-null   object
 1   message  2814 non-null   object
 2   label    2814 non-null   int64
dtypes: int64(1), object(2)
memory usage: 87.9+ KB
```

DATASET 2

```
In [18]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5693 entries, 0 to 5725
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    5693 non-null   object
 1   spam    5693 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 133.4+ KB
```

DATASET 3

```
In [17]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5157 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5157 non-null   object
 1   Message   5157 non-null   object
dtypes: object(2)
memory usage: 120.9+ KB
```

DATASET 4

```
In [15]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5329 entries, 0 to 5795
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    5329 non-null   object
 1   target  5329 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 124.9+ KB
```

find the datatypes in the DataFrame

DATASET 1

```
In [17]: data.describe()
```

Out[17]:

	label
count	2814.000000
mean	0.161692
std	0.368233
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

DATASET 2

```
In [19]: data.describe()
```

Out[19]:

	spam
count	5693.000000
mean	0.240119
std	0.427193
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

DATASET 3

```
In [18]: data.describe()
```

Out[18]:

	Category	Message
count	5157	5157
unique	2	5157
top	ham	Go until jurong point, crazy.. Available only ...
freq	4516	1

DATASET 4

```
In [16]: data.describe()
```

Out[16]:

	target
count	5329.000000
mean	0.317320
std	0.465477
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

No of rows and Columns

DATASET 1

```
In [25]: Columns = data.shape[1]
         Rows = data.shape[0]
         print("Rows :", Rows)
         print("Columns :", Columns)

         Rows : 2814
         Columns : 3
```

DATASET 2

```
In [20]: Columns = data.shape[1]
         Rows = data.shape[0]
         print("Rows :", Rows)
         print("Columns :", Columns)

         Rows : 5693
         Columns : 2
```

DATASET 3

```
In [19]: Columns = data.shape[1]
         Rows = data.shape[0]
         print("Rows :", Rows)
         print("Columns :", Columns)

         Rows : 5157
         Columns : 2
```

DATASET 4

```
In [17]: Columns = data.shape[1]
         Rows = data.shape[0]
         print("Rows :", Rows)
         print("Columns :", Columns)

         Rows : 5329
         Columns : 2
```

Text Preprocessing

Adding the text Length Column for each record

DATASET 1

```
In [29]: data['Length'] = data['message'].apply(len)
         data['Length'].max()

Out[29]: 28649
```

DATASET 2

```
In [24]: data['Length'] = data['text'].apply(len)
         data['Length'].max()

Out[24]: 31055
```

DATASET 3

```
In [99]: data['Length'] = data['Message'].apply(len)
         data['Length'].max()

Out[99]: 910
```

DATASET 4

```
In [18]: data['Length'] = data['text'].apply(len)
         data['Length'].max()

Out[18]: 232305
```

Description of the data in the DataFrame

DATASET 1

```
In [30]: data.describe()
```

Out[30]:

	label	Length
count	2814.000000	2814.000000
mean	0.161692	3240.960554
std	0.368233	3685.167456
min	0.000000	17.000000
25%	0.000000	950.500000
50%	0.000000	2029.000000
75%	0.000000	4030.750000
max	1.000000	28649.000000

DATASET 2

```
In [25]: data.describe()
```

Out[25]:

	spam	Length
count	5693.000000	5693.000000
mean	0.240119	1543.172317
std	0.427193	1886.930857
min	0.000000	13.000000
25%	0.000000	508.000000
50%	0.000000	979.000000
75%	0.000000	1891.000000
max	1.000000	31055.000000

DATASET 3

```
In [100]: data.describe()
```

Out[100]:

	Length
count	5157.000000
mean	79.103936
std	58.382922
min	2.000000
25%	36.000000
50%	61.000000
75%	118.000000
max	910.000000

DATASET 4

```
In [19]: data.describe()
```

Out[19]:

	target	Length
count	5329.000000	5329.000000
mean	0.317320	4164.186527
std	0.465477	6030.253952
min	0.000000	362.000000
25%	0.000000	2390.000000
50%	0.000000	3296.000000
75%	1.000000	4492.000000
max	1.000000	232305.000000

DATASET 1

```
In [54]: data.groupby('label').describe()
```

Out[54]:

	Length							
	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
label								
0	2359.0	3130.379822	3297.716213	17.0	1023.0	2060.0	3886.0	28649.0
1	455.0	3814.279121	5222.017845	46.0	623.0	1725.0	4967.0	28571.0

DATASET 2

```
In [26]: data.groupby('spam').describe()
```

Out[26]:

	Length							
	count	mean	std	min	25%	50%	75%	max
spam								
0	4326.0	1614.353213	1741.930460	13.0	575.5	1122.0	2036.5	31055.0
1	1367.0	1317.913680	2272.067352	18.0	402.0	694.0	1252.5	28432.0

DATASET 3

```
In [123]: data.groupby('Category').describe()
```

Out[123]:

	Length							
	count	mean	std	min	25%	50%	75%	max
Category								
ham	4516.0	70.869353	56.708301	2.0	34.0	53.0	91.0	910.0
spam	641.0	137.118565	30.399707	7.0	130.0	148.0	157.0	223.0

DATASET 4

```
In [21]: data.groupby('target').describe()
```

Out[21]:

	Length							
	count	mean	std	min	25%	50%	75%	max
target								
0	3638.0	3461.003573	3173.399352	362.0	2400.25	3183.5	4054.25	92469.0
1	1691.0	5677.007096	9466.602237	736.0	2364.00	3809.0	6211.50	232305.0

Word Tokenization

DATASET 1

```
In [59]: import nltk
         nltk.download('punkt')

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\parth\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

Out[59]: True
```

```
In [85]: from nltk.tokenize import word_tokenize
         Ham_Words_Length = [len(word_tokenize(title)) for title in data[data['label']==0
         Spam_Words_Length = [len(word_tokenize(title)) for title in data[data['label']==
         print("\nHam Words Length :", max(Ham_Words_Length))
         print("\nSpam Words Length :", max(Spam_Words_Length))
         if max(Ham_Words_Length) > max(Spam_Words_Length):
             print("\nHam Text Length is Larger")
         else:
             print("\nSpam Text Length is Larger")

Ham Words Length : 6608

Spam Words Length : 6586

Ham Text Length is Larger
```

❖ For ham email, the maximum number of ham words used in an email is 6608.

❖ For spam email, the maximum number of spam words used in an email is 6586.

DATASET 2

```
In [30]: import nltk
         nltk.download('punkt')

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\parth\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!

Out[30]: True
```

```
In [32]: from nltk.tokenize import word_tokenize
         Ham_Words_Length = [len(word_tokenize(title)) for title in data[data['spam']==0]
         Spam_Words_Length = [len(word_tokenize(title)) for title in data[data['spam']==1
         print("\nHam Words Length :", max(Ham_Words_Length))
         print("\nSpam Words Length :", max(Spam_Words_Length))
         if max(Ham_Words_Length) > max(Spam_Words_Length):
             print("\nHam Text Length is Larger")
         else:
             print("\nSpam Text Length is Larger")

Ham Words Length : 6350

Spam Words Length : 6131

Ham Text Length is Larger
```

❖ For ham email, the maximum number of ham words used in an email is 6350.

❖ For spam email, the maximum number of spam words used in an email is 6131.

❖ It's evident that the spam emails have less words as compared to ham emails.

DATASET 3

In [124]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\parth\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Out[124]: True

In [130]:
```
from nltk.tokenize import word_tokenize
Ham_Words_Length = [len(word_tokenize(title)) for title in data[data['Message']=
Spam_Words_Length = [len(word_tokenize(title)) for title in data[data['Message']
print("\nHam Words Length :", max(Ham_Words_Length))
print("\nSpam Words Length :", max(Spam_Words_Length))
if max(Ham_Words_Length) > max(Spam_Words_Length):
    print("\nHam Text Length is Larger")
else:
    print("\nSpam Text Length is Larger")
```

DATASET 4

In [22]:
```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Error loading punkt: <urlopen error [WinError 10060] A
[nltk_data]     connection attempt failed because the connected party
[nltk_data]     did not properly respond after a period of time, or
[nltk_data]     established connection failed because connected host
[nltk_data]     has failed to respond>
```

Out[22]: False

In [24]:
```
from nltk.tokenize import word_tokenize
Ham_Words_Length = [len(word_tokenize(title)) for title in data[data['target']==0].text.values]
Spam_Words_Length = [len(word_tokenize(title)) for title in data[data['target']==1].text.values]
print("\nHam Words Length :", max(Ham_Words_Length))
print("\nSpam Words Length :", max(Spam_Words_Length))
if max(Ham_Words_Length) > max(Spam_Words_Length):
    print("\nHam Text Length is Larger")
else:
    print("\nSpam Text Length is Larger")
```

```
Ham Words Length : 17677

Spam Words Length : 18622

Spam Text Length is Larger
```

Removing Punctuations and Stop Words

DATASET 1

```
import string
class Data_Clean():
    def __init__(self):
        pass
    def Message_Cleaning(self, message):
        Text = [char for char in message if char not in string.punctuation]
        Text = ''.join(Text)
        Text_Filtered = [word for word in Text.split() if word.lower() not in st
        Text_Filtered = ' '.join(Text_Filtered)
        return Text_Filtered
    def Clean(self, U_data):
        C_Data = U_data.apply(self.Message_Cleaning)
        return C_Data
Cleaned_Data = Data_Clean()
data['Cleaned Text'] = Cleaned_Data.Clean(data['text'])
data.head()
```

DATASET 2

```
import string
class Data_Clean():
    def __init__(self):
        pass
    def Message_Cleaning(self, message):
        Text = [char for char in message if char not in string.punctuation]
        Text = ''.join(Text)
        Text_Filtered = [word for word in Text.split() if word.lower() not in st
        Text_Filtered = ' '.join(Text_Filtered)
        return Text_Filtered
    def Clean(self, U_data):
        C_Data = U_data.apply(self.Message_Cleaning)
        return C_Data
Cleaned_Data = Data_Clean()
data['Cleaned Text'] = Cleaned_Data.Clean(data['text'])
data.head()
```

DATASET 3

```
import string
class Data_Clean():
    def __init__(self):
        pass
    def Message_Cleaning(self, message):
        Text = [char for char in message if char not in string.punctuation]
        Text = ''.join(Text)
        Text_Filtered = [word for word in Text.split() if word.lower() not in st
        Text_Filtered = ' '.join(Text_Filtered)
        return Text_Filtered
    def Clean(self, U_data):
        C_Data = U_data.apply(self.Message_Cleaning)
        return C_Data
Cleaned_Data = Data_Clean()
data['Cleaned Text'] = Cleaned_Data.Clean(data['text'])
data.head()
```

DATASET 4

```
import string
class Data_Clean():
    def __init__(self):
        pass
    def Message_Cleaning(self, message):
        Text = [char for char in message if char not in string.punctuation]
        Text = ''.join(Text)
        Text_Filtered = [word for word in Text.split() if word.lower() not in stopwords.words('english'
        Text_Filtered = ' '.join(Text_Filtered)
        return Text_Filtered
    def Clean(self, U_data):
        C_Data = U_data.apply(self.Message_Cleaning)
        return C_Data
Cleaned_Data = Data_Clean()
data['Cleaned Text'] = Cleaned_Data.Clean(data['text'])
data.head()
```

Stemming

DATASET 1

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)
data['Cleaned Text'] = data['Cleaned Text'].apply(transform_text)
data.head()
```

DATASET 2

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)
data['Cleaned Text'] = data['Cleaned Text'].apply(transform_text)
data.head()
```

DATASET 3

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)
data['Cleaned Text'] = data['Cleaned Text'].apply(transform_text)
data.head()
```

DATASET 4

```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
def transform_text(text):
    text = text.lower()
    text = nltk.word_tokenize(text)

    y = []
    for i in text:
        if i.isalnum():
            y.append(i)
    text = y[:]
    y.clear()
    for i in text:
        if i not in stopwords.words('english') and i not in string.punctuation:
            y.append(i)
    text = y[:]
    y.clear()

    for i in text:
        y.append(ps.stem(i))

    return " ".join(y)
data['Cleaned Text'] = data['Cleaned Text'].apply(transform_text)
data.head()
```

Data Visualization & interpretation

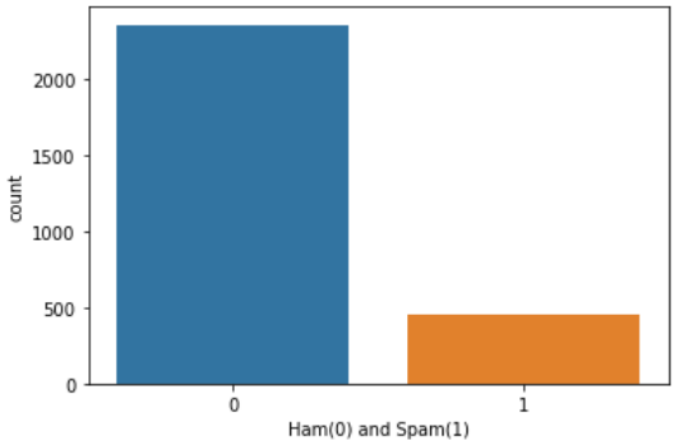
Countplot the Spam & Ham ratio

The countplot is used to represent the occurrence(counts) of the observation present in the categorical variable. It uses the concept of a bar chart for the visual depiction.

