Email Spam Detection Using Machine Learning

1. **Data Collection**

importing pandas library to read the data

```
import pandas as pd
```

loading the dataset from Google Sheets

```
data=pd.read_csv('https://docs.google.com/spreadsheets/d/1egtZgqs61fgCmI_eHxqPePYlEMAACDW8lEAjfC9cgi4/export?format=csv&gid=332598977')
```

Reading the data

This data consists of 2 columns - v1 which is the category column that labels the mail content as ham or spam; v2 is the column containing the mail/messages. For better understanding, the columns are renamed - v1-> Category,v2 -> Message

```
data.rename(columns={'v1':'Category','v2':'Message'},inplace=True)
```

```
data.head(3)
```

	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...

```
data.tail(3)
```

	Category	Message
5569	0	Pity, * was in mood for that. So...any other s...
5570	0	The guy did some bitching but I acted like i'd...
5571	0	Rofl. Its true to its name

2. **Data Organization**

```
data.shape # gives information about the data i.e, total number of rows and columns
```

```
(5572, 2)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

From the above info, it shows that there is no null values present

```
data.describe() # gives the summary of the data
```

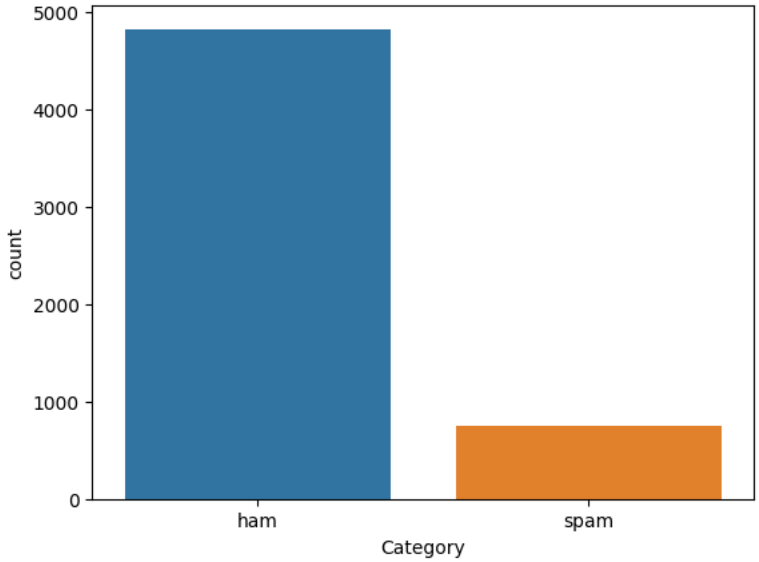
3. **Data Visualization**

importing necessary libraries for Data Visualization

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.countplot(data=data,x=data['Category'])
```

```
<Axes: xlabel='Category', ylabel='count'>
```



The above graph shows that number of ham mails is more than spam ones.

```
data.groupby('Category')['Category'].count()
```

```
Category
ham     4825
spam     747
Name: Category, dtype: int64
```

```
# Total number of ham mails = 4825
# Total number of spam mails = 747
#Total mails= 5572
Ham = (4825/5572)*100
print('Ham :-',Ham)

Spam=(747/5572)*100
print('Spam :-',Spam)

# Ham is rounded to 87 and Spam is rounded to 13 for more convinience
```

```
Ham :- 86.59368269921033
Spam :- 13.406317300789663
```

```
plt.pie((87,13),labels=['Ham','Spam'])
plt.title('Mails by Category')
plt.show()
```

4. **Data Preparation**

As the category data which is needed for detection has only string values, replacement by numbers is to be done.

0 for ham, 1 for spam

```
data['Category']=data['Category'].replace({'ham':0,'spam':1})
```

Dividing the dataset into input and output columns, i.e input consists of independent variables (Message) and output has dependent variables(Category).

```
# input column
x= data['Message']

# output column
y=data['Category']
```

importing needed libraries

```
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)
```

Feature extraction:-

For machine learning, using numerical data is more easy because process is faster. So, the string data is to be converted to numbers. This is done using the method of Feature Extraction, and the module TfidfVectorizer.

```
from sklearn.feature_extraction.text import TfidfVectorizer

new= TfidfVectorizer(min_df=1,stop_words='english',lowercase=True)
```

- min_df is for checking if the word's score is less than 1, can be ignored
- stop_words = unnecessary words to be ignored
- lowercase= words in lower case to be included for analysis

```
x_train_new=new.fit_transform(x_train) # input train values implied with the feature extraction is trained for the model building

x_test_new=new.transform(x_test) # since its a test data, its not fitted.
```

5. **Model Building**

   Since, the dataset is composed of dependent variable which falls under binary category, models like Logistic Regression, Decision Tree Classifier, Support Vector Classifier are used.