DATASET 1

In [78]:
```
import seaborn as sn
ham = data.loc[data['label']==0]
spam = data.loc[data['label']==1]
spam['Length'].plot(bins=60, kind='hist')
data['Ham(0) and Spam(1)'] = data['label']
sn.countplot(data['Ham(0) and Spam(1)'], label = "Count")
```

Out[78]: <AxesSubplot:xlabel='Ham(0) and Spam(1)', ylabel='count'>

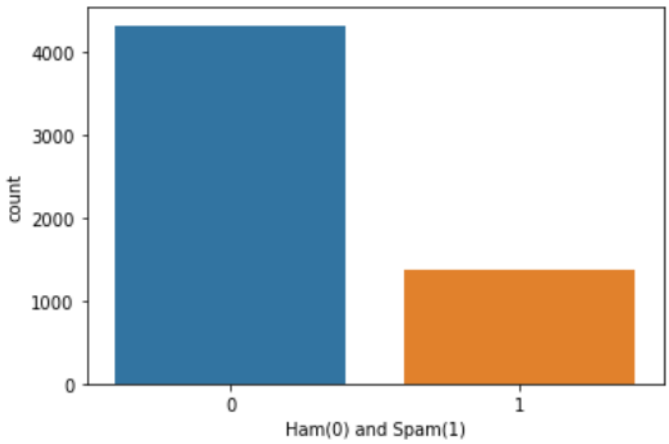


Here, there are around 2500 ham emails and 500 spam emails.
The ham and spam ratio is 5:1 that means in count of 6 emails there will be atleast 5 ham emails and atleast 1 spam email.

DATASET 2

In [121]:
```
import seaborn as sn
ham = data.loc[data['spam']==0]
spam = data.loc[data['spam']==1]
spam['Length'].plot(bins=60, kind='hist')
data['Ham(0) and Spam(1)'] = data['spam']
sn.countplot(data['Ham(0) and Spam(1)'], label = "Count")
```

Out[121]: <AxesSubplot:xlabel='Ham(0) and Spam(1)', ylabel='count'>



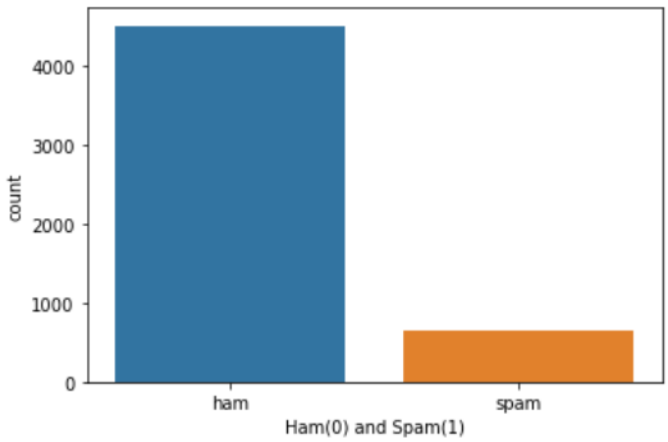
Here, there are around 4200 ham emails and 1200 spam emails.

The ham and spam ratio is 7:3 that means in count of 10 emails there will be atleast 7 ham emails and atleast 3 spam email.

DATASET 3

In [138]:
```
import seaborn as sn
ham = data.loc[data['Category']=='ham']
spam = data.loc[data['Category']=='spam']
spam['Length'].plot(bins=60, kind='hist')
data['Ham(0) and Spam(1)'] = data['Category']
sn.countplot(data['Ham(0) and Spam(1)'], label = "Count")
```

Out[138]: <AxesSubplot:xlabel='Ham(0) and Spam(1)', ylabel='count'>

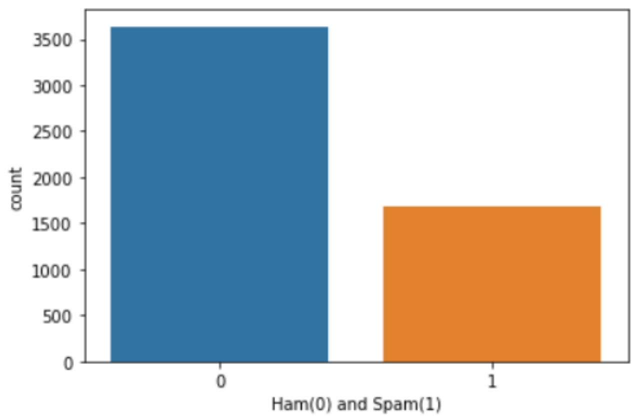


Here, there are around 4200 ham emails and 500 spam emails.
The ham and spam ratio is 5:1 that means in count of 6 emails there will be atleast 5 ham emails and atleast 1 spam email.

DATASET 4

In [27]:
```
import seaborn as sn
ham = data.loc[data['target']==0]
spam = data.loc[data['target']==1]
spam['Length'].plot(bins=60, kind='hist')
data['Ham(0) and Spam(1)'] = data['target']
sn.countplot(data['Ham(0) and Spam(1)'], label = "Count")
```

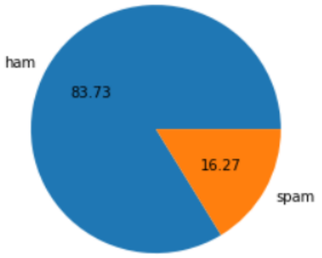
Out[27]: <AxesSubplot:xlabel='Ham(0) and Spam(1)', ylabel='count'>



Pie Plot the Spam & Ham Ratio

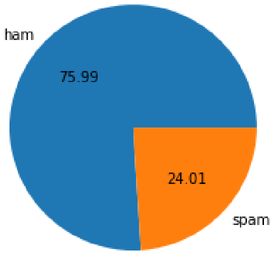
DATASET 1

```
In [8]: import matplotlib.pyplot as plt
        plt.pie(data['label'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
        plt.show()
```



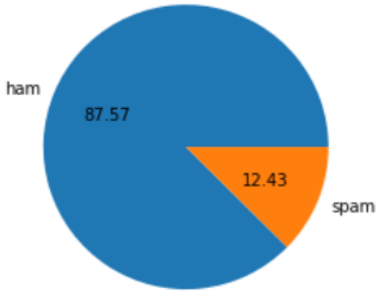
DATASET 2

```
In [28]: import matplotlib.pyplot as plt
         plt.pie(data['spam'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
         plt.show()
```



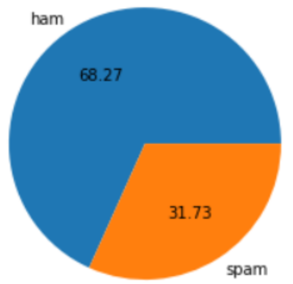
DATASET 3

```
In [139]: import matplotlib.pyplot as plt
          plt.pie(data['Category'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
          plt.show()
```



DATASET 4

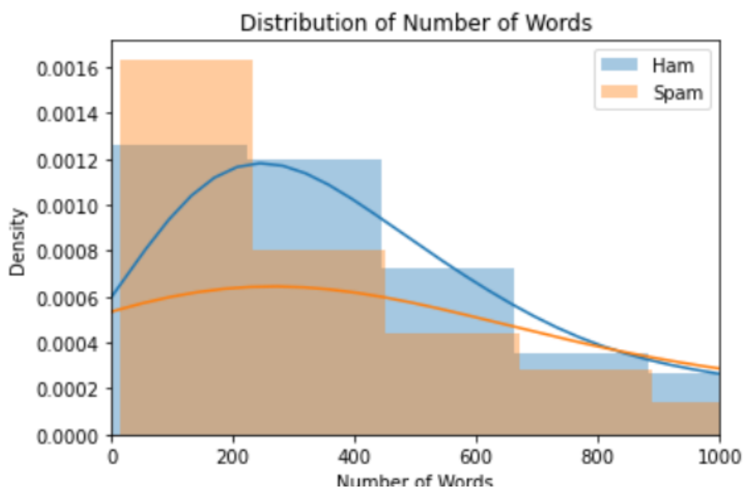
```
In [28]: import matplotlib.pyplot as plt
         plt.pie(data['target'].value_counts(), labels=['ham','spam'],autopct="%0.2f")
         plt.show()
```



Distplot the Spam & Ham record's length after tokenizing

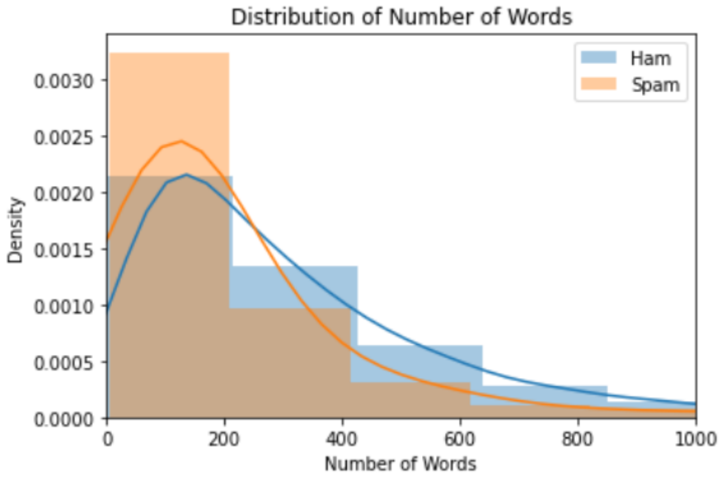
DATASET 1

```
In [88]: ax = sn.distplot(Ham_Words_Length, norm_hist = True, bins = 30, label = 'Ham')
         ax = sn.distplot(Spam_Words_Length, norm_hist = True, bins = 30, label = 'Spam')
         print()
         plt.title('Distribution of Number of Words')
         plt.xlabel('Number of Words')
         plt.legend()
         plt.xlim(0, 1000);

         plt.show()
```



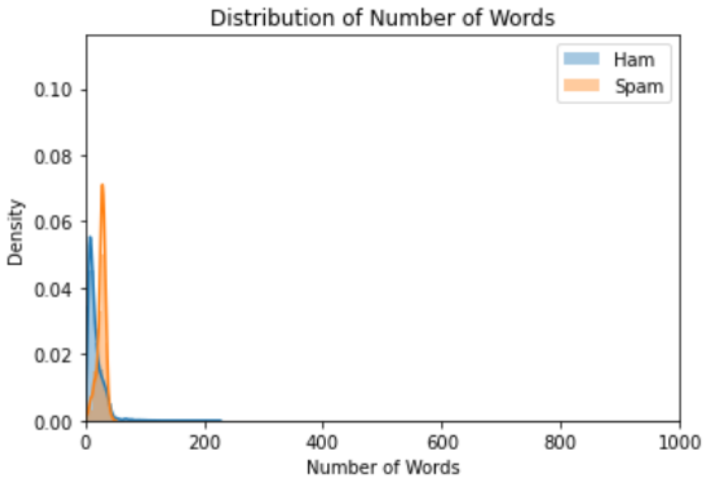
DATASET 2

```
In [128]: ax = sn.distplot(Ham_Words_Length, norm_hist = True, bins = 30, label = 'Ham')
          ax = sn.distplot(Spam_Words_Length, norm_hist = True, bins = 30, label = 'Spam')
          print()
          plt.title('Distribution of Number of Words')
          plt.xlabel('Number of Words')
          plt.legend()
          plt.xlim(0, 1000);

          plt.show()
```



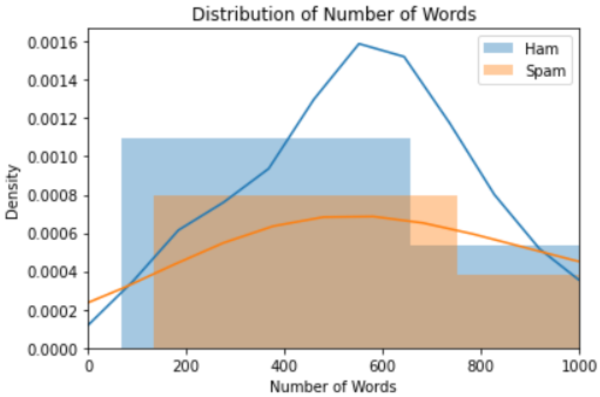
DATASET 3

In [141]:
```
ax = sn.distplot(Ham_Words_Length, norm_hist = True, bins = 30, label = 'Ham')
ax = sn.distplot(Spam_Words_Length, norm_hist = True, bins = 30, label = 'Spam')
print()
plt.title('Distribution of Number of Words')
plt.xlabel('Number of Words')
plt.legend()
plt.xlim(0, 1000);

plt.show()
```



DATASET 4

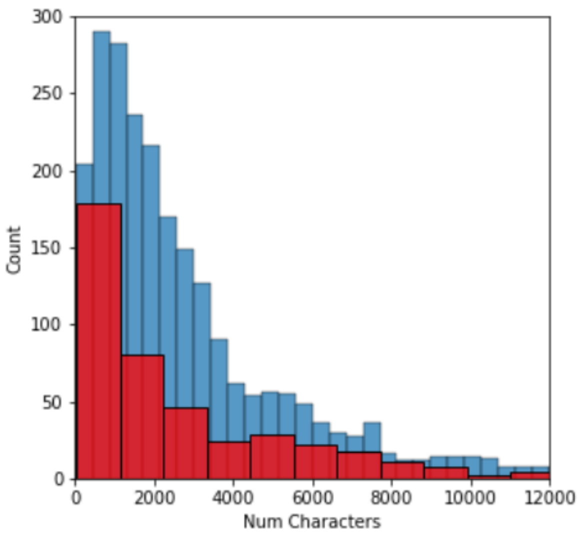
```
In [29]: ax = sn.distplot(Ham_Words_Length, norm_hist = True, bins = 30, label = 'Ham')
         ax = sn.distplot(Spam_Words_Length, norm_hist = True, bins = 30, label = 'Spam')
         print()
         plt.title('Distribution of Number of Words')
         plt.xlabel('Number of Words')
         plt.legend()
         plt.xlim(0, 1000);

         plt.show()
```



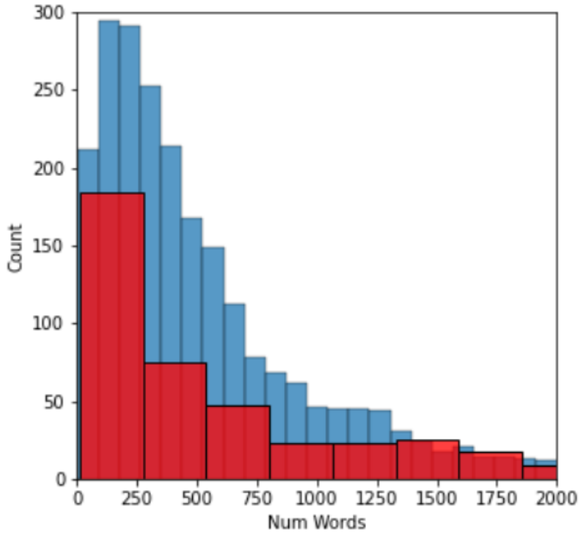
Histplot the number of characters, words and sentences

DATASET 1

```
In [89]: data['Num Characters'] = data['message'].apply(len)
         data['Num Words'] = data['message'].apply(lambda x:len(nltk.word_tokenize(x)))
         data['Num Sentences'] = data['message'].apply(lambda x:len(nltk.sent_tokenize(x)
         )
         import seaborn as sns
         plt.figure(figsize=(5,5))
         sns.histplot(data[data['label'] == 0]['Num Characters'])
         sns.histplot(data[data['label'] == 1]['Num Characters'], color='red')
         plt.xlim(0, 12000);
         plt.ylim(0, 300);
```

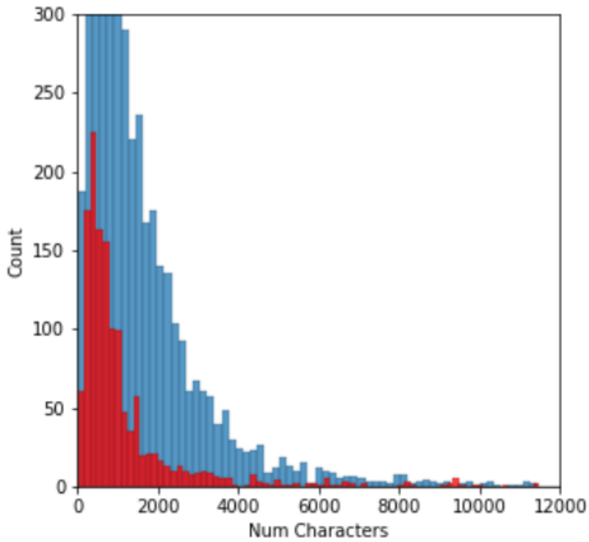


In [90]:
```
import seaborn as sns
plt.figure(figsize=(5,5))
sns.histplot(data[data['label'] == 0]['Num Words'])
sns.histplot(data[data['label'] == 1]['Num Words'], color='red')
plt.xlim(0, 2000);
plt.ylim(0, 300);
```

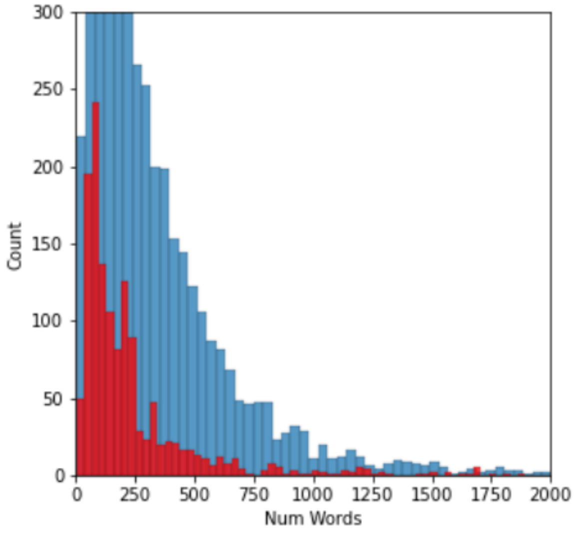


DATASET 2

In [129]:
```
data['Num Characters'] = data['text'].apply(len)
data['Num Words'] = data['text'].apply(lambda x:len(nltk.word_tokenize(x)))
data['Num Sentences'] = data['text'].apply(lambda x:len(nltk.sent_tokenize(x))
)
import seaborn as sns
plt.figure(figsize=(5,5))
sns.histplot(data[data['spam'] == 0]['Num Characters'])
sns.histplot(data[data['spam'] == 1]['Num Characters'], color='red')
plt.xlim(0, 12000);
plt.ylim(0, 300);
```

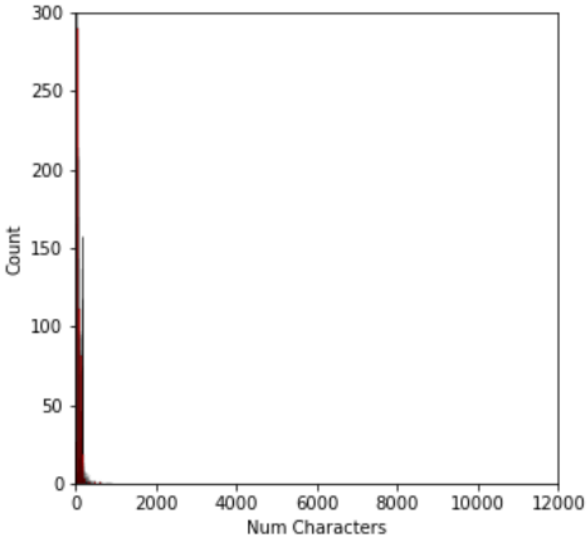


In [130]:
```
import seaborn as sns
plt.figure(figsize=(5,5))
sns.histplot(data[data['spam'] == 0]['Num Words'])
sns.histplot(data[data['spam'] == 1]['Num Words'], color='red')
plt.xlim(0, 2000);
plt.ylim(0, 300);
```

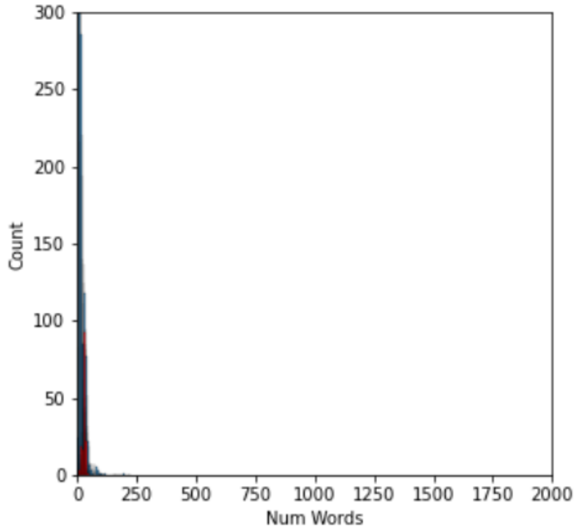


DATASET 3

In [168]:
```
data['Num Characters'] = data['Message'].apply(len)
data['Num Words'] = data['Message'].apply(lambda x:len(nltk.word_tokenize(x)))
data['Num Sentences'] = data['Message'].apply(lambda x:len(nltk.sent_tokenize(x)
)
import seaborn as sns
plt.figure(figsize=(5,5))
sns.histplot(data[data['Category'] == 'spam']['Num Characters'])
sns.histplot(data[data['Category'] == 'ham']['Num Characters'], color='red')
plt.xlim(0, 12000);
plt.ylim(0, 300);
```

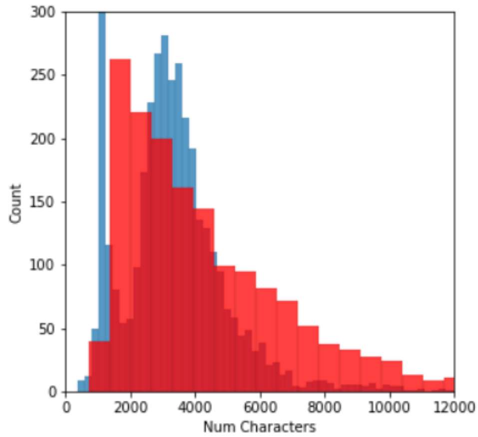


In [169]:
```
import seaborn as sns
plt.figure(figsize=(5,5))
sns.histplot(data[data['Category'] == 'ham']['Num Words'])
sns.histplot(data[data['Category'] == 'spam']['Num Words'], color='red')
plt.xlim(0, 2000);
plt.ylim(0, 300);
```

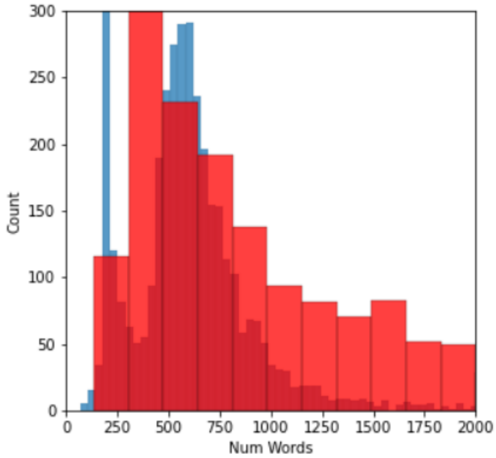


DATASET 4

In [30]:
```
data['Num Characters'] = data['text'].apply(len)
data['Num Words'] = data['text'].apply(lambda x:len(nltk.word_tokenize(x)))
data['Num Sentences'] = data['text'].apply(lambda x:len(nltk.sent_tokenize(x))
)
import seaborn as sns
plt.figure(figsize=(5,5))
sns.histplot(data[data['target'] == 0]['Num Characters'])
sns.histplot(data[data['target'] == 1]['Num Characters'], color='red')
plt.xlim(0, 12000);
plt.ylim(0, 300);
```