(1) Logistic Regression

```
from sklearn.linear_model import LogisticRegression

model_1=LogisticRegression() # defining the model

model_1.fit(x_train_new,y_train) # training the data
```

```
▾ LogisticRegression
LogisticRegression()
```

Model evaluation for (1)

```
score_1=model_1.score(x_train_new,y_train)
print('Accuracy Score for Logistic Regression model with training data =',score_1)
score_2=model_1.score(x_test_new,y_test)
print('Accuracy Score for Logistic Regression model with test data =',score_2)
```

```
Accuracy Score for Logistic Regression model with training data = 0.9674669059905766
Accuracy Score for Logistic Regression model with test data = 0.9524663677130045
```

(2) DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
```

```
model_2=DecisionTreeClassifier()
```

```
model_2.fit(x_train_new,y_train)
```

```
DecisionTreeClassifier()
```

Model Evaluation for (2)

```
s_1=model_2.score(x_train_new,y_train)
print('Accuracy Score for Decision Tree Classification model with training data =',s_1)
s_2=model_2.score(x_test_new,y_test)
print('Accuracy Score for Decision Tree Classification model with test data =',s_2)
```

```
Accuracy Score for Decision Tree Classification model with training data = 1.0
Accuracy Score for Decision Tree Classification model with test data = 0.9623318385650225
```

(3) Support Vector Classifier

```
from sklearn.svm import SVC
```

```
model_3=SVC()
model_3.fit(x_train_new,y_train)
```

```
SVC()
```

Model Evaluation for (3)

```
s_a=model_3.score(x_train_new,y_train)
print('Accuracy Score for Support Vector Classification model with training data =',s_a)
s_b=model_3.score(x_test_new,y_test)
print('Accuracy Score for Support Vector Classification model with test data =',s_b)
```

```
Accuracy Score for Support Vector Classification model with training data = 0.9977563383441777
Accuracy Score for Support Vector Classification model with test data = 0.97847533632287
```

Classification Reports for all 3 models

```
from sklearn.metrics import classification_report # importing the library used for printing classification reports
```

```
p1=model_1.predict(x_test_new)
print(' 1. Classification Report for Model 1 - Logistic Regression:-' ,'\n ', classification_report(y_test,p1))

pred1=model_2.predict(x_test_new)
print('2. Classification Report for Model 2 -  Decision Tree Classification:-' ,'\n ', classification_report(y_test,pred1))

pred_a=model_3.predict(x_test_new)
print('3. Classification Report for Model 3 -  SVC:-' ,'\n ', classification_report(y_test,pred_a))
```

```
 1. Classification Report for Model 1 - Logistic Regression:-
               precision    recall  f1-score   support

           0       0.95      1.00      0.97       948
           1       0.99      0.69      0.81       167
```

```
    accuracy                           0.95      1115
   macro avg       0.97      0.84      0.89      1115
weighted avg       0.95      0.95      0.95      1115

2. Classification Report for Model 2 -  Decision Tree Classification:-
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       948
           1       0.91      0.83      0.87       167

    accuracy                           0.96      1115
   macro avg       0.94      0.91      0.92      1115
weighted avg       0.96      0.96      0.96      1115

3. Classification Report for Model 3 -  SVC:-
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       948
           1       1.00      0.86      0.92       167

    accuracy                           0.98      1115
   macro avg       0.99      0.93      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

From the above classification reports of the 3 models used for training the machine in order to detect the spam mails, "model_3" i.e, the model built by using the algorithm Support Vector Classifier(SVC) is found to be the best one.

```
import pickle # modeule used for loading and storing files

pickle.dump(model_3,open('/content/drive/MyDrive/ONE/Model_for_Spamdetection.pkl','wb'))
```

Hence, the model built using SVC alogrithm is saved.