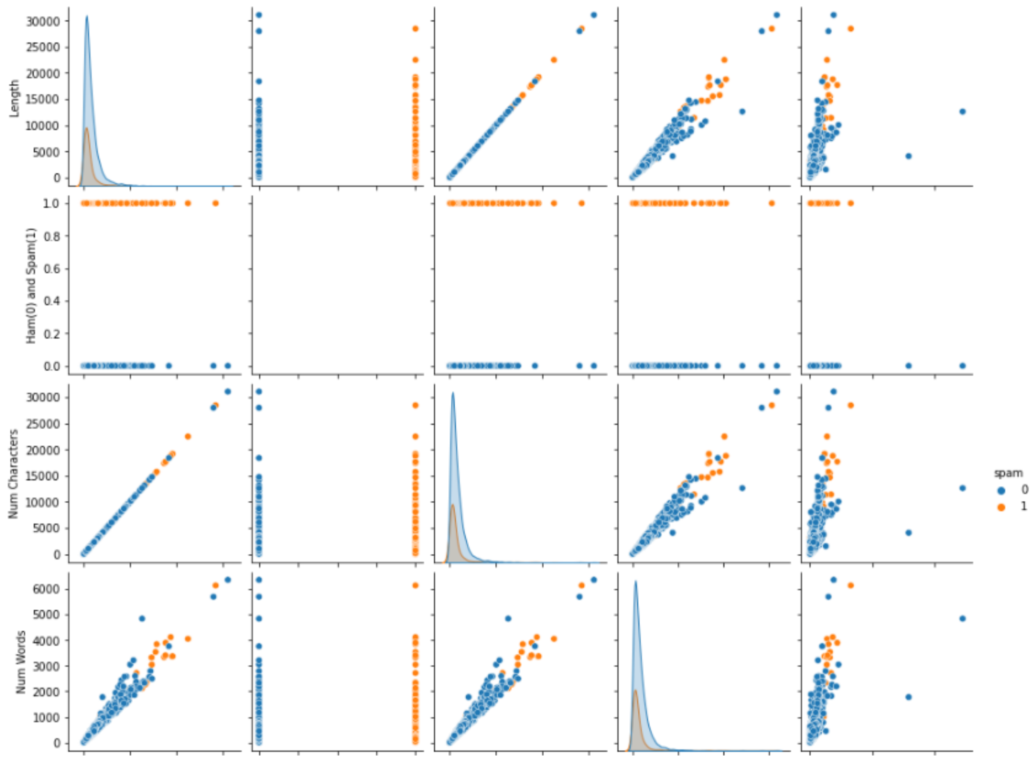
```
In [31]: import seaborn as sns
         plt.figure(figsize=(5,5))
         sns.histplot(data[data['target'] == 0]['Num Words'])
         sns.histplot(data[data['target'] == 1]['Num Words'], color='red')
         plt.xlim(0, 2000);
         plt.ylim(0, 300);
```

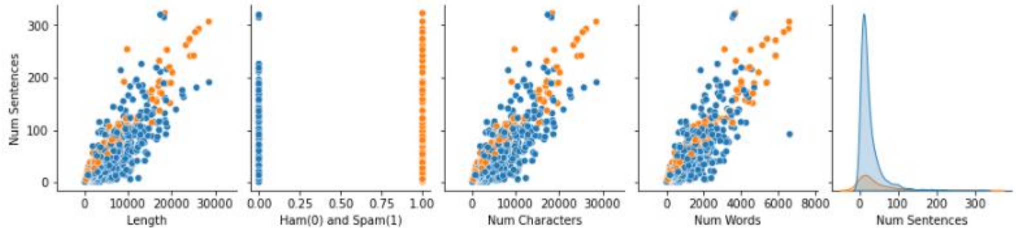


Scatter plot

DATASET 1

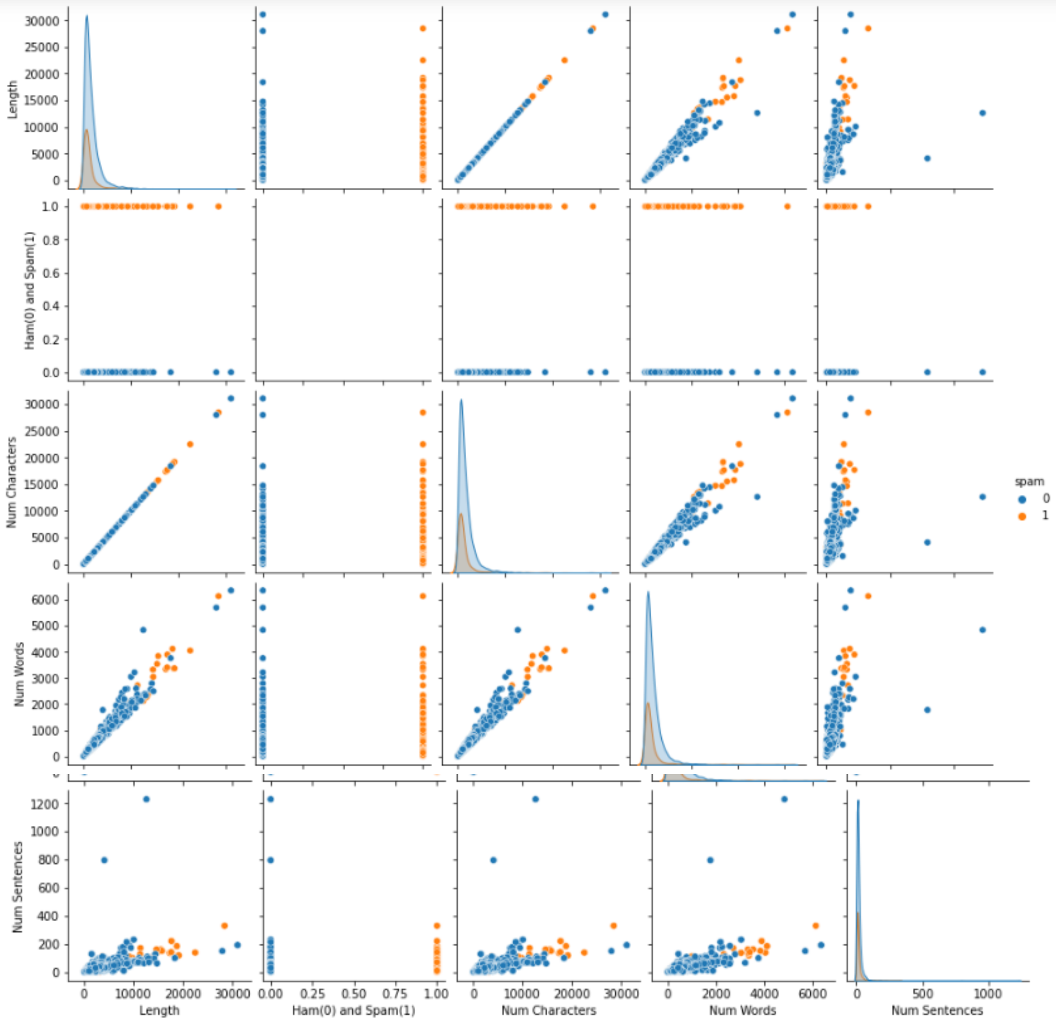
```
In [92]: import seaborn as sn
         sn.pairplot(data, hue='label')
```





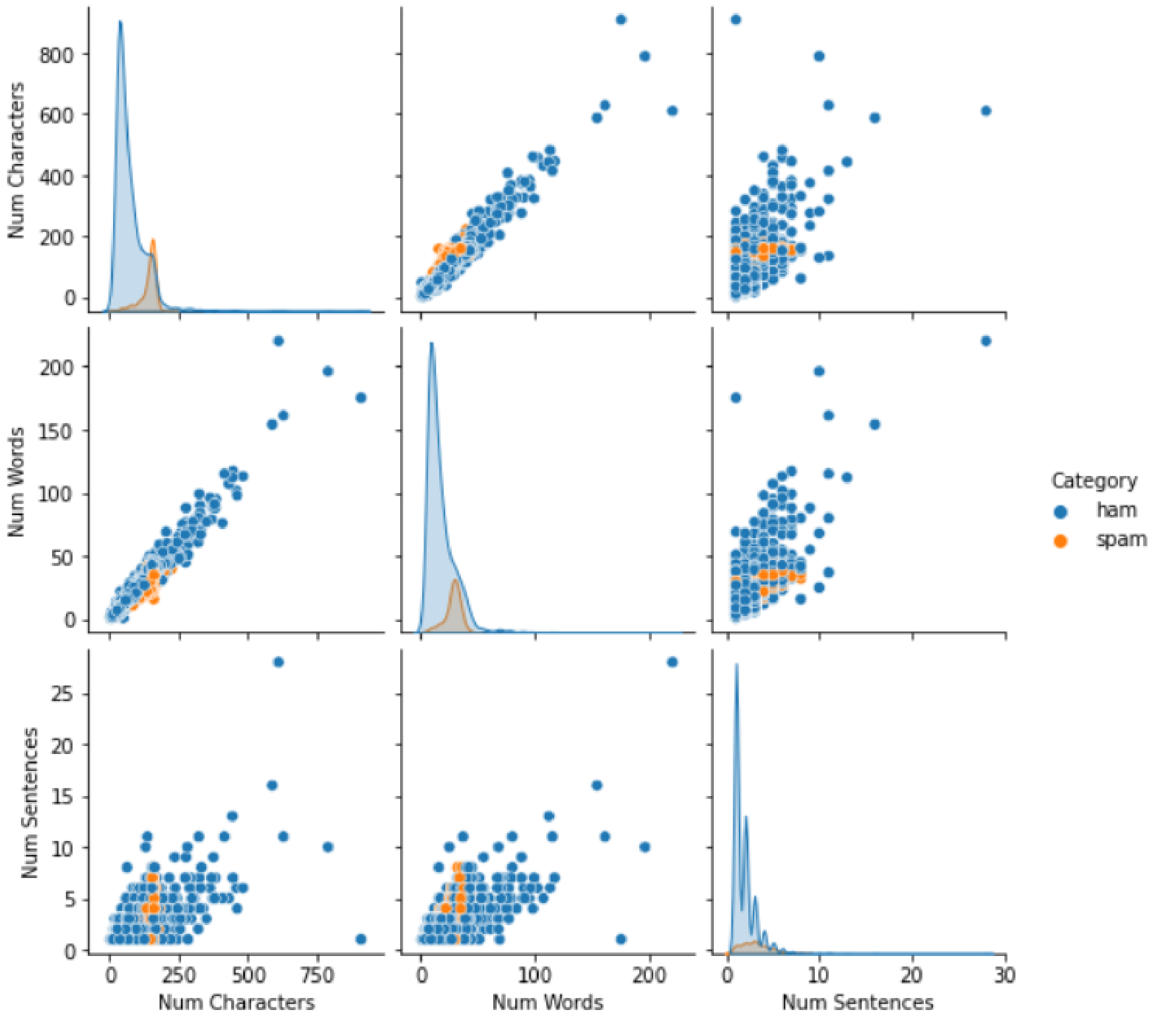
DATASET 2

In [131]:
```
import seaborn as sn
sn.pairplot(data, hue='spam')
```



DATASET 3

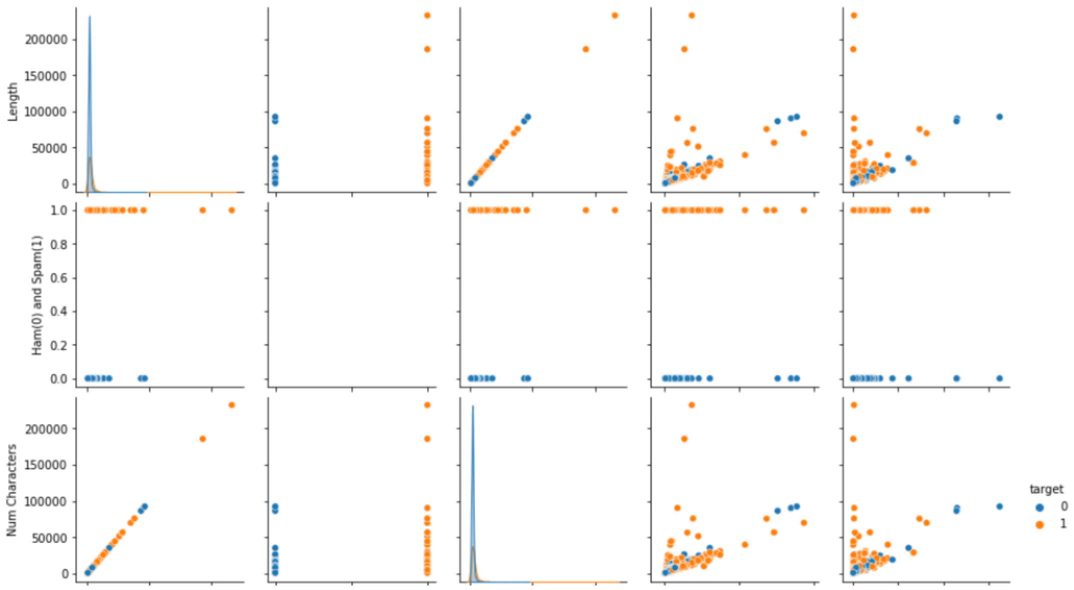
In [171]:
```
import seaborn as sn
sn.pairplot(data, hue='Category')
```

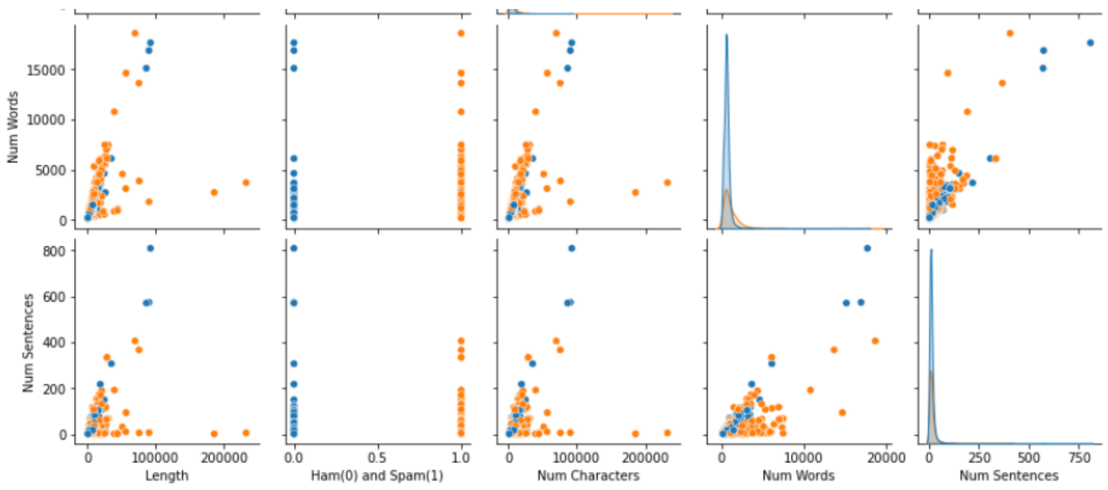


DATASET 4

```
2]: import seaborn as sn
    sn.pairplot(data, hue='target')
```

```
2]: <seaborn.axisgrid.PairGrid at 0x2384a8957f0>
```





Python Packages Used

NumPy

- Python has a strong set of data types and data structures.
- Numpy is a data handling library, particularly one which allows us to handle large multi-dimensional arrays along with a huge collection of mathematical operations.
- It is also known for its speed of execution and vectorization capabilities.
- It provides MATLAB style functionality and it is also a core dependency for other majorly used libraries like pandas, matplotlib and so on.
- Matrix (and multi-dimensional array) manipulation capabilities like transpose, reshape, etc are possible.
- One of its disadvantage is its high performance comes at a cost. The data types are native to hardware and not python, thus incurring an overhead when numpy objects have to be transformed back to python equivalent ones and vice-versa

Pandas

- Pandas is a python library that provides flexible and expressive data structures (like dataframes and series) for data manipulation.
- It is built on top of numpy, pandas is as fast and yet easier to use. Pandas provides capabilities to read and write data from different sources like CSVs, Excel, SQL Databases, HDFS and many more.
- It provides functionality to add, update and delete columns, combine or split dataframes/series, handle datetime objects, impute null/missing values, handle time series data, conversion to and from numpy objects and so on.
- Some of its advantages are it is extremely easy to use and with a small learning curve to handle tabular data,Compatible with underlying NumPy objects and go to choice for most Machine Learning libraries like scikit-learn and capability to prepare plots/visualizations out of the box (utilizes matplotlib to prepare different visualization under the hood).

Collections

- The Python collection module is defined as a container that is used to store collections of data such as list, dictionary, set, and tuple, etc.
- It was introduced to improve the functionalities of the built-in collection containers.
- Python collection module was first introduced in its 2.4 release.

Matplotlib

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. It is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.
- Matplotlib is a high customizable low-level library that provides a whole lot of controls and knobs to prepare any type of visualization/figure.
- One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

SCIKIT-LEARN

- Scikit-learn provides a simple yet powerful fit-transform and predict paradigm to learn from data, transform the data and finally predict.
- Using this interface, it provides capabilities to prepare classification, regression, clustering and ensemble models.
- It also provides a multitude of utilities for pre-processing, metrics, model evaluation techniques, etc.

Math

- This is the most basic math module that is available in Python. It covers basic mathematical operations like sum, exponential, modulus, etc
- This library is not useful when dealing with complex mathematical operations like multiplication of matrices.
- The calculations performed with the functions of the python math library are also much slower.

Natural Language Toolkit (NLTK)

- The list includes low-level tasks such as tokenization (it provides different tokenizers), n-gram analysers, collocation parsers, POS taggers, NER and many more.
- NLTK is primarily for English based NLP tasks.
- Provides a huge array of algorithms and utilities to handle NLP tasks, right from low-level parsing utilities to high-level algorithms like CRFs.

Seaborn

- Built on top of matplotlib, seaborn is a high-level visualization library
- It provides sophisticated styles straight out of the box (which would take some good amount of effort if done using matplotlib).
- It provides capabilities to perform regression analysis, handling of categorical variables and aggregate statistics.
- seaborn provides a range of visualizations and capabilities to work with multivariate analysis.

Word Cloud

- A tag cloud (word cloud or wordle or weighted list in visual design) is a novelty visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text.
- Tags are usually single words, and the importance of each tag is shown with font size or color.
- A word cloud lets us easily identify the keywords in a text where the size of the words represents their frequency.

XGBoost

- XGBoost is an open-source software library that implements optimized distributed gradient boosting machine learning algorithms under the Gradient Boosting framework.
- One of the most widely used libraries/algorithms used in various data science competitions and real-world use cases, XGBoost is probably one of the best-known variants.
- A highly optimized and distributed implementation, XGBoost enables parallel execution and thus provides immense performance improvement over gradient boosted trees

CLASSIFICATION ALGORITHMS

KNN

- KNN is a supervised machine learning algorithm whose goal is to learn a function such that f(X) = Y where X is the input, and Y is the output.
- KNN can be used both for classification as well as regression.
- It is non parametric as it does not make an assumption about the underlying data distribution pattern.
- Lazy algorithm as KNN does not have a training step. All data points will be used only at the time of prediction, and thus the prediction step is costly.
- KNN uses feature similarity to predict the cluster that the new point will fall into.

Working of KNN

- In the training phase, the model will store the data points.
- In the testing phase, the distance from the query point to the points from the training phase is calculated to classify each point in the test dataset.
- Various distances can be calculated, but the most popular one is the Euclidean distance (for smaller dimension data)

Limitation of KNN

- Time complexity and space complexity is enormous, which is a major disadvantage of KNN.
- If the data point is far away from the classes present (no similarity), KNN will classify the point even if it is an outlier.

IMPLEMENTATION OF KNN

KNN Using Sklearn

```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(n_neighbors = 1)

classifier.fit(X_train, Y_train)

Y_pred = classifier.predict(X_test)
```

Prediction on test set

```
pred_prob1 = classifier.predict_proba(X_test)
```

Accuracy

```
import sklearn.metrics as metrics

from sklearn.metrics import precision_score, \
    recall_score, confusion_matrix, classification_report, \
    accuracy_score, f1_score

print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))

print("Precision:",metrics.precision_score(Y_test, Y_pred))

print("Recall:",metrics.recall_score(Y_test, Y_pred))

print("F1 Score :",f1_score(Y_test,Y_pred))
```

DATASET 1

```
Accuracy: 0.8869179600886918
Precision: 0.9375
Recall: 0.379746835443038
F1 Score : 0.5405405405405406
```

DATASET 2

```
Accuracy: 0.9813391877058177
Precision: 0.9764150943396226
Recall: 0.9452054794520548
F1 Score : 0.9605568445475638
```

DATASET 3

```
Accuracy: 0.9333333333333333
Precision: 0.8554216867469879
Recall: 0.6228070175438597
F1 Score : 0.7208121827411168
```

DATASET 4

```
Accuracy: 0.9671746776084408
Precision: 0.9644128113879004
Recall: 0.9377162629757786
F1 Score : 0.9508771929824562
```

CONFUSION MATRIX

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(Y_test, Y_pred)

print(cm)
```

DATASET 1

```
[[370   2]
 [ 49  30]]
```

DATASET 2

```
[[845   5]
 [  7 282]]
```

DATASET 3

```
[[699  12]
 [ 43  71]]
```

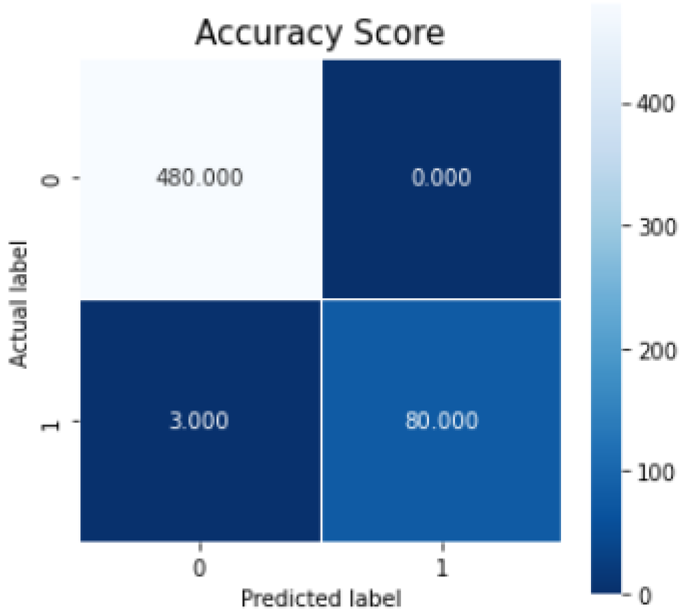
DATASET 4

```
[[715   1]
 [  3 347]]
```

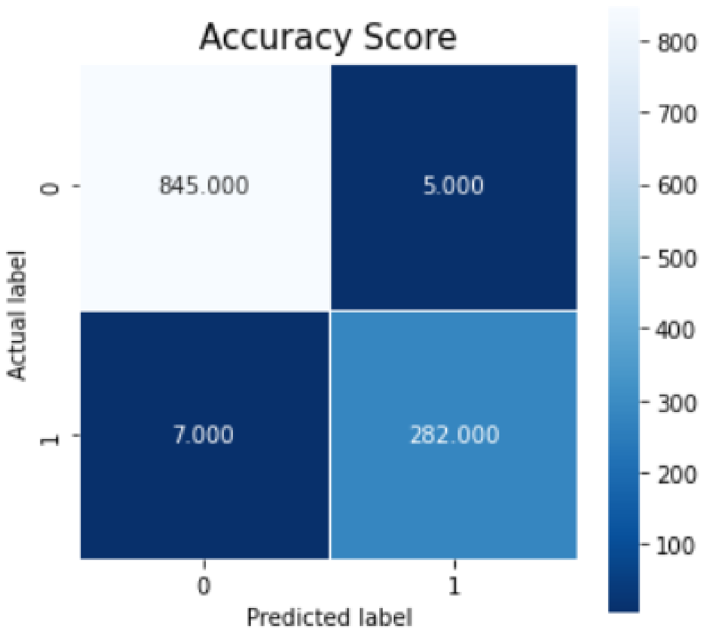
HEAT MAP

```
plt.figure(figsize=(5,5))

sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');

plt.ylabel('Actual label');

plt.xlabel('Predicted label');

all_sample_title = 'Accuracy Score'

plt.title(all_sample_title, size = 15);
```

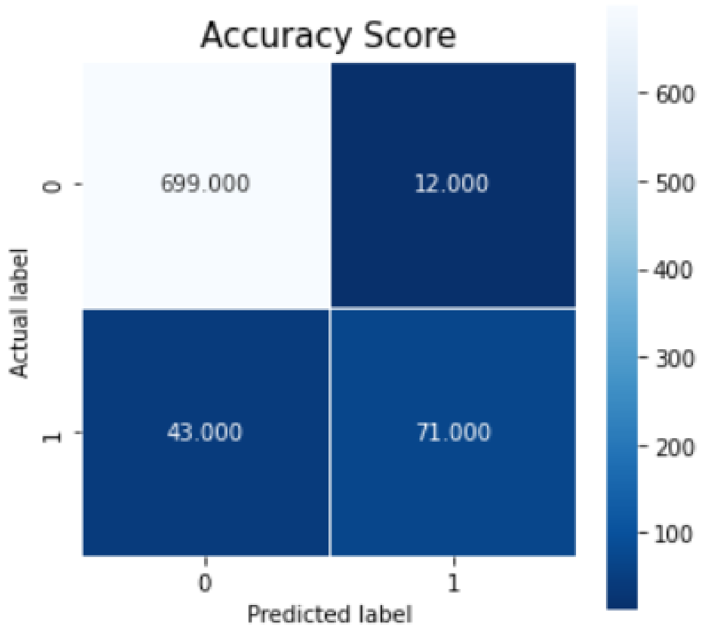
DATASET 1



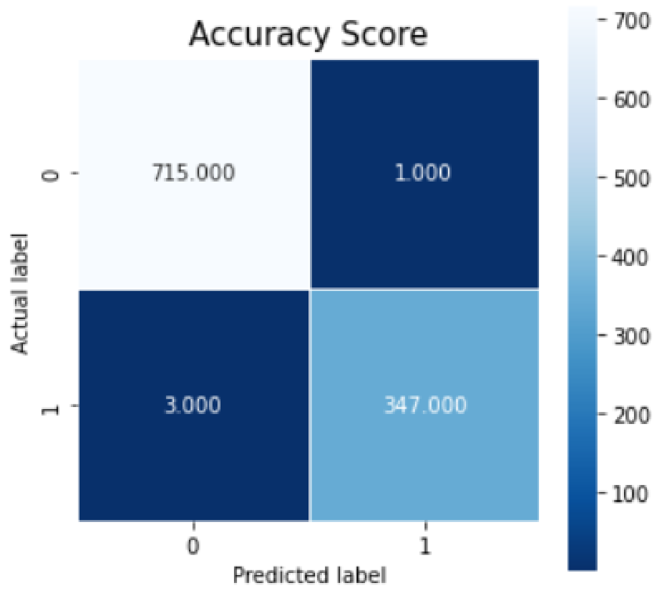
DATASET 2



DATASET 3



DATASET 4



Validation Of KNN

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = Y_train, cv = 10)
print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
accuracies
```

DATASET 1

```
Accuracy: 83.67 %

array([0.37222222, 0.87777778, 0.87777778, 0.9       , 0.89444444,
       0.87777778, 0.9       , 0.89444444, 0.90555556, 0.86666667])
```

DATASET 2

```
Accuracy: 97.89 %

array([0.97260274, 0.98356164, 0.97260274, 0.97802198, 0.98076923,
       0.99175824, 0.97252747, 0.96978022, 0.98901099, 0.97802198])
```

DATASET 3

```
Accuracy: 88.30 %

array([0.87878788, 0.88484848, 0.88787879, 0.88787879, 0.87878788,
       0.88181818, 0.88181818, 0.88787879, 0.88181818, 0.87878788])
```

DATASET 4

```
Accuracy: 97.10 %

array([0.96480938, 0.95601173, 0.97067449, 0.96774194, 0.97947214,
       0.97947214, 0.97360704, 0.96774194, 0.95894428, 0.99120235])
```

Simple K-fold Cross Validation

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(data['Cleaned Text'].values.astype('U')).toarray()
Y = data['target'].values
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
accuracy1 = []
kf = KFold(n_splits=5, random_state=None)
for train_index, test_index in kf.split(X):

    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]

    # Standardization
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    # Training the Model
    model = KNeighborsClassifier( n_neighbors = 5 )
    model.fit( X_train, Y_train.ravel() )
```

```
# Predicting Test Data Set
Y_pred = model.predict( X_test )

# Confusion Matrix
print("\n\nConfusion Matrix\n\n", confusion_matrix(Y_test,Y_pred), end = "\n")

# F1 Score
print("\nF1 Score : ", f1_score(Y_test,Y_pred), end = "\n")

# Accuracy Score
accuracy1.append(accuracy_score(Y_test, Y_pred))
print("\nAccuracy Score : ", accuracy_score(Y_test,Y_pred))
```

DATASET 1

```
Confusion Matrix

 [[ 56 413]
 [  4  90]]

F1 Score :  0.30150753768844224

Accuracy Score :  0.25932504440497334
```

```
Confusion Matrix

 [[ 86 385]
 [  1  91]]

F1 Score :  0.3204225352112676

Accuracy Score :  0.31438721136767317
```

```
Confusion Matrix

 [[ 99 370]
 [  0  94]]

F1 Score :  0.33691756272401435

Accuracy Score :  0.3428063943161634
```

```
Confusion Matrix

 [[ 70 407]
 [  1  85]]

F1 Score :  0.2941176470588235

Accuracy Score :  0.2753108348134991
```

```
Confusion Matrix

 [[374  99]
 [  8  81]]

F1 Score :  0.6022304832713755

Accuracy Score :  0.8096085409252669
```

DATASET 2

```
Confusion Matrix

 [[  1   0]
 [861 277]]

F1 Score :  0.3915194346289752

Accuracy Score :  0.2440737489025461
```

```
Confusion Matrix

 [[909   1]
 [104 125]]

F1 Score :  0.7042253521126761

Accuracy Score :  0.9078138718173837
```

```
Confusion Matrix

 [[1134    5]
 [   0    0]]

F1 Score :  0.0

Accuracy Score :  0.9956101843722563
```

DATASET 3

```
Confusion Matrix

 [[878   2]
 [118  34]]

F1 Score :  0.3617021276595745
```

DATASET 4

```
Confusion Matrix

 [[711   1]
 [125 229]]

F1 Score :  0.7842465753424658

Accuracy Score :  0.8818011257035647


Confusion Matrix

 [[750   0]
 [112 204]]

F1 Score :  0.7846153846153847

Accuracy Score :  0.8949343339587242


Confusion Matrix

 [[703   1]
 [115 247]]

F1 Score :  0.8098360655737704

Accuracy Score :  0.8911819887429644
```

Stratified K-fold Cross Validation

```
from sklearn.model_selection import StratifiedKFold
accuracy2 = []
skf = StratifiedKFold(n_splits=5, random_state=None)
for train_index, test_index in kf.split(X):

    #print("Train:", train_index, "\nValidation:",test_index)

    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = Y[train_index], Y[test_index]

    # Standardization
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

```
# Training the Model
model = KNeighborsClassifier( n_neighbors = 5 )
model.fit( X_train, Y_train.ravel() )

# Predicting Test Data Set
Y_pred = model.predict( X_test )

# Confusion Matrix
print("\n\nConfusion Matrix\n\n", confusion_matrix(Y_test,Y_pred), end = "\n")

# F1 Score
print("\nF1 Score : ", f1_score(Y_test,Y_pred), end = "\n")

# Accuracy Score
accuracy2.append(accuracy_score(Y_test, Y_pred))
print("\nAccuracy Score : ", accuracy_score(Y_test,Y_pred))
```

DATASET 1

```
Confusion Matrix

 [[ 56 413]
 [  4  90]]

F1 Score :  0.30150753768844224

Accuracy Score :  0.25932504440497334
```

```
Confusion Matrix

 [[ 86 385]
 [  1  91]]

F1 Score :  0.3204225352112676

Accuracy Score :  0.31438721136767317
```

```
Confusion Matrix

 [[ 99 370]
 [  0  94]]

F1 Score :  0.33691756272401435

Accuracy Score :  0.3428063943161634
```

```
Confusion Matrix

 [[ 70 407]
 [  1  85]]

F1 Score :  0.29411764705882350

Accuracy Score :  0.2753108348134991

Confusion Matrix

 [[374  99]
 [  8  81]]

F1 Score :  0.6022304832713755

Accuracy Score :  0.8096085409252669
```

DATASET 2

```
Confusion Matrix

 [[  1   0]
 [861 277]]

F1 Score :  0.3915194346289752

Accuracy Score :  0.2440737489025461

Confusion Matrix

 [[909   1]
 [104 125]]

F1 Score :  0.7042253521126761

Accuracy Score :  0.9078138718173837

Confusion Matrix

 [[1134    5]
 [   0    0]]

F1 Score :  0.0

Accuracy Score :  0.9956101843722563
```

DATASET 3

```
Confusion Matrix

 [[878   2]
 [118  34]]

F1 Score :  0.3617021276595745
```

DATASET 4

```
Confusion Matrix

 [[711   1]
 [125 229]]

F1 Score :  0.7842465753424658

Accuracy Score :  0.8818011257035647


Confusion Matrix

 [[750   0]
 [112 204]]

F1 Score :  0.7846153846153847

Accuracy Score :  0.8949343339587242


Confusion Matrix

 [[703   1]
 [115 247]]

F1 Score :  0.8098360655737704

Accuracy Score :  0.8911819887429644
```

ACCURACY

```
print("Mean Accuracy of K Fold : ", np.mean(accuracy1))
print("Mean Accuracy of Stratified K Fold : ", np.mean(accuracy2))
```

DATASET 1

```
Mean Accuracy of K Fold :  0.4002876051655152
Mean Accuracy of Stratified K Fold :  0.4002876051655152
```

DATASET 2

```
Mean Accuracy of K Fold :  0.8275663448497201
Mean Accuracy of Stratified K Fold :  0.8275663448497201
```

DATASET 3

```
Mean Accuracy of K Fold :  nan
Mean Accuracy of Stratified K Fold :  nan
```

DATASET 4

```
Mean Accuracy of K Fold :  0.8997942375956803
Mean Accuracy of Stratified K Fold :  0.8997942375956803
```

LOGISTIC REGRESSION

- It's a classification algorithm, that is used where the response variable is categorical.
- The idea of Logistic Regression is to find a relationship between features and probability of particular outcome.
- This type of a problem is referred to as Binomial Logistic Regression, where the response variable has two values 0 and 1 or pass and fail or true and false.

Cost Function of the Logistic Regression

- Cost Function is a function that measures the performance of a Machine Learning model for given data.
- Cost Function is basically the calculation of the error between predicted values and expected values and presents it in the form of a single real number.

Gradient Descent in Logistic Regression

- Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function that minimizes a cost function (cost).
- The learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a cost function.

IMPLEMENTATION

Training the Model

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(solver = 'newton-cg')
logreg.fit(X_train, Y_train)
```

Prediction the test data

```
Y_pred = logreg.predict(X_test)
```

```
from sklearn.model_selection import GridSearchCV
param = {'solver' : ['newton-cg', 'lbfgs','saga','sag']}
logreg = LogisticRegression()
```

```
grid_search = GridSearchCV(estimator = logreg, param_grid = param, cv = 5)
grid_result = grid_search.fit(X_train, Y_train)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

DATASET 1

```
Best: 0.989344 using {'solver': 'newton-cg'}
```

DATASET 2

```
Best: 0.986608 using {'solver': 'newton-cg'}
```

DATASET 3

```
Best: 0.971636 using {'solver': 'newton-cg'}
```

DATASET 4

```
Best: 0.996952 using {'solver': 'sag'}
```

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(solver = 'newton-cg')
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)

pred_prob2 = logreg.predict_proba(X_test)
```

Accuracy

```
import sklearn.metrics as metrics
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
print("Precision:",metrics.precision_score(Y_test, Y_pred))
print("Recall:",metrics.recall_score(Y_test, Y_pred))
```

DATASET 1

```
Accuracy: 0.99644128113879
Precision: 1.0
Recall: 0.9775280898876404
```

DATASET 2

```
Accuracy: 0.9938488576449912
Precision: 0.0
Recall: 0.0
```

DATASET 3

```
Accuracy: 0.9689922480620154
Precision: 0.910958904109589
Recall: 0.875
```

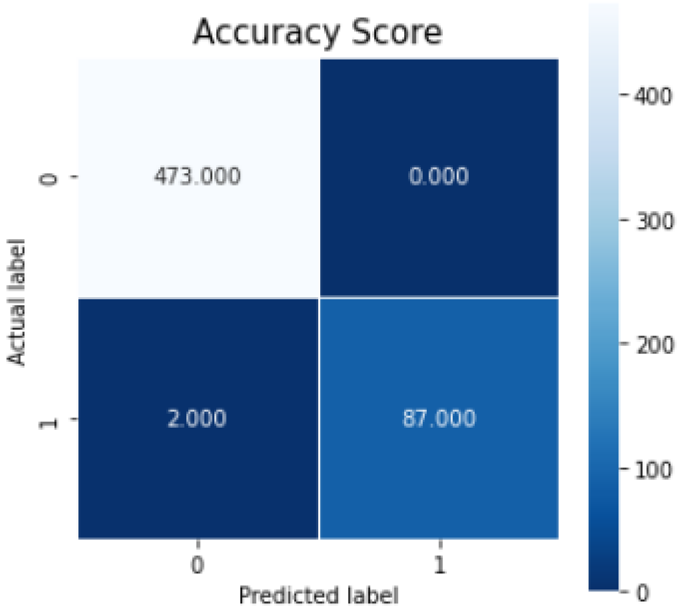
DATASET 4

```
Accuracy: 0.9924882629107982
Precision: 0.9941176470588236
Recall: 0.9825581395348837
```

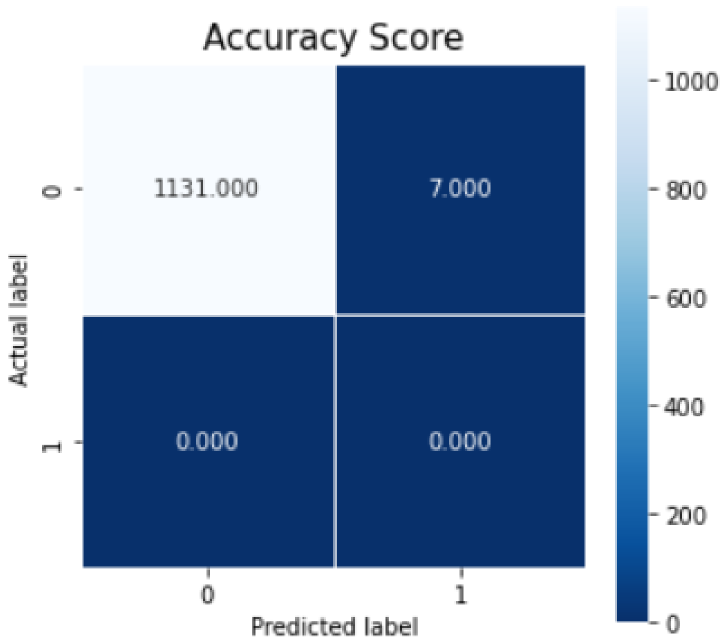
HEAT MAP

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
cm = metrics.confusion_matrix(Y_test, Y_pred)

plt.figure(figsize=(5,5))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score'
plt.title(all_sample_title, size = 15);
```

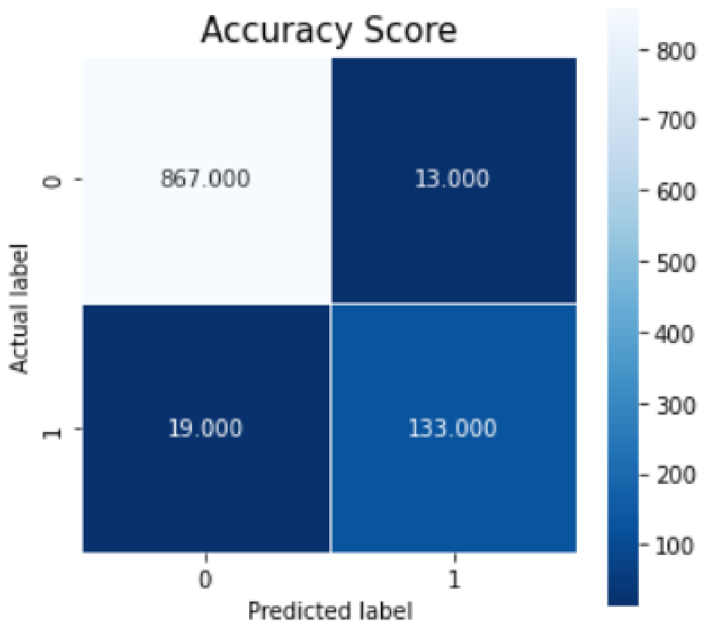
DATASET 1



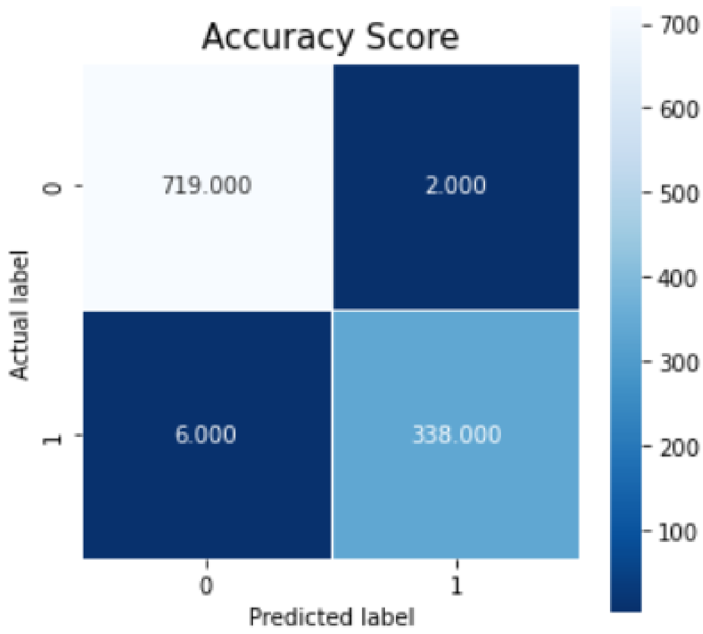
DATASET 2



DATASET 3



DATASET 4



SVM

- Support Vector Machines are supervised learning models for classification and regression problems. They can solve linear and nonlinear problems and work well for many practical problems.
- The algorithm creates a line which separates the classes in case e.g., in a classification problem.
- The goal of the line is to maximizing the margin between the points on either side of the so-called decision line.
- According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors.
- Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin
- The hyperplane for which the margin is maximum is the optimal hyperplane.

Splitting the Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size = 0.2, random_state = 2 )
```

Finding the best kernel trick

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV
param_grid = {
    'kernel' : ['linear', 'rbf']
}
clf = svm.SVC()
grid_search = GridSearchCV(estimator = clf, param_grid = param_grid, cv = 5)
grid_result = grid_search.fit(X_train, Y_train)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

DATASET 1

```
Best: 0.994669 using {'kernel': 'linear'}
```

DATASET 2

```
Best: 0.990777 using {'kernel': 'linear'}
```

DATASET 3

```
Best: 0.976242 using {'kernel': 'linear'}
```

DATASET 4

```
Best: 0.993431 using {'kernel': 'rbf'}
```

Training the model using on the train data

```
from sklearn import svm
clf = svm.SVC(kernel = 'linear', probability=True)
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

Making prediction on the test data

```
pred_prob3 = clf.predict_proba(X_test)
```

Accuracy

```
from sklearn import metrics
print("Accuracy:", metrics.accuracy_score(Y_test, Y_pred))
print("Precision:",metrics.precision_score(Y_test, Y_pred))
print("Recall:",metrics.recall_score(Y_test, Y_pred))
```

DATASET 1

```
Accuracy: 0.9946714031971581
Precision: 1.0
Recall: 0.963855421686747
```

DATASET 2

```
Accuracy: 0.9894644424934153
Precision: 0.9825783972125436
Recall: 0.9757785467128027
```

DATASET 3

```
Accuracy: 0.9748062015503876
Precision: 0.9719626168224299
Recall: 0.8188976377952756
```

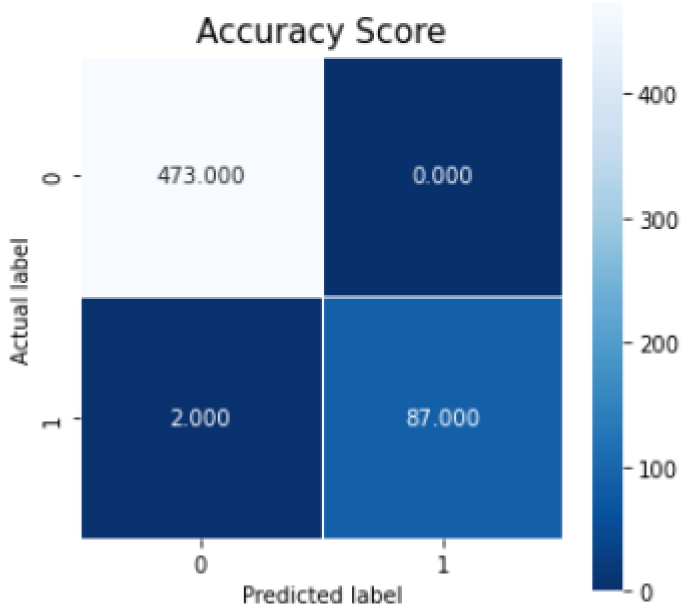
DATASET 4

```
Accuracy: 0.9962476547842402
Precision: 0.9971264367816092
Recall: 0.9914285714285714
```

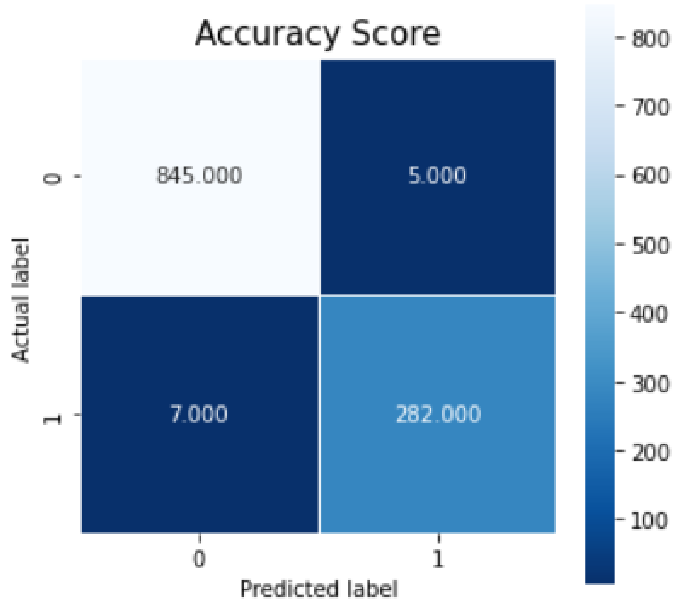
HEAT MAP

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(Y_test, Y_pred)
plt.figure(figsize=(5, 5))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Accuracy Score'
plt.title(all_sample_title, size = 15);
```

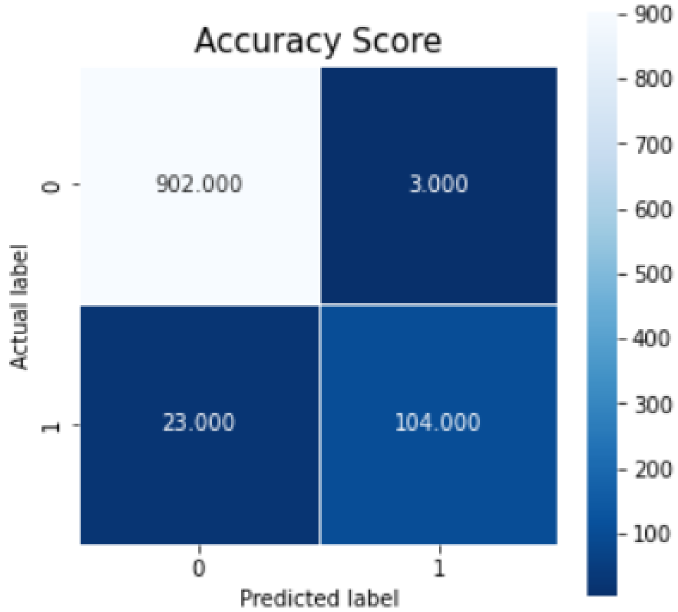
DATASET 1



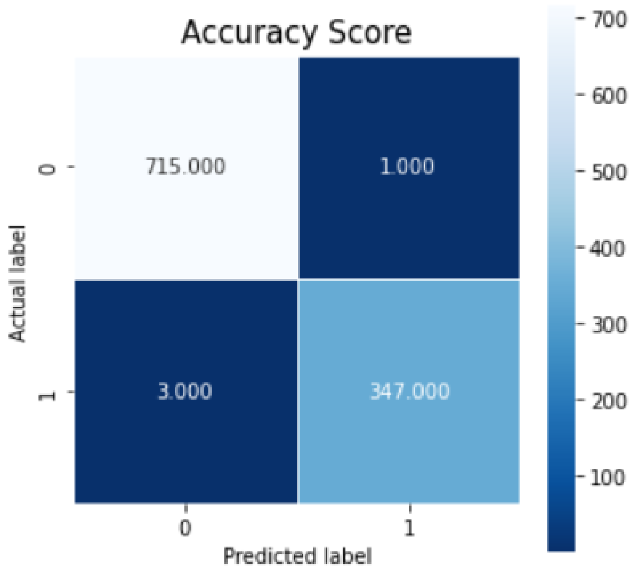
DATASET 2



DATASET 3



DATASET 4



Algorithm Comparisons

from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
cv = CountVectorizer()
tfidf = TfidfVectorizer(max_features=3000)

data

DATASET 1

	subject	message	label	Length	Ham(0) and Spam(1)	Num Characters	Num Words	Num Sentences	Cleaned Text	Cleaned Text Length
0	job posting - apple-iss research center	content - length : 3386 apple-iss research cen...	0	2856	0	2856	584	18	content length 3386 appleiss research center u...	1654
2	query : letter frequencies for text identifica...	i am posting this inquiry for sergei atamas (...	0	1435	0	1435	280	19	post inquiri sergei atama satama umabnet ab um...	909
3	risk	a colleague and i are researching the differin...	0	324	0	324	60	4	colleagu research differ degre risk perceiv ho...	182
4	request book information	earlier this morning i was on the phone with a...	0	1046	0	1046	232	12	earlier morn phone friend mine live south amer...	562
5	call for abstracts : optimality in syntactic t...	content - length : 4437 call for papers is the...	0	4492	0	4492	861	42	content length 4437 call paper best good enoug...	2821
...	...	...	...	...	...	...	...	...	...	...
2888	love your profile - ysuolvpv	hello thanks for stopping by ! ! we have taken...	1	262	1	262	58	6	hello thank stop taken mani new pic made hot n...	117
2889	you have been asked to join kiddin	the list owner of : " kiddin " has invited you...	1	2163	1	2163	533	27	list owner kiddin invit join mail list listbot...	1107
2890	anglicization of composers ' names	judging from the return post , i must have sou...	0	1039	0	1039	211	13	judg return post must sound like kind selfproc...	553
2891	re : 6 . 797 , comparative method : n - ary co...	gotcha ! there are two separate fallacies in t...	0	2949	0	2949	617	20	gotcha two separ fallaci argument nari compari...	1532
2892	re : american - english in australia	hello ! i ' m working on a thesis concerning a...	0	700	0	700	177	14	hello work thesi concern attitud toward americ...	388

2814 rows × 10 columns

DATASET 2

	text	spam	Length	Num Characters	Num Words	Num Sentences	Cleaned Text	Cleaned Text Length	Ham(0) and Spam(1)
0	Subject: naturally irresistible your corporate...	1	1484	1484	325	9	subject natur irresist corpor ident lt realli ...	780	1
1	Subject: the stock trading gunslinger fanny i...	1	598	598	90	1	subject stock trade gunsling fanni merril muzo...	460	1
2	Subject: unbelievable new homes made easy im ...	1	448	448	88	4	subject unbeliev new home made easi im want sh...	254	1
3	Subject: 4 color printing special request add...	1	500	500	99	5	subject 4 color print special request addit in...	332	1
4	Subject: do not have money , get software cds ...	1	235	235	53	5	subject money get softwar cd softwar compat ai...	120	1
...	...	...	...	...	...	...	...	...	...
5721	Subject: re : research and development charges...	0	1189	1189	298	6	subject re research develop charg gpg forward ...	720	0
5722	Subject: re : receipts from visit jim , than...	0	1167	1167	245	20	subject re receipt visit jim thank invit visit...	773	0
5723	Subject: re : enron case study update wow ! a...	0	2131	2131	516	18	subject re enron case studi updat wow day supe...	1228	0
5724	Subject: re : interest david , please , call...	0	1060	1060	277	6	subject re interest david pleas call shirley c...	634	0
5725	Subject: news : aurora 5 . 2 update aurora ve...	0	2331	2331	445	29	subject news aurora 5 2 updat aurora version 5...	1439	0

5693 rows × 9 columns

DATASET 3

	Category	Message	Length	Num Characters	Num Words	Num Sentences	Cleaned Text	Cleaned Text Length
0	0	Go until jurong point, crazy.. Available only ...	111	111	24	2	go jurong point crazi avail bugi n great world...	76
1	0	Ok lar... Joking wif u oni...	29	29	8	2	ok lar joke wif u oni	21
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	155	155	37	2	free entri 2 wkli comp win fa cup final tkt 21...	131
3	0	U dun say so early hor... U c already then say...	49	49	13	1	u dun say earli hor u c alreadi say	35
4	0	Nah I don't think he goes to usf, he lives aro...	61	61	15	1	nah dont think goe usf live around though	41
...	...	...	...	...	...	...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...	160	160	35	4	2nd time tri 2 contact u u pound prize 2 claim...	100
5568	0	Will ü b going to esplanade fr home?	36	36	9	1	ü b go esplanad fr home	23
5569	0	Pity, * was in mood for that. So...any other s...	57	57	15	2	piti mood soani suggest	23
5570	0	The guy did some bitching but I acted like i'd...	125	125	27	1	guy bitch act like id interest buy someth els ...	68
5571	0	Rofl. Its true to its name	26	26	7	2	rofl true name	14

DATASET 4

	text	target	Length	Num Characters	Num Words	Num Sentences	Cleaned Text	Cleaned Text Length	Ham(0) and Spam(1)
0	From ilug-admin@linux.ie Mon Jul 29 11:28:02 2...	0	4098	4098	818	16	ilugadminlinuxi mon jul 29 112802 2002 returnp...	2771	0
1	From gort44@excite.com Mon Jun 24 17:54:21 200...	1	2189	2189	520	13	gort44excitecom mon jun 24 175421 2002 returnp...	1366	1
2	From fork-admin@xent.com Mon Jul 29 11:39:57 2...	1	3598	3598	640	11	forkadminxentcom mon jul 29 113957 2002 return...	2596	1
3	From dcm123@btamail.net.cn Mon Jun 24 17:49:23...	1	1918	1918	451	1	dcm123btamailnetcn mon jun 24 174923 2002 retu...	1458	1
4	From ilug-admin@linux.ie Mon Aug 19 11:02:47 2...	0	3060	3060	620	7	ilugadminlinuxi mon aug 19 110247 2002 returnp...	2192	0
...	...	...	...	...	...	...	...	...	...
5791	From ilug-admin@linux.ie Mon Jul 22 18:12:45 2...	0	3732	3732	703	12	ilugadminlinuxi mon jul 22 181245 2002 returnp...	2645	0
5792	From fork-admin@xent.com Mon Oct 7 20:37:02 20...	0	3334	3334	687	18	forkadminxentcom mon oct 7 203702 2002 returnp...	2300	0
5793	Received: from hq.pro-ns.net (localhost [127.0...	1	5050	5050	1134	5	receiv hqpronsnet localhost 127001 hqpronsnet ...	3651	1
5794	From razor-users-admin@lists.sourceforge.net T...	0	8068	8068	1460	18	razorusersadminlistssourceforgenet thu sep 12 ...	6201	0
5795	From rssfeeds@jmason.org Mon Sep 30 13:44:10 2...	0	1084	1084	197	2	rssfeedsjmasonorg mon sep 30 134410 2002 retur...	811	0

5329 rows × 9 columns

```
X = tfidf.fit_transform(data['Cleaned Text']).toarray()
y = data['target'].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=2)

svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier()
dtc = DecisionTreeClassifier(max_depth=5)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=50, random_state=2)
bc = BaggingClassifier(n_estimators=50, random_state=2)

clfs = {
    'SVM SVC' : svc,
    'KNN' : knc,
    'Decision Tree': dtc,
    'Logistic Regression': lrc,
    'Random Forest': rfc,
    'Bagging Classifier': bc,
}

def train_classifier(clf, X_train, y_train, X_test, y_test):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    return accuracy, precision

accuracy_scores = []
```

```
precision_scores = []

for name, clf in clfs.items():

    current_accuracy, current_precision = train_classifier(clf, X_train, y_train, X_test, y_test)

    print("For ", name)
    print("Accuracy - ", current_accuracy)
    print("Precision - ", current_precision, "\n")
    accuracy_scores.append(current_accuracy)
    precision_scores.append(current_precision)
```

DATASET 1

```
For  SVM SVC
Accuracy -  0.9946714031971581
Precision -  1.0

For  KNN
Accuracy -  0.9875666074600356
Precision -  0.9871794871794872

For  Decision Tree
Accuracy -  0.9715808170515098
Precision -  0.958904109589041

For  Logistic Regression
Accuracy -  0.9786856127886323
Precision -  0.9863013698630136

For  Random Forest
Accuracy -  0.9928952042628775
Precision -  1.0

For  Bagging Classifier
Accuracy -  0.9840142095914742
Precision -  0.9743589743589743
```

DATASET 2

```
For  SVM SVC
Accuracy -  0.990342405618964
Precision -  0.9826388888888888

For  KNN
Accuracy -  0.9798068481123793
Precision -  0.9889705882352942

For  Decision Tree
Accuracy -  0.9244951712028094
Precision -  0.7976539589442815

For  Logistic Regression
Accuracy -  0.9727831431079894
Precision -  0.9708029197080292

For  Random Forest
Accuracy -  0.9798068481123793
Precision -  0.9925925925925926

For  Bagging Classifier
Accuracy -  0.9631255487269534
Precision -  0.9273356401384083
```

DATASET 4

```
For  SVM SVC
Accuracy -  0.9953095684803002
Precision -  0.9971181556195965

For  KNN
Accuracy -  0.9681050656660413
Precision -  0.9817073170731707

For  Decision Tree
Accuracy -  0.9784240150093808
Precision -  0.9794721407624634

For  Logistic Regression
Accuracy -  0.9906191369606003
Precision -  0.9941860465116279

For  Random Forest
Accuracy -  0.9887429643527205
Precision -  0.9912790697674418

For  Bagging Classifier
Accuracy -  0.9849906191369606
Precision -  0.9826589595375722
```

```
performance_df = pd.DataFrame({'Algorithm':clfs.keys(),'Accuracy':accuracy_scores,'Precision':precision_scores}).sort_values('Precision',ascending=False)
performance_df
```

DATASET 1

	Algorithm	Accuracy	Precision
0	SVM SVC	0.994671	1.000000
4	Random Forest	0.992895	1.000000
1	KNN	0.987567	0.987179
3	Logistic Regression	0.978686	0.986301
5	Bagging Classifier	0.984014	0.974359
2	Decision Tree	0.971581	0.958904

DATASET 2

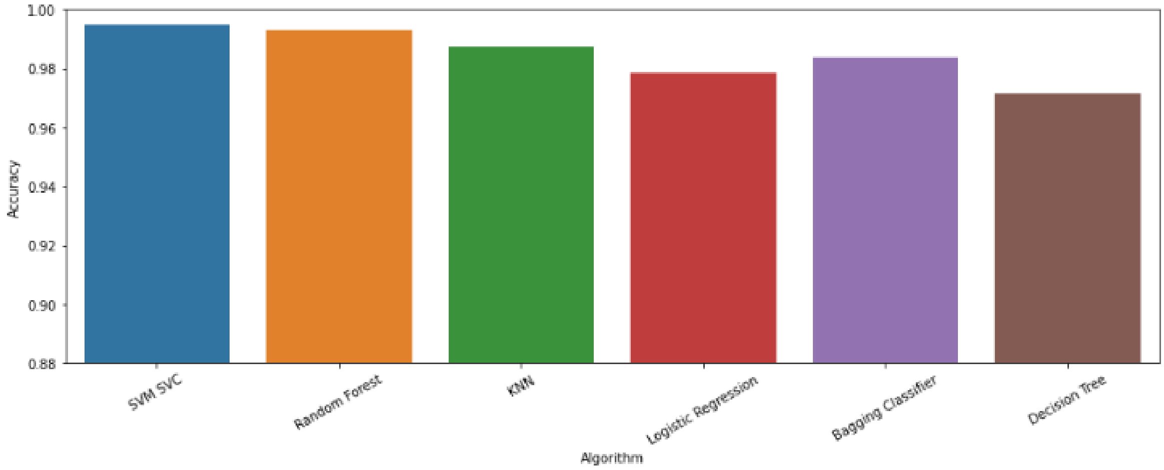
	Algorithm	Accuracy	Precision
4	Random Forest	0.979807	0.992593
1	KNN	0.979807	0.988971
0	SVM SVC	0.990342	0.982639
3	Logistic Regression	0.972783	0.970803
5	Bagging Classifier	0.963126	0.927336
2	Decision Tree	0.924495	0.797654

DATASET 4

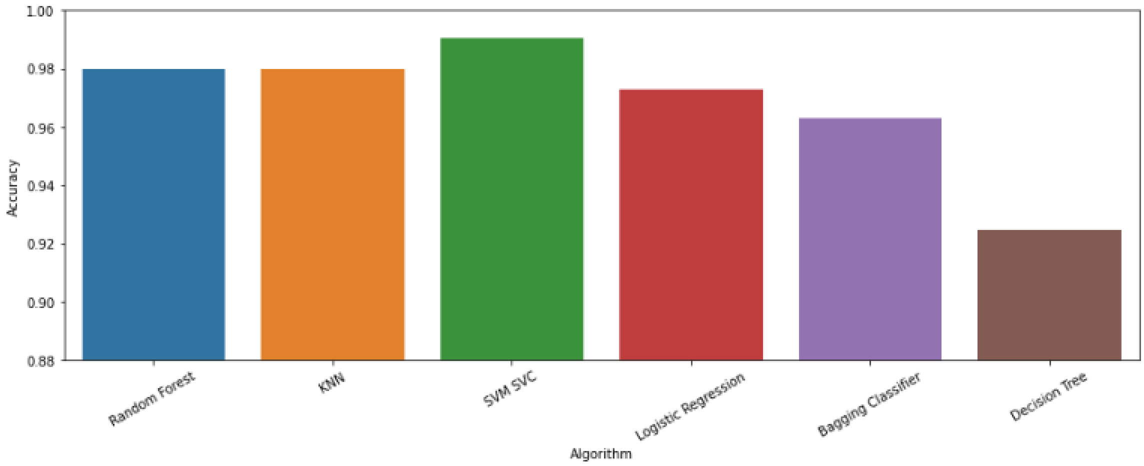
	Algorithm	Accuracy	Precision
0	SVM SVC	0.995310	0.997118
3	Logistic Regression	0.990619	0.994186
4	Random Forest	0.988743	0.991279
5	Bagging Classifier	0.984991	0.982659
1	KNN	0.968105	0.981707
2	Decision Tree	0.978424	0.979472

```
plt.figure(figsize=(15, 5));
sns.barplot(x = 'Algorithm', y ='Accuracy', data = performance_df);
plt.ylim(0.88, 1.0);
plt.xticks(rotation = 30);
```

DATASET 1



DATASET 2



DATASET 4